27th Australasian Transport Research Forum, Adelaide, 29 September – 1 October 2004

Paper title: **An algorithm for reducing detection load in transportation oriented DSMS**

Author(s) name(s): Hideto Ikeda*, Nikolaos Vogiatzis**, Waskitho Wibisono*, Yu He*

Organisation(s): *Cyberspace Technology Laboratory: Department of Computer Science, Ritsumeikan University
**Transport Systems Centre: University of South Australia

Contact details:
Postal address: 1-1-1 Noji-higashi, Kusastu, Shiga 525-8577 JAPAN

Telephone: +81-775-61-2691
Facsimile: +81-775-61-2669
email: hikeda@cyber.is.ritsumei.ac.jp

Abstract (200 words):
In order to integrate transportation systems such as road traffic, rail signalling, bus management, air traffic control and shipping lane systems, it is important to establish a holistic concept design and technology inventory before beginning. These systems are all ubiquitous in today's societies; they impact on our daily personal and commercial lives, and as such we need to ensure that if they are integrated, that it is done so in a seamless and secure way. At the heart of the integration process is the 3-Layer Object Model (3LOM). It consists of four primary components in three layers: 1) local units at the lowest layer; 2) a distributed database management system in the middle layer; and 3) a knowledge management and central control system at the highest layer. In this paper we continue the design of a future integrated transport management and will do so by concentrating on the middle layer; the conduit of 3LOM. It is responsible for the data acquisition management for the front-end devices in the bottom layer, the data management function, and reports refined information to the knowledge management system in the top layer. Specifically, we will propose an algorithm that allows the development of the middle layer of the 3LOM, and how it needs to function in order to succeed.

Introduction

Conventional database management systems (DBMS) are designed to manage and process business data(Vogiatzis, Ikeda, Woolley and He, 2003, Ikeda, 2003, Ikeda, Vogiatzis, Wibisono, Mojarrabi and Woolley, 2004, Ikeda and Vogiatzis, 2003). They, by their very nature, do not reflect the object character of the physical world; rather they natively prefer the relationships between processes.

In a traffic management context, these systems need to be able to collect data every second (and more specifically at every event which may well occur in less than a second from the previous event), whether or not they report it back to the management system(Vogiatzis, Ikeda and Wibisono, 2004). SCATS for example reports traffic volumes to a flat file (a physical disc file that contains collections of data, typically comma separated, which make up records) every 5 or 15 minutes depending on the setting of the system. However those counts are the number of vehicles triggering an event during the prescribed time-frame. Current database management systems such as Relational Database Management Systems (RDBMS) are not capable to manage streaming data, as is needed in the traffic management sense.

Ikeda, Vogiatzis et al (2004) are currently in the process of designing a new generation traffic management and traffic micro-simulation tool, and they identified a significant challenge in such a system, being how does one make automated and human-guided decisions in a time-frame that would reduce the number of invalid decisions for the movement of the vehicles because of new events occurring that render any previous information useless. They concede that one will probably never be able to make decisions so fast as to eliminate the problem completely (not at least much into the future), and until such time as one can, they have identified the need for a technique now that will minimise this problem.

Thus, a new data management system known as a Data Stream Management System (DSMS) is needed and is has been developed by researchers at Stanford (Babu, Subramanian and Widom, 2001) and an implementation for this new traffic management/micro-simulation tool is proposed for the process management of the data streams for integrated transport systems. This does not suggest that conventional DBMS are no longer of any import within a transport system; rather the system functions *sine qua non* both a DBMS *and* a DSMS. One of the important features of a DSMS is its ability to query a database whilst looking over a live stream of data, such as one would find in a data traffic network (Babu *et al.*, 2001). Babcock, Babu, Datar, Motwani and Widom (2002) report that queries in DSMS are of two distinct types, being either 1) one-time queries (queries performed on a historical data set within a database); or 2) continuous queries (queries performed on a dynamic set of data entering the database as the query is being performed without loss of validity), each of which can be either predefined or ad hoc queries. Heuristically predefined queries are continuous in nature; however scheduled one-time queries can also be predefined.

In the context of systems which monitor external activities, such as traffic management systems, predefined queries allow for specified detection criteria to be defined, observed, and analysed as part of a monitoring system. For example, a traffic control system has to detect various potentially hazardous conditions such as traffic accidents, fire, earthquakes,

air-pollution, and road flooding and then formulate complex but well-organised responses quickly.

The challenge that exists is that of finding the most efficient manner by which all the data being presented to the DSMS from the traffic network can be analysed. The load on the analysing computers would be extraordinarily high if every event were to be analysed. Unfortunately, there does not seem to be a great deal in the literature that can suggest an efficient algorithm that will allow traffic management systems, and IMAGINATION(Vogiatzis *et al*, 2003, Vogiatzis *et al*, 2004) in the long term, to be able to detect important conditions from within the data stream.

Alert (Schreier, Pirahesh, Agrawal and Mohan, 1991) is an architecture that provides a mechanism for implementing event-condition-action style database triggers (software commands that are issued as soon as an event is detected by a database management system) in a conventional SQL (Structured Query Language: a computer language used to make the querying of data within a database more efficient for the programmer) database, by using continuous queries defined over specially created *append only* database tables. The Aurora Project (Brandeis University, Brown University and Massachusetts Institute of Technology, 2003) is in the process of developing a new type of data processing system specifically targeted towards the monitoring of stream data within monitoring applications. Specifically, this project:

> *'... addresses three broad application types in a single, unique framework: 1) Real-time monitoring applications continuously monitor the present state of the world and are, thus, interested in the most current data as it arrives from the environment. In these applications, there is little or no need (or time) to store such data; 2) Archival applications are typically interested in the past. They are primarily concerned with processing large amounts of finite data stored in a time-series repository; and 3) Spanning applications involve both the present and past states of the world, requiring combining and comparing incoming live data and stored historical data. These applications are the most demanding as there is a need to balance real-time requirements with efficient processing of large amounts of disk-resident data…' (Brandeis University et al, 2003)*

One can easily see that the traffic management, and specifically the transport system integration process, is a perfect example of all three of these three broad application types. At the core of the Aurora system, consists of a large network of triggers. However, these two research themes only try to provide mechanisms to solve detection problem within conventional DBMS or a new processing system. There is no research being conducted with relation to solving the problem of how to reduce the system load for detecting predefined serious conditions over all of data streams and respond them in real time, as is required, ultimately, within a transport environment.

The Detection Problem

Detection problem from the application perspective

In most of data stream applications, a group of sensors is responsible for collecting data relating to numerous environmental (as a rule, we refer to environment as being more than

just natural ecology, but rather the transport eco-system as a whole, which includes the natural environment) states in real time. An application system analyses the collected data in order to identify if any serious conditions have occurred which either have affected or will affect the environment in real time. Then, the application system employs some sort of mechanism(s) to control environment, again in real time, in order to ensure environmental recovery and to return the environment to its nominal status. For example, within the context of a transportation system in a city setting, an application system would usually collect traffic information through many sensors, such as vehicle/speed sensors, noise level meters, and airborne pollution sensors and so on. In fact, a human being such as the police can be considered as a kind of traffic sensor that provides real time information relating to traffic conditions.

The main challenge is to develop a problem detection system that can identify any problems within the traffic network (such as traffic congestion) in an automated way, and then to inform all the appropriate controllers of the problem, a project solution, and the methodology required to solve the problem. These controllers, however, are not only electronic devices. They can be human controllers, and the information presented to them may be as a series of alternatives upon which the human controllers can make a decision on how to proceed. Once a decision has been reached, then the detection system can provide a detailed methodology so as to ensure that a uniform and repeatable action plan can be executed. The rules within the system are referred to as *condition-action* rules, or more commonly as *control rules*.

In the application of such a system in transport, one finds that the most frequent *operation* is that of incident detection, therefore, it is important that in designing an integrated transport system, one needs to address the issue of system load to the data stream management system, being a DSMS, so as to ensure that the system can manage the detection operation for *all* data streams within the application. Remember, that a data stream can be any type of device, human or electronic, that provides information the management system which needs to be analysed and acted upon in real-time.

We can see this by realising that in a typical sphere of operation, there could be literally thousands of sensors that monitor their environment continuously, with data arriving at the DSMS concurrently and non-linearly. If the DSMS attempts to analyse each data packet against hundreds of condition-rules in real-time, it will place an unduly heavy load on the detection system, rendering it useless as it will not be able to respond within a time-frame that would provide the optimum value to network users. That being the case, it is necessary to identify a more efficient approach, and this is what we will explore now.

Control Rules

We begin by formalising the *detection problem* by first defining some elementary *control-rules* that we will use later in this paper.

A collection of control rules is in fact a collection of *control sentences*. Each control sentence describes both a critical condition that needs to be detected and a corresponding action that needs to be performed. We will use the formal ‘language’ definition format used to describe language context free grammars (The Free Dictionary.com, 2004):

$$CR := \{< control >,...\}$$

The term <control> represents a pair being a *condition* and an *action*. The sentence inform us/the system that when a condition is satisfied, that a corresponding action needs to be executed. A control term can be described as:

$$< control >:= if\ < condition > then < action >$$

The term <condition> is defined as a logical combination of simple conditions as follows:

$$\begin{aligned} &< condition >:=< simple\_condition > \\ &|\,!< condition > \\ &|< condition >< logical\_op >< condition > \\ &|\,(< condition >) \end{aligned}$$

$$\begin{aligned} &< simple\_condition > \\ &:=< sensor >< comparison\_op >< value > \end{aligned}$$

To aid the reader, the above notation reads: <condition> is composed of either a <simple condition> or (| is the logical symbol for 'or') the negation of an existing <condition> (! is the logical symbol for 'not') or is composed of the result of the application of a logical operator on two conditions, or finally, a collection of conditions.

We find that <action> can be defined in exactly the same manner as follows:

$$\begin{aligned} &< action >:=< simple\_action > \\ &|< action >< logical\_op >< action > \\ &|\,(< action >) \end{aligned}$$

$$\begin{aligned} &< simple\_action >:=< action\_name > \\ &[(< parameter >_1, < parameter >_2 \ldots < parameter >_3)] \end{aligned}$$

$$< comparision\_op >:= > | < | = | \neq | \leq | \geq$$

$$< logical\_op >:= and | or$$

Example of Control-Rules as applied in a traffic management context

In order to assist the reader in understanding the definition of control rules, we give an example of how knowledge is defined within the system:

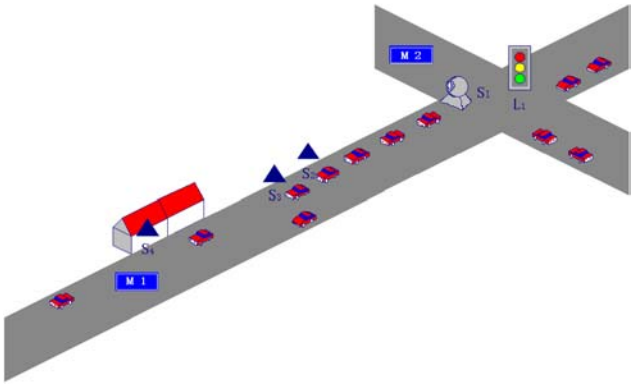

Figure 1 Example of real-time environment control

Figure 1 illustrates a crossing of road M_1 and M_2. There are 4 sensors in this area with sensor S_1 being a video camera, S_2 is a vehicle speed sensor for monitoring vehicle speeds on road M_1. S_3 is a sensor which calculates the queue length and S_4 is an in-house fire alarm (sensor) that detects the occurrence a fire in the house. For the purpose of the example, we want to define two 'emergency' conditions which will affect the flow of traffic on the roads, and these will be traffic congestion and a fire outbreak.

We consider two ways of detecting if congestion has occurred:

1. The presence detectors show that the vehicle spacing is less than ½ a metre;
2. Average vehicle speed in a 15 minute period is calculated as being less that 5 km/h and a camera reports 'abnormal'.

To solve this problem, it may be decided to increase the green time at controller L1 on the affected movement direction, but to keep the total cycle time the same for the intersection. For example, we could describe this *control-rule* as:

Knowledge 1: ***if*** *((S_3 < ½ metre)* ***or*** *(S_1 = 'abnormal' and S_2 < 5))* ***then***

```
{
    change_traffic_signal_phase(L1, 10, false, false, true);
}
```

Where c*hange_traffic_signal_phase(L1, 10, false, false, true)* means change the signal phase for controller L1, increase green time by 10, do not change the cycle time, do not alter yellow phase, adjust effective red.

In the case of a house fire we may expect that we need to detect the following criteria:

1. Fire sensor S4 has triggered;
2. The average speed of vehicles on M1 is less than 10 km/h in a 15 minute period.

The actions that we may want to initiate as a result of the condition being met might be:

Knowledge 2: ***if*** *((S_2 < 10)* ***and*** *(S_4 =true))* ***then***

```
{
    close_road(M1) and call_fire_department(M1);
}
```

Canonical Condition

When considering the condition clauses, we note that they are a kind of *predicate logic*. Therefore, we are able to transform each of the condition clauses into its disjunctive form,

being of simple conjunctive forms. The simple conjunctive condition is referred to as the *canonical condition*.

With the canonical condition, each logical operation is only a conjunction, meaning that when all simple conditions in canonical form are true, then the canonical condition is also true. This suggests that we can monitor one data stream from the traffic network to look for the canonical condition, and thus lowering the system load.

So, when a selected simple condition is evaluated to being true, we then begin looking at other data streams to identify if the canonical condition is true or not.
Looking at the two knowledge statements above, knowledge 1 and 2, we observe that knowledge 1 is in disjunctive normal form because it is composed of two canonical conditions:
c_1: $S_3 <$ ½ metre
c_2: S_1 = 'abnormal' and $S_2 < 5$

Whereas knowledge 2 is a canonical condition:
c_3: $S_2 < 10$ **and** S_4 =true

Definition of the Detection Problem

Even though the detection problem can be formalised in many different ways, we will formalise it with relation to a DSMS, as ultimately a DSMS will be used for the development of a real time transport environment controller. Unfortunately, this can not be done without providing the reader with some fundamental mathematics. We encourage the reader to read and understand the mathematics presented below as it will enhance their understanding of the algorithm, however, it is possible to move directly to the algorithm if the reader so chooses.

Definition 1 (specification of detection problem). The detection is defined as a 4-tuple <*S, C, d, l*> which satisfies the following conditions:

(1) *S* is a finite set which is the set of all data streams,
(2) *C* is a finite set which is the set of all canonical conditions,
(3) *d* is a function from $C \times S$ ->{0,1}, where $d(c_i, s_j) = 1$ means that the canonical condition c_i needs to monitor data stream s_j permanently and otherwise not.
(4) *l* is a function *S*-> *R* ,where *R* is the set of non-negative real numbers and $l(s_j)$ is detection load $l(s_j)$ for monitor data stream s_j.

Using the example from figure 1 we see that:

S: $\{s_1, s_2, s_3, s_4\}$
C: $\{c_1, c_2, c_3\}$

$d(c_1, s_3) = d(c_2, s_1) = d(c_2, s_2) = d(c_3, s_2) = d(c_3, s_4) = 1$
$d(c_1, s_1) = d(c_1, s_2) = d(c_1, s_4) = d(c_2, s_3) = d(c_2, s_4) = d(c_3, s_1) = d(c_3, s_3) = 0$

Assume that the detecting load $l(s_j)$ for monitoring data streams s_1, s_2, s_3, s_4 is 4, 1, 2, 3 respectively. Then *l* by Definition 1 is:

$l(s_1)=4$, $l(s_2)=1$, $l(s_3)=2$, $l(s_4)=3$

In terms of detection problem defined above, a solution, the set of permanent detection stream, can be defined as follows:

Definition 2 (Solution of detection problem). Given 4-tuple <*S, C, d, l*>, which satisfy $d(c_i, s_j) \subset S \times C$ and *l: S->R*, find $Z : S_0 \subset 2^s$ satisfy:

$\widetilde{l}(Z) \le \widetilde{l}(S')$, for any $S' \subset 2^s$

$\left|(Z \times \{C\}) \cap D\right| \ge 1$, for $\forall c \subset C$, where $\widetilde{l}(S') = \sum_{s \in S'} l(s)$ for any $S' \subset 2^s$

The data streams which form Z are referred to as *permanently monitored data streams (PMDS)*, the other data streams are referred to as *eventually analysed data streams (EADS)*. It may be the case that we only monitor the PMDS until such time as a positive event is identified, after which we may begin monitoring the EADS. This allows us to detect whether or not the condition identified is a canonical condition.

Finding a solution to the detection problem

In this next section, we begin by discussing a number of preliminary topics that will then lead us to the reduction algorithm.

Integer linear programming

It is important to begin by introducing a very famous optimisation problem known as the *integer linear programming problem(Gomory, 1963)*. To clarify, linear programming deals with a class of optimisation problems where both the objective functions to be optimised and all the constraints are linear in terms of the decision variables.

Specifically, an *Integer Linear Program* (LP) is a problem <A, B, C> that can be expressed as follows (the so-called Standard Form):

1. Minimize the value of CX
2. Satisfy the condition $AX = B$, $X \ge 0$
3. x_i is an integer, for each $x_i \in X$

Where X is the vector of variables to be solved for, A is a matrix of known coefficients, and C and B are vectors of known coefficients. The expression CX is called the objective function, and the equations $AX = B$ are called the constraints. All these entities must have consistent dimensions, of course, and you can add "transpose" symbols to taste. The matrix A is generally not square, hence you don't solve an LP by just inverting A. Usually A has more columns than rows, and $AX = B$ is therefore quite likely to be under-determined, leaving great latitude in the choice of x with which to minimize CX.

Minimum-Weight Covering Problem

In order to provide an algorithm for the detection problem, we shall define a mathematical problem called minimum weight covering problem.

Definition 1 (Minimum-Weight Covering Problem):
A 3-tuple $< C, S, w >$ is called a minimum-weight covering problem or MWCP in short when the following conditions are satisfied:

1. $C = \{c_1, c_2, ..., c_n\}$ is a non-empty finite set
2. $S = \{s_1, s_2, ...s_m\}$ is a non-empty set of subsets in C
3. $\bigcup_{i=1}^{m} s_i = C$
4. $w : S \to R_+$ a positive function, where R_+ is the set of all positive real numbers.

Definition 2 (A solution of MWCP):
For MWCP $< C, S, w >$, a subset A of S is called a solution of $< C, S, w >$ if it satisfies the following conditions:

1. $\bigcup_{s \in A} s = C$
2. $\sum_{s \in A} w(s) \leq \sum_{s \in S'} w(s)$ for any subset S' of S

In definition 2, the condition (1) is referred to the covering condition and the condition (2) is referred to weight minimizing condition. The minimum value $\sum_{s \in A} w(s)$ is called the solution value, denoted by $w(P)$, because the value is a unique value defined by MWCP $< C, S, w >$. By the above definitions, we can the following proposition directly.

Proposition 1
A solution A of MWCP $< C, S, w >$ does not include the empty subset in S, that is, if an element s of S is empty, s is not included in any solution A.

We leave the proof of this for a later forum.

The Algorithm

We now reach the point of this paper. However, before we describe the algorithm, we recap the rationale for the algorithm. The reader will recall that in order to integrate a transportation system at the computer systems level, one needs to read, analyse and act upon many data input streams (data streams). Further, the reader will recall that it is not possible to analyse all the data entering from the data streams, as it would place undue load on the analysis computer, and thereby slowing the computer to a point where calculations would not be performed in real time. Finally, the reader will recall that by defining a particular class of data to monitor, it is able to reduce the amount of actively reviewed material, further investigate sections of data that breach certain system defined conditions, and finally act upon them.

What follows is the algorithm based on the above description.

Algorithm 1:
(Step 0) For integer $i = 1$ to $2^{|S|}$, repeat Step1 through Step 6.
(Step 1) Transform i to the binary expression, say $b_{i1} b_{i2} ... b_{im}$.

(Step 2) Make subset $S_i = \{s_j \mid b_{ij} = 1, j = 1,...m\}$

(Step 3) Check the covering condition of C, that is, $\bigcup_{s \in S_j} s = C$. If S_i is not a covering of C, skip the following Steps and go to Step 5, otherwise go to Step 4.

(Step 4) Calculate the weight of S_i, $w(S_i) = \sum_{s \in S_i} w(s)$ and set the value to w_i

(Step 5) If $i < 2^{|S|}$ then increment i and go to Step 1, otherwise go to Setp 6.

(Step 6) Find all S_i's having the minimum value.

It is possible to validate algorithm 1 by using the definitions stated above. In a practical setting, the number of sensors in the transport network could number in the thousands. In fact, within the Adelaide, South Australia metropolitan area, there are about 600 signalised intersections. If we assume there is an average of 10 detectors per intersections that leaves us with approximately 6000 detectors embedded in the road surface itself. In the context of the middle layer of the 3LOM(Ikeda *et al.*, 2004), and assuming that a traffic management system based on IMAGINATION(Vogiatzis *et al.*, 2003) is devised that also includes natural environment detectors and other 'non' traffic volume detectors are placed in the network(Vogiatzis *et al.*, 2004), one could easily see that the number of sensors/detectors of all types in a city the size of Adelaide could reach the tens of thousands. However, even if someone were to use high performance computing platforms, it is still not possible to find the solution to algorithm 1 directly.

Reduction theorems for the MWCP

In almost all practical cases, it is possible to construct a set of conditions classified into specific groups. For example, all detectors currently located on approaches in the road collect the same type of information (simply they are located in a different geographical location). We will leave the proofs of these theorms for a later forum.

Theorem 1 (Reduction by partition of the Condition Set)
For the MWCP $P =< C, S, w >$, if there is a set of subsets of C, $C_1, C_2, ..., C_k$ satisfying the following conditions;

(1) $\bigcup_{i=1}^{k} C_i = C$,

(2) $C_i \bigcap C_j = \phi$ for any i, j with $i \neq j$,

(3) For any s in S, there exists unique integer j in $\{1,2,...k\}$ such that $s \subset C_j$,

then

$$w(P) = \sum_{i=1}^{k} w(P_i),$$

where $S_i = \{s \in S \mid s \subset C_i\}$ and $P_i =< C_i, S_i, w|_{S_i} >$.

Theorem 2(Reduction by unique sensor)
Let $P =< C, S, w >$ be a MWCP. For an element c of C, if $|\{s_i \in S \mid c \in s_i\}| = 1$, then

$w(P) = w(P') + w(s_0)$, where s_0 is the element of $\{s_i \in S \mid c \in s_i\}$, $S' = \{s - s_0 \mid s \in S\}$ and $P' =< C - s, S', w|_{S'} >$.

Theorem 3 (Reduction by Smaller-Weight Superset)
Let $P =< C, S, w >$ be a MWCP. If there is a pair of elements s_1 and s_2 in S such that $s_1 \subset s_2$ and $w(s_1) \geq w(s_2)$, then $w(P) = w(P')$, where $P' =< C, S - \{s_1\}, w|_{S-\{s_i\}} >$.

Theorem 4 (Reduction by branch)
Let $P =< C, S, w >$ be MWCP and s_0 be an element of S. One of two equations:
$w(P) = w(P_1) + w(s_0)$ or $w(P) = w(P_2)$ is satisfied, where:

$P_1 =< C - s_0, S', w|_{S'} >$, where $S' = \{s - s_0 \mid s \in S\}$,
$P_2 =< C, S - \{s_0\}, w|_{S-\{s_0\}} >$.

Improved Algorithm for MWCP

By using the stated theorems above, we can improve algorithm 1:

Algorithm 2(Improved Algorithm for MWCP)
(Step 0) Set $P =< C, S, w >$
(Step 1) Check the conditions of Theorem 1. If $P = P_1 \oplus P_2 \oplus \cdots \oplus P_k$ then this algorithm is applied to each smaller problem P_i recursively, otherwise go to Step 2.
(Step 2) Check the conditions of Theorem 2. If there is an element c of C with $|\{s_i \in S \mid c \in s_i\}| = 1$,then this algorithm is applied to each smaller problem $P' =< C - s, S', w|_{S'} >$, where $S' = \{s - s_0 \mid s \in S\}$ recursively, otherwise go to Step 3.
(Step 3) Check the conditions of Theorem 3. If there is a pair of elements s_1 and s_2 in S such that $s_1 \subset s_2$ and $w(s_1) \geq w(s_2)$, then this algorithm is applied to each smaller problem $P' =< C, S - \{s_1\}, w|_{S-\{s_i\}} >$, recursively, otherwise go to Step 4.
(Step 4) If P satisfies none of Theorem1, 2 and 3, select an element s_0 in S and construct the following two problems:

1) $P' =< C - s_0, S', w|_{S'} >$, where $S' = \{s - s_0 \mid s \in S\}$
2) $P' =< C, S - \{s_0\}, w|_{S-\{s_0\}} >$

and apply this algorithm to both P'.

How it all fits

In the case of IMAGINATION, we are interested in developing a new type of integrated traffic management and traffic micro-simulation tool. Among the many issues that one faces is the problem of data management, storage and manipulation. We accept that these systems (in this case we also include the current crop of traffic management/micro-simulation systems) can generate a great deal of data; however which data is important, which is noise, what can/can not be ignored etc. In the case of a DSMS, we are collecting and analysing

data 'on-the-fly', so the algorithm itself can monitor streams of data without there being undue load on the systems themselves.

In order to integrate various transportation systems in a real-time context, we need to monitor a number of inbound data streams; e.g. road temperature, atmospheric emissions/pollutants, wind velocity and direction, and noise level, to name a few. Sensing for environmental changes along with traffic volume changes provides us with an insight into the condition of traffic within a given section of a road network. However, this sort of data needs to be analysed twice; once at the time of detection to ensure that the traffic management system can manage the movement of traffic in real time, and second after the fact as a part of periodic reporting. Naturally, if the data was only reported on after the fact, it would be far too late from a tactical traffic movement prospective to do anything to alleviate traffic congestion, accidents, or other specific events that we are testing for.

The problem faced by such a system wanting to react in real-time is that the volume of data being collected by the streaming system would be so large, that the time required to analyse all the in the stream would be much more than is available to be able to react in real-time. Hence it is important to identify an algorithm that can filter out the bulk of the typical data in the system and only begin monitoring/analysing data that shows some 'difference' to the rest of the data. For example, if the audio stream detects an audio signature that somehow matches a traffic accident, then the system would begin the process of deep analysis from that time on at the origin location of the noise.

Conclusion

In this paper, we discussed the integration of transportation systems, and specifically traffic management systems. We looked at some of the problems associated with the collection of data in an environment where there is a potential of millions of second by second data points could be generated and some how managed in order to improve the management of transport networks into the long term. Specifically, we concentrated on how we manage the data points so as to detect particular network conditions when they occur. We formalised this by defining a group of language 'tags' to describe control rules within the transport network. Then we formalised the detection problem by transforming it from the application domain into a mathematical domain. In conjunction with the formalisations, we showed that detecting events of interest within a transport network is a special class of integer linear programming problem.

Finally, we proposed an algorithm for finding the optimised solution to detecting events of interest, and we explained that the algorithm is not only unique, but also efficient for practical application; more so than existing general integer linear programming algorithms.

References

Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J. (2002) *Models and Issues in Data Stream Systems*, In *21st ACM Symposium on Principles of Database Systems (PODS 2002)*, http://dbpubs.stanford.edu:8090/pub/2002-19

Babu, S., Subramanian, L. and Widom, J. (2001) *A Data Stream Management System for Network Traffic Management*, In *Workshop on Network-Related Data Management (NRDM 2001)*, http://dbpubs.stanford.edu/pub/2001-20

Brandeis University, Brown University and Massachusetts Institute of Technology (2003) *The Aurora Project*, 'June 2004', http://www.cs.brown.edu/research/aurora/

Gomory, R. E. (1963) *An algorithm for integer solutions of linear programs* In *Recent Advances in Mathematical Programming*(Eds, Graves, R. L. and Wolfe, P.) McGraw-Hill, pp. 269-302.

Ikeda, H. (2003) *Personal Communication - On a Three Layered Model for Transportation*, Vogiatzis, N., email

Ikeda, H. and Vogiatzis, N. (2003) *Personal Communications - Modelling Transportation Systems using Database Objects*, Vogiatzis, N.,

Ikeda, H., Vogiatzis, N., Wibisono, W., Mojarrabi, B. and Woolley, J. E. (2004) *Three Layer Object Model for Integrated Transportation System*, In *1st International Workshop on Object Systems and Systems Architecture*,

Schreier, U., Pirahesh, H., Agrawal, R. and Mohan, C. (1991) *Alert: An architecture for transforming a passive DBMS into an active DBMS*, In *International Conference on Very Large Data Bases*,

The Free Dictionary.com (2004) *Context-free grammar: encyclopaedia article about context-free grammar*, 'June 2003', http://encyclopedia.thefreedictionary.com/Context-free+grammar

Vogiatzis, N., Ikeda, H. and Wibisono, W. (2004) *On the Locality-Scope Model for Improving the Performance of Transportation Management Systems - ABSTRACT accepted*, In *27th Australasian Transportation Research Forum*,

Vogiatzis, N., Ikeda, H., Woolley, J. and He, Y. (2003) *The Journal of the Eastern Asia Society for Transportation Studies,* **5,** 2092-2107.