EFFICIENT TRAFFIC CONGESTION DETECTION METHOD IN VANET

Dhanya P.M[1], S.Ananth[2]
[1,2]Department of Computing Science
Sri Shakthi Institute of Engineering and Technology
Coimbatore, India

Abstract: ***Increasing road traffic density has led to the fact that, currently, motorway traffic congestions are one of the most common phenomena that motorists have to face in their trips. The distributed traffic information systems have come up as one of the most important approaches for detecting traffic flow problems on a road with the help of periodical messages that are transmitted across a vehicular ad hoc network (VANET). This paper presents an event-driven architecture (EDA) as a novel mechanism to get insight into VANET messages to detect different levels of traffic jams; it also takes into account environmental data that come from external data sources, such as weather conditions. The concept model that lies under EDAs is complex event processing (CEP).The effectiveness of the proposed congestion detection mechanism is evaluated through the simulation using NS2.***

Keywords: ***vehicular ad hoc network (VANET), Complex event processing (CEP), event-driven architecture (EDA).***

I. INTRODUCTION

Automobile traffic is a major problem in modern societies. Millions of hours and gallons of fuel are wasted everyday by vehicles stuck in traffic.

Traffic management, or intelligent transportation system, include several systems that provide traffic safety, guidance and optimization Distributed TISs try to avoid the problems of roadside equipment (RSE) approaches. A RSE-based TIS usually depends on infrastructure sensors that are installed in some segments of a road; consequently, its installation could be expensive, and it is limited to certain parts of the road.

By adding short-range communication capabilities to vehicles, the devices form a mobile ad-hoc network, allowing cars and road-sided wireless equipments to exchange information about road conditions. This is often referred to in the literature as Vehicular Adhoc Networks (VANET) [1].The users of a VANET, drivers or passengers, can be provided with useful information and with a wide range of interesting services. One category of such services includes safety applications, like various types of warnings: ice on road, intersection violation, cars in front braking, and collision avoidance and mitigation in situations like: lane changing, lane merging and preparation for imminent collision. Another important class of applications that can be deployed over VANETs is concerned with traffic operations and maintenance: dynamic route planning, weather conditions publishing and adaptive signal control in intersections [2].

To deal with huge quantity of events, over the last years, event-driven architectures (EDAs) have arisen as a new architectural paradigm. Event-driven architecture (EDA) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events. The concept model that lies under EDAs is complex event processing (CEP). We can see a CEP system as a sophisticated form of EDA that can deal with a large number of heterogeneous events from different streams and perform event aggregation and correlation in a very short time. CEP could be used to get insight into these streams and detect useful information. Thus, VANETs are a clear example of that kind of environments.

II. RELATED WORK

In recent years, several researchers have addressed the issue of distributed detection and propagation of traffic congestion information. A system that uses vehicle based GPS systems to discover and disseminate traffic congestion information; the system is called COC for VANET. This system maintains and disseminates three types of information: Raw Information (level 1), density information (level 2) and congestion areas information (level 3). Higher levels contain aggregated information.

University of Maryland proposed a novel system for congestion detection in VANET: StreetSmart that uses clustering as a data aggregation technique to combine related recordings of unusually slow speed. StreetSmart uses clustering algorithms that work over a distributed network where each node analyses the collected statistics eliminating the need for a central entity [6]. Clustering is the process of combining data points that are similar to each other by some measure.

Recently, many algorithms and methods have been proposed to detect congestions in distributed TISs [1]. For example, the approach in applies pattern recognition techniques. Although that work does not make use of any infrastructure element and it can deal with low penetration rates, it does not distinguish among lanes and requires a learning phase before being used. On the other hand, proposes a fuzzy-logic based mechanism to distinguish among different levels of traffic congestions; however, the system relies on ad hoc messages, in addition to beacon messages.

On the other hand, several CEP solutions have been developed in diverse fields, where it is necessary to cope with a high volume of information, which is the case for financial transactions, business decisions, or radio-frequency identification-based services. In this line, the use of CEP-based EDAs in the TIS scope has also arisen in recent researches, e.g., [5]. In

that work, the authors propose EDAs that are fed with data gathered from infrastructure sensors along a road. Later, such EDAs make up high-level events that are useful for detecting traffic problems on the road.

III. COMPLEX EVENT PROCESSING

The main goal of a CEP system is to detect real world situations, called activities like traffic congestion along a motorway. Fig. 1, left, the CEP is based on the idea that an activity can be split in simpler ones. In our case, traffic congestion can be split, for example, in several groups of slow vehicles. In turn, each of these activities can also be split in sub activities with lower level of abstraction [3]. Finally, some of them are reflected as clouds of interrelated rough events in the lowest layer of an IS. In the current scope, the target IS is the own VANET. Fig. 1, right, The CEP system takes as input the rough events and makes up a layered hierarchy of events with different levels of abstraction to compose one or more complex events that represent the initial real-world activity. Then, these complex events could be sent to a back-end system so that it performs some kind of action or procedure.

To do so, the CEP tries to find relationships, described as predefined patterns, among the events of the IS. The patterns are specified in the CEP system as event-processing rules (EPRs) which comprises both the pattern definition and the action to be triggered whenever the pattern is met. Based on the EPRs, event processing agents (EPAs) are designed. An EPA is composed of the EPRs that generate events of a certain level of abstraction. It also contains an event processing engine (EPE). This engine is in charge of running the different EPRs of the EPA and performing their associated actions.

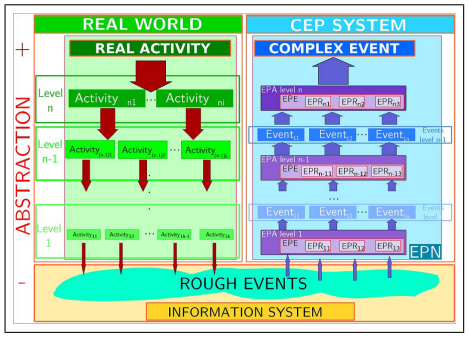

Figure 1: CEP concept model

IV. EDA FOR TRAFFIC CONGESTION

EDA acts as a middleware between the network level, which is in charge of the VANET communications at the low level, and the higher level, which holds the back-end applications. Fig. 2 summarizes the general structure of the EDA, which takes beacon messages from the network layer as rough events, and the EPAs perform a CEP processing of them afterward; moreover, the EDA takes as input events from data sources that state the road environment [5].

These data sources basically inform about the weather conditions on the EgoV road. Such events are then merged with the events made up from the beacon messages; consequently, the EPAs work in a cooperative way, and a hierarchy is composed. As a result, the EDA generates a traffic alarm whenever a traffic jam is detected. Such type of event is sent to a back-end application so that they could be used, for example, either to alert both the driver and the passengers of the EgoV or to display a warning message on the information panels of the motorway depending on where the EDA is running.

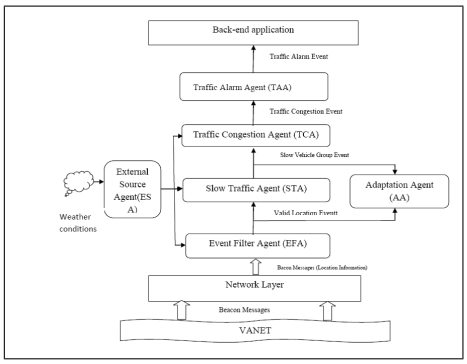

Figure 2: EDA components schema

A. Operation Mode

Each vehicle with VANET capabilities can indicate the lane where it is currently driving in each of its beacons. As far as that field is concerned, the EDA might deal with two different situations. In the former case, vehicles in the VANET can accurately indicate the driving lane in their beacons. In the latter case, vehicles cannot accurately estimate their driving lane because of several circumstances. It will detect traffic congestions without taking into account the lane information from the received beacons.

B. EPA Goals

Each EPA depicted in Fig. 2 performs a particular task and focuses on dealing with events with a certain level of abstraction. In particular, the goals of each of these EPAs are listed as follows :

a) The event filter agent (EFA) reads the Location events (beacons messages) from the network layer, and it is in charge of discarding events that are useless for the remainder of the EPAs. As a result, a new stream of events ValidLocationevents is created.

b) The slow traffic agent (STA) is fed with the stream of Valid Location events and monitors the traffic conditions along the motorway. Periodically, it creates a Slow VehicleGroupevent that contains the number of vehicles that are driving at very low speed.

c) The traffic congestion agent (TCA) continuously reads the stream of Slow Vehicle Group events. When the TCA detects a high density of slow traffic, it generates a TrafficCongestion event. This event represents the activity that traffic congestion has arisen in the motorway.

d) The traffic alarm agent (TAA) takes as input the stream of TrafficCongestionevents and categorizes them into three levels of congestion through a fuzzy classifier.

e) The adaptation agent (AA) is in charge of deciding which operation mode, i.e., lane or raw, is the most suitable at every moment. The AA also adjusts some inner parameters of the EFA through a heuristic that tries to maximize the number of detected slow vehicles.

V. IMPLEMENTATION OF THE PROPOSED EVENT-DRIVEN ARCHITECTURE

The present EDA has been developed by using the open source CEP solution Esper. It provides its own Structured Query Language (SQL)-like EPL, and it has a lightweight EPE embedded in Java. Each EPR of the EDA has been implemented by an EPL statement through which a stream of events is run and, as output, generates new events that feed other event streams.

A. Operation Mode Swapping

Each EPA comprises two different EPR sets, one for each operation mode; therefore, whenever the EDA changes from one mode to another, each EPA disables its EPR set for the outgoing mode and activates its set that corresponds to the incoming mode.

B. EFA

a) Lane Mode: The EFA is composed of the validation EPR. This rule is in charge of defining the sliding-time. This way, the validation EPR discards the messages that are out of date in a certain number of seconds. It also discards the messages that do not come from a vehicle in the same road and direction in which the EDA is interested. The EPL statement that implements that rule is given as follows:

Based on the fact that LEACH does not take into account the residual energy of the nodes during the selection of cluster heads in the set-up phase, we tend to develop the present energy and also the times being selected as CH or VCH. We first consider about the threshold T (n) and are modified to the following equation:

insert into ValidLocationEvent
select $*$ **from** BeaconMessage bm
where bm.ts - current_ts $<$ *sliWinSize* **and**
bm.road = *HomeRoad* **and**
bm.direction = *HomeDirection*

b) Raw Mode: In this mode, the EPR set of the EFA also comprises the aforementioned validation EPR; therefore, the EPR set for the EFA remains the same for both modes, because the key functionality of this EPA does not take into account any lane information.

C. STA

a) Lane Mode: When the EDA runs in this mode, the STA comprises the following three different EPRs: 1) the average-speed EPR; 2) the slow-vehicle EPR; and 3) the slowvehicle lane-group EPR The first EPR reads the stream of Valid Location events and creates an Average Speed event that is compound of the same fields that a Valid Location event comprises, along with the following two new fields: 1) avgSpeed, 2) segId, EPL statement is given as follows:

insert into AverageSpeedEvent
select vl.id as id,
vl.ts **as** ts,
vl.road **as** road,
vl.direction **as** direction,
mapFunc.getSegId(vl.position) **as** segId,
vl.lane **as** lane,
vl.laneConfidence **as** laneConfidence,
vl.position **as** position,
avg(vl.speed) **as** avgSpeed,
vl.speed **as** currentSpeed,
from ValidLocationEvent

This way, the STA, through the slow-vehicle EPR, can detect the vehicles that should be considered slow from the Average Speed event stream. Taking into account that the weather can have an effect on that consideration, this EPR makes use of the weather conditions of the road to accomplish its goal. The statement code for that rule is given as follows:

insert into SlowVehicleEvent
select asp.id **as** id
asp.ts **as** ts,
asp.road **as** road,
asp.direction **as** direction,
asp.segId **as** segId,
asp.lane **as** lane,
asp.laneConfidence **as** laneConfidence
asp.avgSpeed **as** avgSpeed,
mapFunc.getDist(asp.position, asp.segId) **as** dist,
from AverageSpeedEvent asp,
WeatherEvent.std:lastevent() we
Where extFunc.getSlowness(asp.avgSpeed, we.info) = slowness_threshold.

Then, the STA uses the slow-vehicle lane-group EPR to gather Slow Vehicle events during a certain amount of time. This EPR groups events according to their segment and lane afterward and counts the number of events in each group. This way, it can detect the number of slow vehicles in each of the lanes of a segment. This approach is shown in the following statement:

insert into SlowVehicleGroupEvent
select current_ts **as** ts,
se.road **as** road,
se.direction **as** direction,
se.segId **as** segId,
se.lane **as** lane,
avg(se.laneConfidence) **as** laneConf,
avg(avgSpeed) **as** avgSpeed,
count(distinct se.id) /
(max(se.dist)—min(se.dist)) **as** density,
max(se.dist) **as** maxDist,
min(se.dist) **as** minDist,
max(se.dist)—min(se.dist) **as** length
from SlowVehicleEvent.win:time(*slowTraffWin*) se
group by sv.segmentId, sv.lane

b) Raw Mode: For this mode, the STA is also composed of the following three EPRs: 1) the average-speed EPR; 2) the slow-vehicle EPR; and 3) the slow-vehicle raw-group EPR.

D. TCA

a) Lane Mode: The most important EPR for the TCA in this mode is the traffic congestion EPR. Such an EPR is in charge of detecting traffic congestion in a motorway. The EPL statement that implements such functionality is described as follows:

insert into TrafficCongestionEvent
select ∗ **from** SlowVehicleGroupEvent svg,
WeatherEvent.std:lastevent() we
where
extFunc.getDense(svg.density, we.info)=*dense_threshold*.

The stream of Traffic Congestion events is read by the following two different EPRs: 1) the lane traffic congestion merger and 2) the multilane afficcongestion.

b) Raw Mode: The TCA in this mode is compound of the following two EPRs: 1) the traffic congestion EPR and 2) the raw traffic congestion merger EPR. The rawtraffic congestionmerger EPR reads pairs of Traffic Congestion events, and whenever it finds two events whose heads and pairs are quite close to each other, it merges both events. As a result, a new Traffic Congestion event that includes the information of both events is generated.

E. TAA

a) Lane Mode: The different rules of this EPA classify the traffic congestions represented by the Traffic Congestion event and the Multilane Traffic Congestion events in different levels; therefore, a set of different alarms is generated, depending on the level of congestion detected. To classify the traffic congestions, the following two EPRs have been implemented: 1) the lane alarm EPR and 2) the multilane alarm EPR which provides a mapping between these values and the following three congestion levels: 1) slight; 2) moderate; and 3) severe.

b) Raw Mode: For this mode, the TAA is compound only of the raw alarm EPR. Similar to the lane alarm EPR, the raw alarm EPR reads the flow of Traffic Congestion events and makes up Traffic Alarm events through the same EFM function.

F. ESA

It is responsible for handling data from data sources that are not the VANET but provide useful information to detect traffic flow problems.

Functionality: The ES registers itself in the RM when the system starts. Next, the LDM asks the RM about the ESs that provides the weather information, and the RM returns a reference. Then, the LDM periodically sends a message to the ESW to get the weather conditions given a location. ESW receives the answer from its data provider, it makes up a Weather Event and forwards it to the LDM finally, the LDM processes it according to its rules and feeds the streams for the rest of EPAs with the new event.

G. AA

The AA has the following two goals: 1) it is in charge of deciding which operation mode is the most suitable at each moment and 2) it should resize some sliding-time windows of the EFA to maximize the number of detected slow vehicles. For this purpose, the second EPR of the AA continuously reads the low of Slow Vehicle events and calculates the number of such events that have been generated during a certain amount of time.

H. EFM

This module is in charge of providing some functionality to the EPAs through a set of methods. These functionalities are listed as follows:

a) Calculating the slowness value for each Average Speed event.
b) Inferring whether a Slow Vehicle Group event is dense enough to be considered a traffic congestion.
c) Classifying a Traffic Congestion event into three levels of congestion to make up a Traffic Alarm event.

VI. SIMULATION RESULT

In order to do research in traffic congestion we need to have a reliable simulation environment and ns2 is used for that purpose. First, the network simulator feeds the information about the network which consist of 20 vehicles. Afterward, the antenna positions are updated in the network simulator based on the information from the mobility model. This feedback loop is endlessly repeated during all the simulation.

a) SIMULATION PARAMETERS:
With regard to the evaluation of the EDA, the following two different measurements have been used: 1) the

detection rate (DR) and 2) the mean time to detection (MTTD). Both measurements are defined by the following equations:

$$DR = \frac{Number\ of\ detected\ traffic\ jams}{Total\ number\ of\ traffic\ jams}$$

$$MTTD = MTTD = 1/n\ \Sigma(tdetection - tstart)$$

Where tdetection is the instant at which the first traffic alarm is generated, and tstart is the start time of the traffic jam.

b) RESULTS:
Fig. 3 shows the DR of the EDA with different penetration rates and packet-loss ratios for traffic congestions. Simulations confirmed that the DR of the EDA was affected by its penetration rate for both operation modes. The lower the penetration rate is, the lower the DR that is achieved .It shows that raw density is not enough to detect the traffic jam.
MTTD: Fig. 4 depicts the MTTD of the EDA and the time gap between the moment at which the traffic jam spreads along the road as many meters as the x-axis indicates and the time at which the EDA generates the traffic alarm that informs about that length. It also shows that the raw mode achieves slightly better MTTDs than the lane mode, given a penetration rate of 100 and a 0 loss ratio.

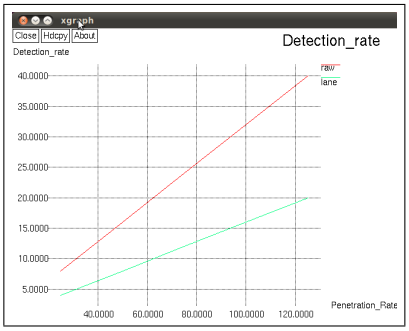

Figure 3: Detection rate of the EDA

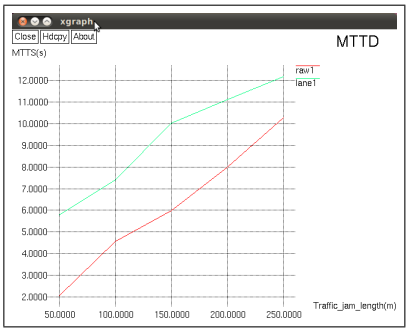

Figure 4: MTTD of the lane and raw modes

VII. CONCLUSION

In this paper, a CEP-based EDA has been put forward as a mechanism for detecting traffic jams in the context of distributed TIS. The system performs a CEP of the beacon messages, and it can detect different levels of congestion on a road; moreover, it can add environmental information such as the current weather condition so that the traffic jam detection is enriched. Results from the different simulations state that the EDA can detect several types of traffic jams. Thus, the EDA achieved better results, as long as the traffic jam covered several lanes.

References

[1] Pallavi Asrodia and Hemlata Patel. Analysis of various packet sniffing tools for network monitoring and analysis. In *International Journal of Electrical, Electronics and Computer Engineering*, volume 1, pages 55–58, 2012.

[2] Victor A Clincy and Nael Abu-Halaweh. A taxonomy of free network sniffers for teaching and research.

[3] Anshul gupta. A research study on packet sniffing tool tcpdump. *International Journal of Communication and Computer Technologies*, 01(06), 2010.

[4] Muna M. Taher Jawhar and Monica Mehrotra. System design for packet sniffer using ndis hooking. *International Journal of Computer Science and Communication*, 1(1):171–173, 2010.

[5] Rajeev S.G. S. Ansari and Chandrasekhar H.S. Packet sniffing: A brief introduction. *IEEE Potentials, Dec 2002- Jan. 2003*, 21(5):17–19, 2002.

