IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 31, 2017, accepted February 13, 2017, date of publication February 15, 2017, date of current version March 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2670024

A Cloud-Based Trust Management Framework for Vehicular Social Networks

XIAO CHEN, (Member, IEEE), AND LIANGMIN WANG
School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China
Corresponding author: L. Wang (wanglm@ujs.edu.cn)

The work was supported by the Natural Science Foundation of Jiangsu under Grant BK20160543 and Grant U1405255.

ABSTRACT The mobile industry's evolution from 4G to 5G will lead to a deep progress on mobile applications that are widely used in some new environments, such as vehicular social networks (VSNs). In VSNs, which are considered the first automobile social networks, vehicular communication can facilitate large-scale data sharing between drivers and their neighbours. However, malicious users of VSNs can also disseminate false information over the network. Traditional public key infrastructure (PKI) cannot recognize these malicious users, as they all have authorized identities. Thus, a trust management mechanism is introduced to secure vehicular social data. This paper demonstrates a high-level trust management model and its deployment scheme based on a vehicular cloud system. We propose a layered trust management mechanism that benefits from efficient use of physical resources (e.g., computing, storage, communication cost) and explore its deployment in a VSN scenario based on a three-layer cloud computing architecture. Moreover, performance modeling of the proposed trust management scheme is conducted through a novel formal compositional approach – Performance Evaluation Process Algebra (PEPA). PEPA has superior features in compositionality and parsimony, which means that it can efficiently model systems with layered architectures and complex behaviours. PEPA also supports various numerical analyses through calculating its underlying continuous time Markov chains (CTMCs) directly or solving a set of approximated ordinary differential equations (ODEs). According to analysis outcomes, we analyzed several key performance properties of the scheme and related capacity issues in deployment. The findings also reveal an efficient investigation approach for evaluating the performances of trust models.

INDEX TERMS Trust management, vehicular social networks, cloud computing, performance evaluation, formal method, PEPA.

I. INTRODUCTION

The fifth-generation (5G) cellular network will provide native support for machine-to-machine (M2M) communication by satisfying fundamental requirements for addressing low-latency-rate services. Low latency and real-time operation demands that data transmission must be completed reliably within a given time interval. Vehicle-to-X communication is a typical example, which can improve the safety and experience of drivers through timely delivery of critical messages (e.g., warning and social messages) [1], [2]. The next-generation vehicles employ a new set of emerging wireless networks for the vehicular environment, i.e., 4G/5G, WiFi and Vehicular Ad hoc Networks (VANETs). Enhanced wireless communication dramatically pushes forward vehicular applications. For example, a traffic jam is a common occurrence on some roads during rush hours. Thus, roads become social areas for vehicles to communicate or exchange information. Human factors are involved in the network, not just for the purpose of safety but also entertainment. Human behaviors and preferences deeply impact networks, especially communications. Hence, vehicular communication can be considered as a social network of automobiles [3].

Opportunistic networking applications, such as introduction services, friend finders, job recommendations, content sharing, and gaming, have close relationships with social networking [3]. Human factors (human mobility, selfishness and user preferences) are also involved in VANET applications. This emerging networking paradigm is called ''Socially Aware Networking'' [4], which makes use of mobile-device users' social relationships to create mobile social networks. As human behaviors and their social characteristics deeply impact on communication networks, this generates the concept of ''Vehicular Social Networks (VSNs)'' formed by socialized vehicles.

VSNs can be directly perceived as a group of in-vehicle individuals with similar preferences, interests or common needs in an on-road environment during some specific periods. Communications of VSNs have characteristics similar to those of VANETs, such as Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) types of communications [5]. Additionally, they include some human factors, as mentioned before, that affect vehicular connectivity and security. Generally, there is a large amount of information, including traffic information (congestion, road conditions, etc.), personal information (locations, voices, videos, pictures, etc.) and vehicle information (warnings, sensor figures, etc.), which can be disseminated and shared between users through their vehicles [6]. As an instance of VSNs, a vehicle can broadcast the message of current available parking places at a destination to other vehicles it has met in order to facilitate their parking experiences. However, potential risk still exists due to the reasons of defective sensors, software viruses, or malicious intent of VSN users.

VSNs provide a platform for individuals to exchange various types of information, so it is important to distinguish between the trustworthy information and the untrustworthy. The trustworthy information can be considered as that shared factually with others, while the untrustworthy is false shared information [7]. For example, a driver may send spurious parking information about a desired destination to others through VSNs so that the driver can obtain a parking place when arriving there. Thus, untrustworthy information sent by selfish or malicious vehicles has a high probability to become harmful data in networks. Currently, traditional PKI forms the first line of defense against unauthorized users; however, it is difficult for PKI to distinguish untrustworthy users from all the authorized users. Therefore, an interesting question is posed: *How can the trustworthiness of information in VSNs be guaranteed?*

This research aims to explore the trust management issue by proposing a trust model architecture and to investigate the deployment of the trust model through a cloud-based vehicular communication system. Section 1 briefly introduces the background of VSNs and related potential issues. Section 2 covers some related works of recent years that make contributions to trust management. Section 3 describes the architecture of the investigated vehicular social network. Section 4 defines a trust management framework based on VSN architecture. Section 5 briefly introduces the adopted formal modeling language – PEPA. Section 6 models the proposed trust management framework using PEPA. Section 7 generates a set of performance analyses by solving underlying Markov chains of PEPA models directly. Section 8 applies an alternative fluid-flow analysis for evaluating models. Section 9 concludes this research and gives some key points for future work.

II. RELATED WORK AND CONTRIBUTIONS

VSNs have been considered in recent years with the development of vehicular communications. Vegni [3] wrote a survey of VSNs in which she summarized the recent literature on VSNs and introduced the main features of VSNs, from novel technologies to social aspects, used for mobile applications such as ''Verse,'' a vehicular social application developed by Luan [8], as well as related issues and challenges. This survey also provides an overview of the state-of-the-art on safety and entertainment applications that run on VSNs.

Yang's [9] research switches focus from the general understanding to a specific issue: trust in VSNs. Yang particularly explains the potential issues of trust management in the communications of VSNs and introduces the basic theory of trust management in a VSN scenario. As trust management has been widely used for on-line social networks, various typical trust models have been reviewed in our initial literature research. For instance, a flow-based trust evaluation scheme, ''GFTrust,'' is proposed by Jiang in Ref [10], in which they address path dependence using network flow to solve the issue referring to path dependence. They also handle another challenge, trust decay, by modeling it as the leakage associated with each node. Moreover, for Chord-based P2P networks, Meng [11] proposed a guarantee-based trust model called ''GeTrust'' that is proved to be effective and efficient in terms of improving the successful transaction rate, resisting complex attacks, reducing network overhead and lowering computational complexity.

In the area of VSNs, we also found some valuable literature. Researchers [12] (e.g., Hussain) studied a hybrid trust establishment and management framework for communications of VSNs. The framework includes two trust management solutions, i.e., email-based social trust and social network-based trust, to target different groups of mobile applications. From another aspect of vehicular trust, the trustworthiness of VSNs can be enhanced by achieving both data trust, which is an assessment of whether and to what extent the information disseminated in VSNs is trustworthy, and node trust, which refers to how trustworthy the nodes of VSNs are. In Li's research [13], ART, an attack-resistant trust management scheme, is proposed for vehicular networks to detect and handle malicious attacks and assess the trustworthiness of both data and mobile nodes of networks. However, as the data in VSNs has been grouped in different application domains, the ART does not fit the condition. The proposed trust model in VSNs must solve the evaluation of data or nodes from different domains.

Additionally, VSN is a joint network between social network and VANET. Thus, it needs a central cloud as its social service provider. Moreover, VSN also needs the roadside infrastructures to support its communications between central clouds and vehicles or between vehicles themselves. In order to deploy a trust management scheme, a cloud-based vehicular network system is required according to our literature reviews [14], [15]. In Ref [14], the authors presented a state-of-the-art survey of vehicular cloud computing (VCC), including extensive applications, cloud formations, key management, inter-cloud communication systems, and a broad scope of privacy and security issues. A general architecture

of VCC is also explored in the survey. Similarly, the article (Ref. [15]) also investigates cloud computing technologies used for vehicular networks. Various transportation services supported by VCC were discussed in the article, such as security and privacy, energy efficiency, resource management, and interoperability.

According to these literature reviews, we have completed some pre-research on trust management of VSNs and its deployment based on VCC architecture. Research in Ref. [16] proposed a general trust management framework targeting VSNs. Furthermore, a VCC-based three-layer communication architecture for VSNs has been proposed in the recent pre-research phase [17]. This work continues our preceding research to investigate a detailed trust management framework to secure vehicular data on VSNs and evaluate a proposed deployment plan for the trust framework. The main contributions of this research can be highlighted as follows:

First, a trust evaluation framework is proposed to solve the trustworthiness issue in a local VSN. Second, a deployment scheme is designed for the framework through building a layered vehicular cloud network architecture. Third, performance modeling scheme of the proposed framework and architecture is conducted through a novel approach that is a formal compositional method based on stochastic process algebra, named PEPA. Forth, performance analysis is completed through the numerical solutions of PEPA models, and it is worth mentioning that a fluid-flow approach is applied to approximate the numerical analysis to overcome its drawback in large-scale quantitative analysis.

III. CLOUD-BASED VEHICULAR SOCIAL NETWORK ARCHITECTURE

A three-layer vehicular cloud network (VCN) architecture is designed to support deployment of the proposed VSN trust management framework. The VCN architecture, shown in Fig. 1, consists of three main layers: the central cloud layer (CCL), road-side cloud layer (RCL) and vehicular cloud layer (VCL).

The CCL gathers a group of server clusters that have powerful computational abilities and massive storage capacities; furthermore, it provides services for road-side facilities and vehicles via V2R (vehicle-to-road-side units) communication. The CCL is designed to preserve some general information (e.g., a profile including vehicle's identity, history trust and friends) of authorized vehicles in VSNs.

The RCL is a set of road-side units, including communication facilities and local servers. This layer plays the role of manager for trust evaluation in a local area. It will create virtual environments for vehicles requesting trust evaluation services and then undertake the calculation of trust degrees as well as the communication with neighboring road-side units. As the RCL is a core part of such a three-layer VCN architecture, our performance modeling and analysis will be based on this layer, which will be detailed in the following sections.

The VCL is a kind of local cloud on the road, which is built on a group of cooperating vehicles and managed by the RCL. Vehicles share their resources (e.g., computation, storage and spectrum) in the cloud. Each vehicle is allowed to reserve cloud services according to its demand. All physical resources can be scheduled dynamically based on the demands of vehicles. The VCL makes physical resources of vehicles virtual; however, virtual resources are managed by a component called the "cloud controller" embedded in the RCL. A vehicle can choose a nearby road-side cloud to reserve a transient cloud service. Once the vehicle leaves the scope of the road-side cloud, its reservation will be transferred to the next road-side cloud through the RCL. This short-lived cloud service is named the "Transient Cloud." The VCL provides physical resources (e.g., computing and storage) to support the RCL to complete the trust evaluation cooperatively. This layer improves the vehicle-resource utilization and QoS of VSNs.

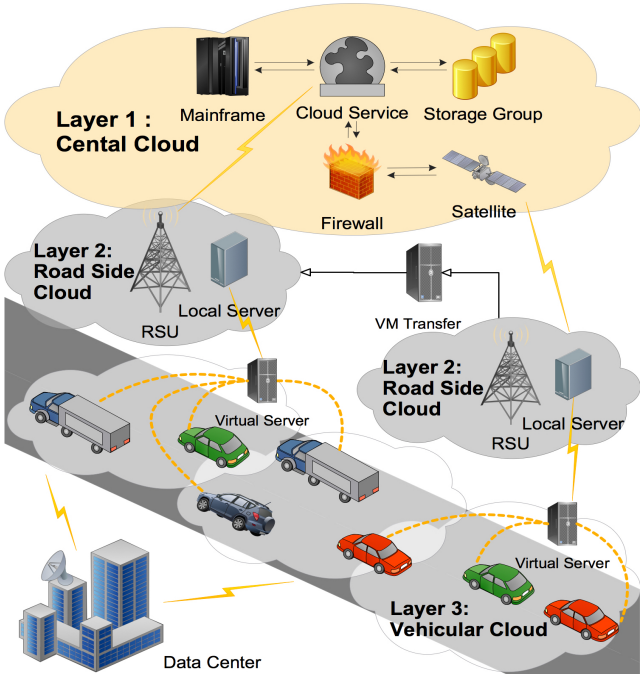


FIGURE 1. Three-layer vehicular cloud network architecture.

Overall, the three-layer VCN architecture supports the deployment of trust evaluation framework which is applied to solve the trustworthiness issue in VSNs. The following section will illustrate details of trust evaluation framework and its implementation based on the architecture.

IV. TRUST MANAGEMENT FRAMEWORK OF VSNs

A. GENERAL TRUST MODEL

According to the preceding VSN architecture, a trust management framework is proposed as shown in Fig. 2. To illustrate the framework, a set of concepts need to be defined as follows:

- **Application Domain:** A group of applications sorted in terms of their properties.

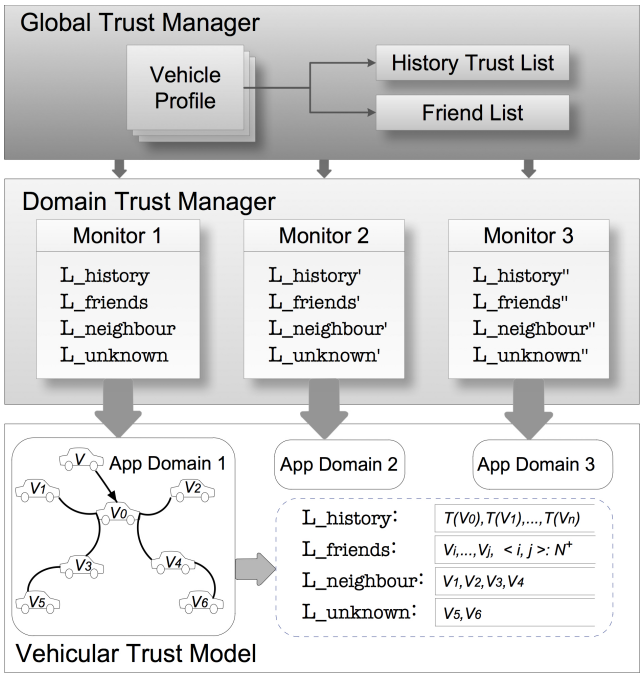


FIGURE 2. Trust management framework in VSNs.

- **Friend Entities:** Fully trusted vehicles of an authorized vehicle in VSNs.
- **Neighbor Entities:** Directly connected neighbors of a given vehicle in the local network.
- **Unknown Entities:** Vehicles that are communicating for the first time in the local network.
- **History Trust:** Trust value calculated (based on a trust evaluation scheme) in the last communication.
- **General Trust:** Overall trust evaluated in each communication including direct trust and indirect trust.
- **Vehicle Profile:** A record of vehicles' general information (e.g., identity, history trust and friend list).

Fig. 2 clearly depicts three levels of the framework: The global trust manager (GTM), associated with the CCL, records each vehicle's profile, including a history trust list and a friend list, and manages the sub-level trust managers. The domain trust manager (DTM), associated with the RCL, undertakes the work of trust evaluation through several monitors that control the calculation of trust degrees based on different application domains. For each vehicle in a monitor, there are four lists in which the information of the history trust value, trusted friends, direct neighbors and unknown indirect neighbors is recorded respectively. Most evaluation processes of the vehicular trust model are conducted in the bottom level of the framework which is associated with the VCL. In this level, the overall trust of a target vehicle will be evaluated in terms of the trust model criterion through virtual resources generated in the VCL.

According to the framework, a general trust model is defined by the formal expressions given in Algorithm 1. As defined in step 4 of the algorithm, the overall trust ($T(V_s \rightarrow V_r)$) is the weighted average of the neighbor

Algorithm 1 General Trust Model

Variables:
V_s: sending vehicle;
V_r: receiving vehicle;
L_{his}: history trust list;
L_{friend}: friend list;
L_{neig}: direct neighbor list;
L_{unkn}: unknown and indirect neighbor list;
$F(V_i)$: trust evaluation function for vehicle i;
T_{neig_dir}: trust evaluation from direct neighbors;
T_{neig_indir}: trust evaluation from indirect neighbors;
$F_{in}(V_s)$: internal friend (same app-domain) set of V_s;
$F_{ex}(V_s)$: external friend (other app-domains) set of V_s;
$T_{fri\_in}(V_s)$: trust evaluation from internal friend set;
$T_{fri\_ex}(V_s)$: trust evaluation from external friend set;
$T(V_s \rightarrow V_r)$: overall trust degree of V_r in the view of V_s.
Input: V_s; V_r.
Output: $T(V_s \rightarrow V_r)$.

1: **Step 1: Trust Calculation Based on Neighbor Nodes,** T_{neig}
2: **for** each $V_i \in L_{neig}(V_s)$ **do**
3: Calculate $T_{neig\_dir} = \sum_i n_i F(V_i)$,
4: n_i is the weight and $\sum_i n_i = 1$;
5: Sort out neighbors of V_i for $L_{neig}(V_i)$;
6: **for** each $V_j \in L_{neig}(V_i)$ **do**
7: Calculate $T_{neigh\_indir} = \sum_j m_j F(V_j)$,
8: m_j is the weight and $\sum_j m_j = 1$;
9: $T_{neig}(V_s \rightarrow V_r) = \alpha \cdot T_{neig\_dir} + \beta \cdot T_{neig\_indir}$,
10: and $\alpha + \beta = 1$.
11: **Step 2: Trust Calculation Based on Friend Nodes,** T_{fri}
12: Sort out friends of V_s to obtain $F_{in}(V_s)$ and $F_{ex}(V_s)$;
13: **for** each $V_k \in F_{in}(V_s)$ **do**
14: Calculate $T_{fri\_in}(V_s) = \sum_k n'_k \cdot F(V_k)$,
15: $\sum_k n'_k = 1$;
16: **for** each $V_l \in F_ex(V_l)$ **do**
17: Calculate $T_{fri\_ex}(V_s) = \sum_l m'_l F(V_l)$,
18: $\sum_k m'_l = 1$;
19: $T_{fri}(V_s \rightarrow V_r) = \alpha' \cdot T_{fri\_in} + \beta' \cdot T_{fri\_ex}$,
20: and $\alpha' + \beta' = 1$.
21: **Step 3: Trust Calculation Based on History,** T_{his}
22: Sort out L_{his} for obtaining $T_{his}(V_s \rightarrow V_r)$
23: **Step 4: Overall Trust Calculation,** $T(V_s \rightarrow V_r)$
24: $T(V_s \rightarrow V_r) = M \cdot T_{neig} + N \cdot T_{fri} + P \cdot T_{his}$,
25: and $M + N + P = 1$.

trust (T_{neig}), friend trust (T_{fri}) and history trust (T_{his}) for a given vehicle V_r that performs as a receiver. Step 1 performs a trust evaluation for each neighbor of the sender vehicle (V_s) through the calculation of trust values based on direct neighbors (T_{neig_dir}) and indirect neighbors (T_{neig_indir}) of V_s respectively. Step 2 obtains the trust value from V_s's friends including internal friends that exist in the same application domain and external friends that come from other application domains. Step 3 takes the historical trust value directly from

the list L_{his}, which represents the overall trust value of a vehicle in the last communication with V_s.

Once the trust model is defined, it should be implemented for trust evaluation in VSNs. A series of activities will be generated when the evaluation starts based on the proposed trust model. The next subsection will demonstrate such activity flow in the VCN.

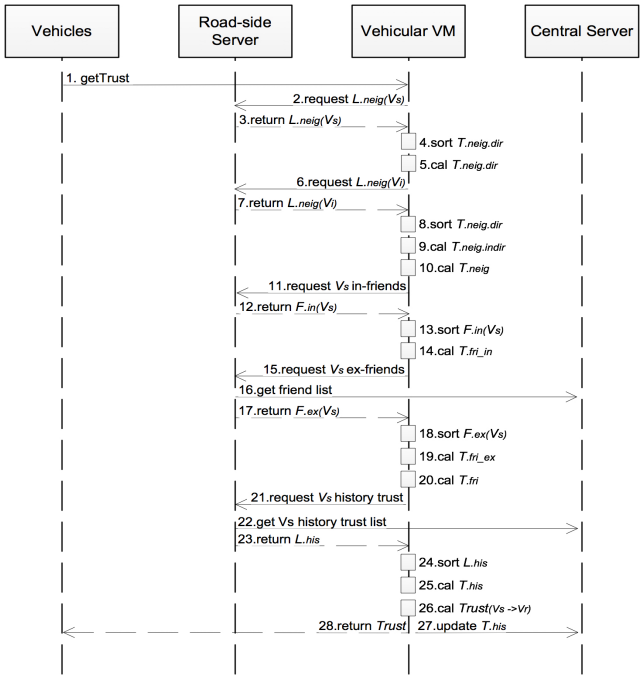


FIGURE 3. Trust model implementation.

B. IMPLEMENTATION OF TRUST MODEL UNDER A VSN

This subsection illustrates the implementation of the proposed trust model under a three-layer cloud-based VSN. Fig. 3 describes the process of completing trust evaluation in a VSN. In this figure, four key components are involved, which are central servers in the CCL, road-side servers in the RCL, a vehicular virtual machine (VVM) generated from vehicles' physical resources, and vehicle that are users. The trust calculation is mainly processed in the VVM, which cooperates with other components (e.g., road-sider servers and central servers). As introduced in preceding sections, central servers in the CCL store the profile of each vehicles, which is necessary information for trust calculation. Road-side servers in the RCL create a VVM for the trust calculation. As most temporary data in the calculating process are stored in road-side servers, the VVM will have many communications with road-side servers to obtain those figures as well as with central servers to obtain some other information (e.g., history trust and neighbor list). Fig. 3 depicts a trust evaluation process based on the general trust model.

According to the general trust model, a vehicle (namely, sending vehicle, V_s) starts the trust evaluation by sending a request, as displayed in step 1 of Fig. 3. The VVM, first, performs the step of calculating the trust value of neighbors (T_{neig}) by sorting the neighbor list of the receiving vehicle from road-side servers. Trust value calculation based on neighbors is conducted from step 2 to step 10 in Fig. 3, including for direct neighbors (steps 2-5) and indirect neighbors (steps 6-9). Second, the VVM conducts trust calculation of the receiving vehicle's friends (T_{fri}) by sorting the friend list from road-side servers. Similarly, both internal friends, in the same application domain, and external friends, in different application domains, are viewed for their trust evaluation in steps 11-14 and steps 15-19 respectively. Trust-based neighbors and friends are all calculated in the VVM in cooperation with road-side servers. Third, the history trust value (T_{his}) is obtained from central servers (steps 21-25); thus, this involves cooperation with central servers (step 22). Finally, the overall trust value, $T(V_s \rightarrow V_r)$, is calculated based on three types of trust values (T_{neig}, T_{fri} and T_{his}) in step 26. Thereafter, the obtained trust value will be returned to the sending vehicle and central severs for updating.

In order to evaluate the proposed trust evaluation framework within a cloud-based VSN, performance models will be generated to describe the trust evaluation process by defining the related activity flow and interactions in systems. The main modeling technique is a formal method called PEPA (Performance Evaluation Process Algebra). The next section will give details of PEPA including its syntax and novel features.

V. MODELLING APPROACH: PERFORMANCE EVALUATION PROCESS ALGEBRA

This section will briefly introduce the PEPA language and its numerical representation schema. The numerical representation schema for PEPA represents a model numerically rather than syntactically supporting the use of mathematical tools and methods to analyze the model.

A. SYNTAX OF PEPA

PEPA syntax consists of six important components, which are given as follows [18]:

- **Prefix**: $(\alpha, r).P$: Prefix is the basic mechanism that describes the behaviour of the system. Such a component will subsequently behave as P after it carries out the activity (α, r), which has action type α and a duration that satisfies the exponential distribution with parameter r.
- **Choice**: $P + Q$: The component $P + Q$ represents a competition between two components. The system may behave either as P or as Q. The activities of both P and Q are enabled. The choice is resolved by a race policy: the component whose activity is completed first proceeds, and the other is discarded.
- **Cooperation**: $P \underset{L}{\bowtie} Q$: The cooperation combinator describes the synchronization of P and Q over the activities in the cooperation set L. In fact, for any activity whose action type is contained in L, P and Q must cooperate to achieve the activity. However, they will proceed

independently and concurrently with any activity whose action type is not included in L.

- **Parallel**: $P||Q$: The component $P||Q$ represents two concurrent but completely independent components, meaning the cooperation set is empty. This is simply a shorthand notation for $P \underset{L=\emptyset}{\bowtie} Q$.
- **Hiding**: P/L: Hiding makes the activities whose action types are in L invisible to an external observer. The component P/L behaves as P except that any activities of types within the set L are hidden.
- **Constant**: $A \stackrel{\text{def}}{=} P$: Constants are components whose meaning is given by a defining equation such as $A \stackrel{\text{def}}{=} P$, which gives the constant A a behavior similar to the behavior of component P.

The syntax of PEPA is given as:

$$P ::= (a, \lambda).P \mid P + Q \mid P\langle L\rangle Q \mid P \mid Q \mid A$$

This PEPA statement involves all four combinators mentioned in the previous paragraph. The last part of this statement $P ::= A$ identifies component P with A. When the rate of action is passive, we use the symbol $\top$.

On the basis of the operational semantic rules, a PEPA model may be regarded as a labeled multi-transition system

$$\left(\mathcal{C}, \mathcal{A}ct, \left\{\stackrel{(\alpha,r)}{\longrightarrow} |(\alpha, r) \in \mathcal{A}ct\right\}\right)$$

where $\mathcal{C}$ is the set of components, $\mathcal{A}ct$ is the set of activities and the multi-relation $\stackrel{(\alpha,r)}{\longrightarrow}$ is given by the rules.

B. FEATURES OF PEPA

PEPA is selected as the main modelling technique due to its obvious advantages in modelling large scale systems [18], such as VSNs. It benefits from the following features: *Compositionality*, the compositional nature of PEPA provides the ability to model a system as the interaction of subsystems; *Formality*, PEPA language has a structured operational semantics and provides a formal interpretation for all expressions; *Abstraction*, PEPA is able to construct complex models from detailed system components, disregarding the details when it is appropriate to do so.

Currently, the popular modelling languages for Markov models are queuing networks and stochastic Petri nets (SPN). A queuing network is a directed graph in which each node represents a queue (also called service centre). Customers representing system jobs flow through the nodes and complete service. The arcs of the network represent the system topology and routing probabilities. The amount of customers currently occupying each service centre represents the current state of the system. Queuing network has the feature compositionality but it is informal. In contrast, SPN is a formal mathematical modelling language. SPN is a directed graph with two kinds of node, places and transitions. The system state is represented through the number of tokens at each place in the network. SPN is an alternative mean of generating stochastic models for performance modelling. However, SPN has some restrictions in compositionality compared with queuing network. It is complex to represent the layered structure of the system. Queuing network provides compositionality but not formality; stochastic Petri nets offer formality but not compositionality; and neither gives abstraction mechanism.

According to the syntax of PEPA, the trust model can be defined in the formal expression. Moreover, PEPA supports model simplification skills that are applied to reduce the cost of analysis. Next section demonstrates these PEPA models.

VI. PEPA MODELS OF TRUST FRAMEWORK

According to the trust model implementation, this section generates a corresponding PEPA model based on PEPA syntax and specifies the model simplification using its related theory (e.g., bisimulation and isomorphism), which reduces the cost of calculation in numerical analysis so as to avoid the state space explosion in solving Markov chains.

A. MODEL CREATION

In terms of the activity flow in Fig.3, we can define the VSN-based trust evaluation model with the PEPA language as follows:

***Model*1**:

$$\begin{aligned}
\boldsymbol{VN} &\stackrel{\text{def}}{=} (getTrust, r_{gt}).(returnTrust, \tau).VN;\\
\boldsymbol{VVM} &\stackrel{\text{def}}{=} (getTrust, \tau).VVM1;\\
VVM_1 &\stackrel{\text{def}}{=} (reqVsNb, r_{rqvsn}).(returnVsNb, \tau).\\
&\quad (sortVsNb, r_{svsn}).(calTnbd, r_{ctnd}).VVM2;\\
VVM2 &\stackrel{\text{def}}{=} (reqViNb, r_{rqvin}).(returnViNb, \tau).\\
&\quad (sortViNb, r_{rvin}).(calTnbid, r_{ctni}).VVM3;\\
VVM3 &\stackrel{\text{def}}{=} (calTnb, r_{ctn}).VVM4;\\
VVM4 &\stackrel{\text{def}}{=} (reqVsFi, r_{rqvsfi}).(returnVsFi, \tau).\\
&\quad (sortVsIf, r_{svsi}).(calTif, r_{cti}).VVM5;\\
VVM5 &\stackrel{\text{def}}{=} (reqVsFe, r_{rqvsfe}).(returnVsFe, \tau).\\
&\quad (sortVsEf, r_{svse}).(calTef, r_{cte}).VVM6;\\
VVM6 &\stackrel{\text{def}}{=} (calTf, r_{ctf}).VVM7;\\
VVM7 &\stackrel{\text{def}}{=} (reqVsHis, r_{rqvsh}).(returnVsHis, \tau).\\
&\quad (sortVsHis, r_{svsh}).(calThis, r_{cth}).VVM8;\\
VVM8 &\stackrel{\text{def}}{=} (calTrust, r_{ct}).VVM9;\\
VVM9 &\stackrel{\text{def}}{=} (updateTrust, r_{ut}).(returnTrust, r_{rt}).VVM;\\
\boldsymbol{RSS} &\stackrel{\text{def}}{=} (reqVsNb, infty).(returnVsNb, r_{rtvsn}).RSS1;\\
RSS1 &\stackrel{\text{def}}{=} (reqViNb, \tau).(returnViNb, r_{rtvin}).RSS2;\\
RSS2 &\stackrel{\text{def}}{=} (reqVsFi, \tau).(returnVsFi, r_{rtvsfi}).RSS3;\\
RSS3 &\stackrel{\text{def}}{=} (reqVsFe, \tau).(getVsFe, \tau).\\
&\quad (returnVsFe, r_{rtvsfe}).RSS4;\\
RSS4 &\stackrel{\text{def}}{=} (reqVsHis, \tau).(getVsHis, \tau).\\
&\quad (returnVsHis, r_{rtvsh}).RSS;
\end{aligned}$$

$$\boldsymbol{CS} \overset{\text{def}}{=} (getVsFe, r_{gvsfe}).(getVsHis, r_{gvsh}).CS;$$

$$\boldsymbol{Sys} \overset{\text{def}}{=} VN[m] \underset{L_1}{\bowtie} VVM[n_1] \underset{L_3}{\bowtie} RSS[n_2] \underset{L_2}{\bowtie} CS[n_3]$$

$$L_1 = \{getTrust,\ returnTrust\}$$

$$L_2 = \{getVsFe,\ getVsHis\}$$

$$L_3 = \left\{ \begin{array}{c} reqVsNb,\ reqViNb,\ reqVsFi,\ reqVsFe, \\ returnViNb,\ returnVsNb,\ returnVsFi, \\ returnVsFe,\ reqVsHis,\ returnVsHis \end{array} \right\}$$

The above PEPA model defines four components: *VN* (Vehicular Nodes), *VVM* (Vehicular Virtual Machine), *RSS* (Road Side Servers) and *CS* (Central Servers). The components all map to the four corresponding elements in Fig. 3 respectively. For each component, the model defines its action flow as depicted in Fig. 3. For instance, *VN*, the first component in the model, includes two activities: *getTrust*, representing Step 1 in Fig. 3, and *returnTrust*, representing Step 28, which is a cooperative activity between *VVM* and *VN*. It is worth mentioning that *returnTrust* is defined as a passive action with a passive rate τ. This means that the *returnTrust* action is completed through cooperation with *VVM* in which there is an identical action named *returnTrust* but with an active rate. According to PEPA theory, for the cooperative action, the component with an active rate dominates this cooperation. Thus, the shared action *returnTrust* in the above PEPA model is actually controlled by the *VVM* component. Similarly, in other parts of the model, cooperative actions are defined.

The VVM component is the core part in the model, as it defines most activities in the trust calculation process. In the model, sub-action flow from state VVM_1 to state VVM_2 represents the trust calculation for direct neighbors (T_{neig_dir}) corresponding to steps 2-5; similarly, sub-action flow from state VVM_2 to state VVM_3 represents the calculation for indirect neighbors (T_{neig_indir}). Finally, the total trust of all neighbors, T_{neig}, is calculated after state VVM_3 and completed at state VVM_4. In this process, *VVM* also cooperates with *RSS*, for which the rates of cooperative actions in *VVM* are passive, τ. Other calculations (e.g., T_{fri} and T_{his}) are modeled in the same away.

Finally, the *Sys* component defines the whole system, in which the cooperation between components is specified with the operation $\bowtie$. All shared actions are grouped into several sets corresponding to each cooperation between components. For each component defined in *Sys*, there is a related parameter denoting the number of instances of each component, such as n_1, n_2, n_3 and m.

B. MODEL SIMPLIFICATION

Model 1 simulates a complete trust evaluation process that is depicted in Fig. 3. However, it suffers from a commonly encountered state space explosion problem. For instance, when the numbers of instances of *VN*, *VVM*, *RSS* and *CS* are each increased from 1 to 2, the number of states in the state space grows from 28 to 2012; and this amount will increase to over one million when the instance number of each component is 4. Thus, with the increase of instance number, Model 1 will soon encounter a state space explosion. As a result, steady-state analysis by solving for the whole state space cannot be conducted. To solve this problem, model simplification techniques must be applied to reduce the scale of the state space.

In the first step, the notion of bisimulation, defined to capture the idea of equivalence as identical observed behaviors, is applied to simplify the model by changing its structure but retaining the same transitions; and the derivatives which resulting from the same transitions in the components are themselves bisimilar.

According to the structure of Model 1, it is clear that the action flow of the *RSS* component is almost the same as some parts of *VVM*, with the only difference being in the type of rates, active or passive. Similarly, the actions of the *CS* component involving *RSS* are cooperative actions with *RSS*. Since the observation mainly focuses on the *VVM* component under various conditions referring to the *VN* component, these duplicate actions in different components can be merged into a single component to reduce the scale of cooperation that dominates the scale of the state space. Thus, Model 1 can be simplified by merging components *RSS* and *CS* with *VVM*. The new model can be updated as the following Model 2.

Model 2:

$$\boldsymbol{VN} \overset{\text{def}}{=} (getTrust, r_{gt}).(returnTrust, \tau).VN;$$

$$\boldsymbol{VVM} \overset{\text{def}}{=} (getTrust, \tau).VVM1;$$

$$VVM_1 \overset{\text{def}}{=} (reqVsNb, r_{rqvsn}).(returnVsNb, r_{rtvsn}).(sortVsNb, r_{svsn}).(calTnbd, r_{ctnd}).VVM2;$$

$$VVM2 \overset{\text{def}}{=} (reqViNb, r_{rqvin}).(returnViNb, r_{rtvin}).(sortViNb, r_{rvin}).(calTnbid, r_{ctni}).VVM3;$$

$$VVM3 \overset{\text{def}}{=} (calTnb, r_{ctn}).VVM4;$$

$$VVM4 \overset{\text{def}}{=} (reqVsFi, r_{rqvsfi}).(returnVsFi, r_{rtvsfi}).(sortVsIf, r_{svsi}).(calTif, r_{cti}).VVM5;$$

$$VVM5 \overset{\text{def}}{=} (reqVsFe, r_{rqvsfe}).(returnVsFe, r_{rtvsfe}).(sortVsEf, r_{svse}).(calTef, r_{cte}).VVM6;$$

$$VVM6 \overset{\text{def}}{=} (calTf, r_{ctf}).VVM7;$$

$$VVM7 \overset{\text{def}}{=} (reqVsHis, r_{rqvsh}).(returnVsHis, r_{rtvsh}).(sortVsHis, r_{svsh}).(calThis, r_{cth}).VVM8;$$

$$VVM8 \overset{\text{def}}{=} (calTrust, r_{ct}).VVM9;$$

$$VVM9 \overset{\text{def}}{=} (updateTrust, r_{ut}).(returnTrust, r_{rt}).VVM;$$

$$\boldsymbol{Sys} \overset{\text{def}}{=} VN[m] \underset{\mathrm{L}}{\bowtie} VVM[n]$$

$$L = \{getTrust,\ returnTrust\}$$

As shown in Model 2, the *CS* and *RSS* components are merged with *VVM* by modifying preceding passive rates for actions cooperating with *CS* and *RSS* in *VVM*. Consequently, Model 2 retains just two components, which generates a much smaller state space compared with Model 1.

However, the core activities of the trust evaluation process are almost preserved, which means that these two models can be considered equivalently. This equivalence has been proved by Zhao [19], who has performed joint work with authors of this paper.

In the second step, as PEPA develops a strong notion of equivalence between components called *isomorphism*, models can be aggregated, but their underlying Markov chains are conserved, which is called *strong isomorphism*. Moreover, if we ignore unobserved activities (for example, if a component carries out several consecutive and unobserved activities), it is better to find a compact forms of model definitions that perform the same visible behavior but a single "Passive Activity" (aggregating a set of unobserved actions) of longer duration. This is termed *week isomorphism*.

Model 3:

$$\begin{aligned}
\boldsymbol{VVM} &\stackrel{\text{def}}{=} (getTrust, \tau).VVM1;\\
1VVM1 &\stackrel{\text{def}}{=} (calTnb, r_{ctn}).VVM2;\\
VVM2 &\stackrel{\text{def}}{=} (calTf, r_{ctf}).VVM3;\\
VVM3 &\stackrel{\text{def}}{=} (calThis, r_{cth}).VVM4;\\
VVM4 &\stackrel{\text{def}}{=} (calTrust, r_{ct}).VVM5;\\
VVM5 &\stackrel{\text{def}}{=} (returnTrust, r_{rt}).VVM;\\
\boldsymbol{VN} &\stackrel{\text{def}}{=} (getTrust, r_{gt}).VN1;\\
VN1 &\stackrel{\text{def}}{=} (returnTrust, \tau).VN;\\
\boldsymbol{Sys} &\stackrel{\text{def}}{=} VN[m] \underset{\mathrm{L}}{\bowtie} VVM[n]\\
L &= \{getTrust,\ returnTrust\}
\end{aligned}$$

For each aggregated action, its rate should be updated as a lump sum of each rate before aggregation, which is given by the following expression:

$$r_\chi = (\frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3} + \cdots + \frac{1}{r_n})^{-1}$$

According to the specification of weak isomorphism [18], unobserved T actions can be considered as a lumpable state in the underlying Markov chain. Hence, we can obtain a simpler form of Model 3. In contrast to preceding models, Model 3 just represents the observed activity behaviors, such as *calTnb* (calculation of T_{neig}), *calTf* (calculation of T_{fri}), *calThis* (calculation of T_{his}) and *calTrust* (calculation of overall trust $T(V_s \rightarrow V_r)$). This simplified model structure can greatly reduce the scale of the state space. Furthermore, Model 3 comprehensively represents the four key steps of obtaining the overall trust value, in which we can observe each calculation step and generate performance evaluations. If we just evaluate the performance of the whole calculation process rather than that of each detailed step, Model 3 can be aggregated in more depth through the weak-isomorphic lumping techniques. Model 3 is, accordingly, lumped into a minimalist style that abstracts all details in a single action. Model 4 achieves this extreme situation.

Model 4:

$$\begin{aligned}
\boldsymbol{VN} &\stackrel{\text{def}}{=} (getTrust, r_{gt}).VN;\\
\boldsymbol{VVM} &\stackrel{\text{def}}{=} (getTrust, \tau).VVM1;\\
VVM1 &\stackrel{\text{def}}{=} (calTrust, r_\chi).VVM;\\
\boldsymbol{Sys} &\stackrel{\text{def}}{=} VN[m] \underset{\mathrm{L}}{\bowtie} VVM[n]\\
L &= \{getTrust,\ returnTrust\}
\end{aligned}$$

In Model 4, all details of calculating trust values based on the general trust model are compressed into a single action *calTrust* with a lumpable rate r_χ. In the following sections, numerical analysis will be conducted based on Model 4, which benefits from a super-small state space scale, and fluid analysis will be implemented with Model 3 by considering detailed calculation steps.

With the simplified PEPA model, numerical analysis can be conducted by directly solving the underlying Markov chains. Section 7 illustrates the basic principles of numerical analysis and corresponding results.

VII. IMMEDIATE NUMERICAL ANALYSIS

A. PERFORMANCE ANALYSIS PRINCIPLES ON THE UNDERLYING STOCHASTIC MODEL

In the PEPA model, each component may experience an activity and behave as a new component,$P \xrightarrow{(\alpha,r)} P'$. P' is a one-step derivative of P. If $P \xrightarrow{(\alpha_1,r_1)} \cdots \xrightarrow{(\alpha_n,r_n)} P'$, P' is the derivative of P. For each component, all derivatives form a derivative set, $ds(C)$. Thus, the PEPA model can be represented as a derivative graph, $D(C)$, in terms of the derivative set of the system. According to PEPA theory, for any activity a in the activity set $Act(P)$, the activity duration is assumed to be an exponentially distributed random variable, and the resulting stochastic PEPA model is a continuous time Markov process due to the memoryless property of the exponential distribution. Before exploring the analysis based on the PEPA model, we need to introduce some terminology. The sojourn time of a component C is an exponentially distributed parameter that is the sum of the activity rates of the activities enabled by C. The exit rate is the rate at which the system leaves the state corresponding to the component C, denoted as $q(C)$. The sojourn time is $q(C)^{-1}$. The transition rate is the rate at which the system changes from state C_i to state C_j, denoted by:

$$q(C_i, C_j) = \sum_{a \in Act(C_i|C_j)} r_a, \quad where$$

$$Act(C_i \mid C_j) = \{\mid a \in Act(C_i) \mid C_i \xrightarrow{a} C_j\}$$

According to Hillston's specification [18], the PEPA model describes an underlying Markov process Q, in which has

the $q(C_i, C_j)$ is the off-diagonal element of the infinitesimal generator matrix of Q.

$$Pr(X(t+\delta t) = C_j \mid X(t) = Ci) = q(C_i, C_j)\delta t + o(\delta t), \quad i \neq j$$

Each diagonal element is equal to the negative sum of the non-diagonal elements of the corresponding row: $q_{ii} = -Q(C_i)$. The steady-state probability distribution of the system can be calculated by solving the matrix equation, $\Pi Q = 0$, with the normalization condition $\sum \Pi(C_i) = 1$ using Gaussian elimination.

$\Pi(\cdot)$ is the steady state probability of the model for each derivative, and $\Pi(C_i)$ is the probability of the model behaving as the derivative C_i.

Performance analyses, such as the utilization and throughput, can usually be derived from the steady-state distribution $\Pi(\cdot)$. In PEPA, reward structures, by Howard [17], are used to define the performance measures. Rewards are related to the states of the Markov process or the transition between states. PEPA uses the *reward structure* based on the yield function, which is a continuous accumulated reward, while a process is within a state. The PEPA models focus on behaviours based on activities, rather than states, and rewards are associated with the activities in the system. The reward associated with the component, and the corresponding state is the sum of the rewards attached to the activities that it enables. If ρ_i is the reward associated with the component C_i, the total reward R can be computed:

$$R = \sum_i \rho_i \Pi(C_i)$$

Thereafter, the performance measures of throughput and utilization can be obtained from R and $\prod(\cdot)$:

$$Utilization(C_i) = R = \sum_i \rho_i \Pi(C_i) \tag{1}$$

In the utilization formula (1), ρ_i is equal to the total number of activities enabled by C_i at each possible state of the underlying process X_i.

$$Throughput = \sum_i \rho_i \Pi(X_i) \tag{2}$$

In the throughput formula, ρ_i is equal to the product of the number of C_i and the transition rate behaving as C_i,

$$\rho_i = n(C_i) \times r_i$$

Finally, when the system reaches the steady state, we obtain the average response time $Response\_time(C_i)$ at component C_i, based on Little's law:

$$Response\_time(C_i) = \frac{Q(C_i)}{\lambda} \tag{3}$$

where $Q(C_i)$ is the average number of event at component C_i in the system and λ is the arrival rate.

B. NUMERICAL ANALYSIS RESULTS

The performance of the trust model is mainly determined by the implementation of the model at the VSNs, which exactly means the competition for resources in the process of trust evaluation in the *VVM* component. Now, we aim to solve two related questions: How many service requests can a given *VVM* configuration support? How much capacity should be planned to satisfy a given number of clients? To address these issues, numerical analysis is conducted to generate performance analyses and capacity planning for *VVM* components based on various numbers of user requests and capabilities of service.

As numerical analysis is based on directly solving of the Markov chains, it is easy to encounter a state space explosion problem, especially as the number of instances at the components increases. Initially, the analysis will be generated for each detailed trust calculation steps (e.g., neighbor trust, friend trust and history trust) in terms of Model 3. To avoid the possibly of encountering state space explosion, we need to restrain the scale of the state space by specifying small values of parameters, especially the numbers of *VN* and *VVM* as well as related rates. Hence, we set the initial parameters values as shown in Table 1.

TABLE 1. Parameter Setting Based on Model 3

Variables	Descriptions	Rates
N_{VN}	number of vehicular nodes	1
N_{VVM}	number of vehicular virtual machines	1
r_{gt}	rate of requesting service	2.0
r_{ctn}	rate of calculating neighbour trust	1.0
r_{ctf}	rate of calculating friend trust	1.0
r_{cth}	rate of calculating history trust	2.0
r_{ct}	rate of calculating overall trust	4.0

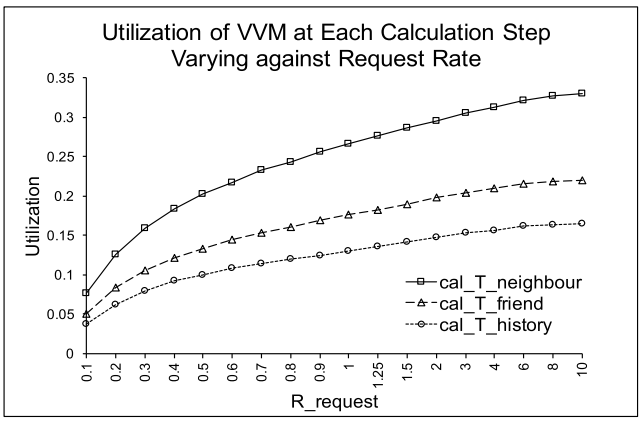


FIGURE 4. Utilization of VVM at each calculation step of trust model varying against request rate.

The initial analysis is conducted, in the first step, based on Model 3 and the related parameters shown in Table 1, and the corresponding outcomes can be viewed in Fig. 4 and Fig. 5. These figures represent the utilization of the *VVM* component but are based on each calculation step, including neighbor

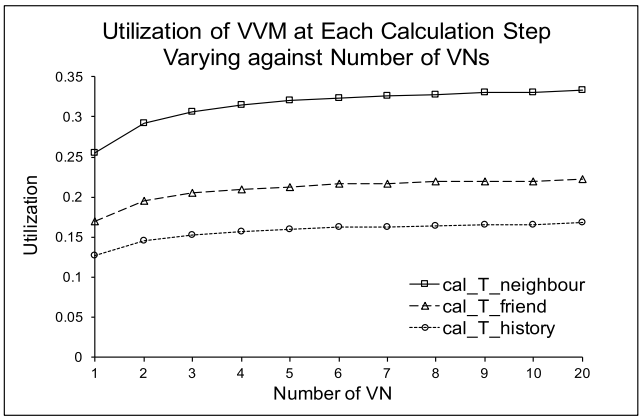


FIGURE 5. Utilization of VVM at each calculation step of trust model varying against number of VN.

trust calculation, friend trust calculation and history trust calculation. Fig. 4 clearly shows that the utilization of *VVM* at each calculation step increases slowly with growing request rates. Fig. 5 similarly depicts a growing trend of *VVM* utilization with a growing number of vehicular nodes. With the increase of workload, utilization grows until reaching a stable level, which indicates a full-load working situation. The analysis in this step focuses on answering the first question raised at the beginning of section 7.2.

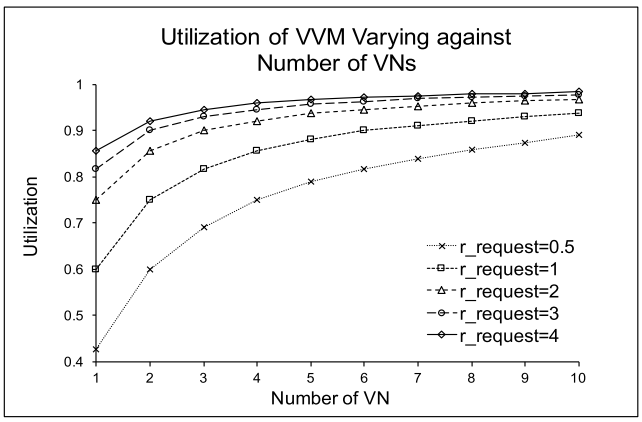


FIGURE 6. Utilization of VVM component varying against number of VNs.

In the second step, the numerical analysis mainly considers performance evaluation of the whole trust calculation process, which means that measurements will be based on component *VVM* as a whole. Model 4 will be used for numerical analysis by solving its underlying Markov Chains. Fig. 6 displays the overall utilization of *VVM* with a growing number of *VNs* as well as various request rates (r_{gt}). It is clear in Fig. 6 that utilization increases and tends to a stable level that finally approaches to the value 1 with increasing number of *VN*. However, on the other hand, the request rate also dominates the trends of the lines. Higher request rates cause their related lines in the the plot to move to higher locations, representing a better utilization.

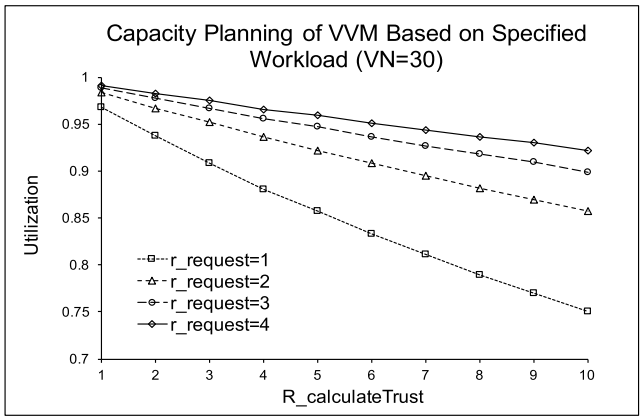


FIGURE 7. Capacity of VVM component based on specified workload.

In the third step, we are trying to explore the capacity planning issue based on a given configuration of customer requirements, namely, the number of *VNs* and the request rate. Fig. 7 shows the utilization of *VVM* varying against the service rate of *VVM* under the condition $N_{VN} = 30$. As shown in the figure, utilization decreases with the increased service rate. This means that when the workload is given, the increased computing capability will cause increased wasting of resources, which is represented by the reduced utilization. Actually, if we expect the utilization to be ideally greater than 95 %, the required computing capability is approximately 3 for the service rate when the request rate is 2; similarly, when the request rate is increased to 3, the corresponding service rate should be configured to approximately 5, which corresponds to 95 % utilization.

Numerical analysis through directly solving Markov chians has the drawback of limited model scale. With an increased number of instances, the state space scale enlarges exponentially which causes a state space explosion. Direct numerical analysis, as a result, cannot be conducted. Thus, a new analysis solution should be adopted to overcome the state space explosion problem.

In the next section, a novel analysis technique will be introduced to solve this problem: fluid-flow approximation, implemented by solving a set of ordinary differential equations (ODEs) derived from PEPA models.

VIII. EFFICIENT FLUID FLOW APPROXIMATION

To extend the use of PEPA for modelling large scale systems, a fluid analysis technique is considered to approximate the well-known approach of analysis via CTMCs. In this section, an alternative approach, developed by Hillston [20], will be applied for related analysis in the performance and capacity aspects. Details of the fluid-flow approximation will be given in the following subsections, especially on how to generate ODEs from a basic PEPA model and some corresponding analysis outcomes.

A. FLUID FLOW APPROXIMATION

PEPA language offers a compositional function for creating models of large-scale systems. Meanwhile, a novel

performance analysis technique, fluid-flow approximation, is provided for large-scale models using PEPA.

Similar to most discrete state-based modeling formalisms, process algebra easily suffers from failure due to the generation of extremely large state spaces, making obtaining the numerical solution via linear algebra costly or even intractable. Fluid-flow approximation generates a set of coupled ordinary differential equations (ODEs) underlying CTMCs. In this approach, two adjustments are made. First, it does not calculate the probability distribution over the entire state space; instead, a more abstract state representation is chosen based on state variables [20]. Second, it is assumed that these state variables are subject to continuous rather than discrete changes [20]. Based on these adjustments, fluid-flow approximation offers an efficient solution with a set of ODEs, and leads to the evaluation of transient and steady-state measures. This approach successfully avoids state space explosion in the analysis via exploring ODEs.

The fluid-flow analysis is performed on the basis of vector form. The system is inherently discrete, with the entries within the numerical vector form always being non-negative. With a change in the system state, the numerical vector form is incremented or decremented in steps of one. When each component type in the model is replicated a large number of times, these steps are relatively small. Thus, we can approximate the movement between states as continuous, rather than occurring in discontinuous jumps. The objective of the fluid-flow approximation is to replace the derivative graph of the PEPA model with a continuous model using a set of ordinary differential equations.

In the fluid-flow approximation, we need to specify the *exit activity* and *entry activity* of the local derivative of a sequential component. An activity (α, r) is an *exit activity* of D if D enables (α, r), such as $D \xrightarrow{(\alpha,r)} D'$. The set of exit activities of D is denoted by $Ex(D)$. The set of local derivatives for an exit activity (α, r) is denoted by $Ex(\alpha, r)$. Similarly, an activity (β, s) is an *entry activity* if a derivative D' enables (β, s), such as $D' \xrightarrow{(\beta,s)} D$. $En(D)$ denotes the set of entry activities of D.

After specifying the concepts of the *exit activity* and *entry activity*, the movements of the numerical state vector of the PEPA model are represented with these concepts. Here, we define $v_{i_j}(t) = N(C_{i_j}, t)$ for the jth entry of the ith sub-vector at time t; $N(C_{i_j}, t)$ denotes the number of instances of the jth local derivative of sequential component C_i. In a very short time interval δt, the change of the vector entry $v_{i_j}(t)$ can be expressed as:

$$\begin{aligned} &N(C_{i_j}, t+\delta t) - N(C_{i_j}, t) \\ &= \underbrace{-\sum_{(\alpha,r)\in Ex(C_{i_j})} r \times \min_{C_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t))\,\delta t}_{exit\ activities} \\ &\quad + \underbrace{\sum_{(\alpha,r)\in En(C_{i_j})} r \times \min_{C_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t))\,\delta t}_{entry\ activities} \end{aligned} \tag{4}$$

$$\begin{aligned} \frac{dN(C_{i_j}, t)}{dt} &= \lim_{\delta t\to 0} \frac{N(C_{i_j}, t+\delta t) - N(C_{i_j}, t)}{\delta t} \\ &= -\sum_{(\alpha,r)\in Ex(C_{i_j})} r \times \min_{C_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t)) \\ &\quad + \sum_{(\alpha,r)\in En(C_{i_j})} r \times \min_{C_{k_l}\in Ex(\alpha,r)} (N(C_{k_l}, t)) \end{aligned} \tag{5}$$

In formula (4), the first block represents the impact of exit activities, and the second block records the impact of the entry activities. Now, we can divide formula (4) by δt and take a limit. If $\delta t \to 0$, we obtain the formula (5). In the following analysis, a set of ODEs can be obtained from the PEPA model based on formula (5). The quantitative analysis is conducted through solving the ODEs.

In the following subsection, PEPA models are converted into a set of ODEs based on above equations (4) and (5). Related performance, such as throughput and queue length, are conducted by solving these ODEs.

B. FLUID-ANALYSIS RESULTS

To observe some particular performance properties (e.g., queue length and response time), measurements should be conducted on the component *VN*. Thus, the preceding model must be modified by defining action flows in *VN* instead of *VVM*. Thus, we need update Model 3 to a new schema, as follows:

***Model* 4**:

$$\begin{aligned} \boldsymbol{VN} &\overset{\text{def}}{=} (getTrust, r_{gt}).VN_1; \\ VN_1 &\overset{\text{def}}{=} (calTnb, \tau).VN_2; \\ VN_2 &\overset{\text{def}}{=} (calTf, \tau).VN_3; \\ VN_3 &\overset{\text{def}}{=} (calThis, \tau).VN_4; \\ VN_4 &\overset{\text{def}}{=} (calTrust, \tau).VN_5; \\ VN_5 &\overset{\text{def}}{=} (returnTrust, \tau).VN; \\ \boldsymbol{VVM} &\overset{\text{def}}{=} (calTnb, r_{ctn}).VVM + (calTf, r_{ctf}).VVM \\ &\quad + (calThis, r_{cth}).VVM + (calTrust, r_{ct}).VVM \\ &\quad + (returnTrust, r_{rt}).VVM; \\ \boldsymbol{Sys} &\overset{\text{def}}{=} VN[m] \underset{L}{\bowtie} VVM[n] \\ L &= \{calTnb,\ calTf,\ calThis,\ calTrust\} \end{aligned}$$

To conduct fluid-flow analysis, the PEPA model must be converted from its form of process algebra to the ordinary differential equations (ODEs). According to the principles of fluid-flow approximation, namely, Equations (8) and (9), we can obtain a set of ODEs from Model 4 as follows:

$$\begin{aligned} \frac{d}{dt}VN &= r_{rt} \cdot \min\ (VN_5(t),\ VVM(t)) \\ &\quad - r_{gt} \cdot VN(t); \\ \frac{d}{dt}VN_1 &= r_{gt} \cdot VN(t) \\ &\quad - r_{ctn} \cdot \min(VN_1(t),\ VVM(t)); \\ \frac{d}{dt}VN_2 &= r_{ctn} \cdot \min(VN_1(t),\ VVM(t)) \\ &\quad - r_{ctf} \cdot \min(VN_2(t),\ VVM(t)); \end{aligned}$$

$$\frac{d}{dt}VN_3 = r_{ctf} \cdot \min(VN_2(t),\ VVM(t)) - r_{cth} \cdot \min(VN_3(t),\ VVM(t));$$
$$\frac{d}{dt}VN_4 = r_{cth} \cdot \min(VN_3(t),\ VVM(t)) - r_{ct} \cdot \min\ (VN_4(t),\ VVM(t));$$
$$\frac{d}{dt}VN_5 = r_{ct} \cdot \min\ (VN_4(t),\ VVM(t)) - r_{rt} \cdot \min\ (VN_5(t),\ VVM(t));$$
$$\frac{d}{dt}VVM = 0;$$

Fluid analysis is performed by solving the above set of equations. Although several different approaches can be used to solve ODEs, we have simulated them over a long time (steady-state) situation. Based on the specifications of the fluid-flow approximation, the time step δt must be chosen as $\delta t \leq 1/r_{max} \cdot m$, where $r_{max} = \max(r_{ctn},\ r_{ctf},\ r_{cth},\ r_{ct},\ r_{rt})$.

In our analysis, we expect to measure the number of waiting vehicle nodes (queue length), denoted by L_m, and the average response time from VVM, denoted by R_m, where m is the number of VN instances. L_m can be obtained directly by solving the ODEs; however, the average response time must be calculated based on the queuing network approximation that is defined in [21] on page 141.

Performance analyses and capacity planning are implemented, taking advantages of the strengths of the above approaches, for both transient queue length and average response time.

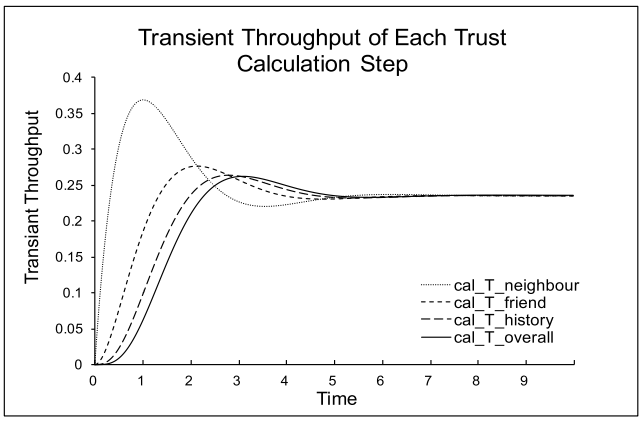


FIGURE 8. Transient throughput of VVM in each calculation step.

1) PERFORMANCE EVALUATION OF VVM INTERNAL BEHAVIORS

This part primarily demonstrates the transient performance of VVM internal behaviors through two aspects: throughput and queue length. Fig. 8 represents the transient throughput at each trust calculation step. It shows the change of throughput from the beginning to the same stable level. Moreover, Fig. 9 describes the transient queue length measured from each step of the calculation of trust values. All lines increase

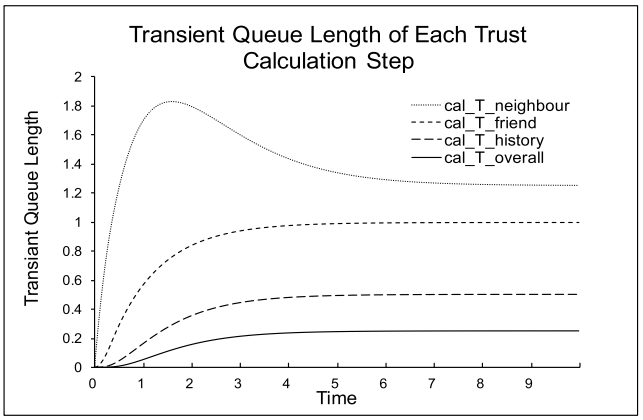


FIGURE 9. Transient queue length of VVM in each calculation step.

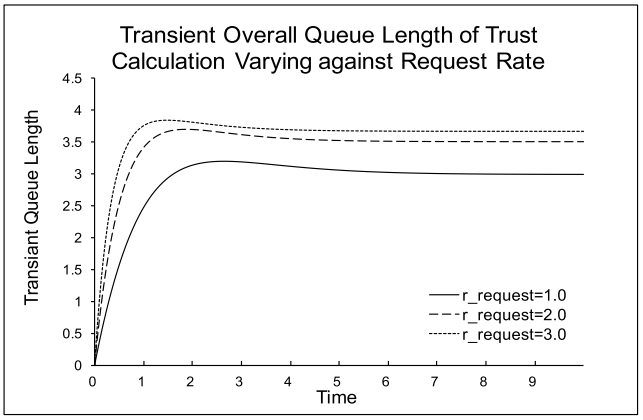


FIGURE 10. Transient overall queue length of trust calculation at VVM varying against request rate.

smoothly with time elapse and tend to stable levels eventually, which indicates a steady state of the complete system. Fig. 9 also reveals the various queuing distributions of each step, which are affected by the service capability at each step. Both Fig. 8 and Fig. 9 clearly show the fluid feature of this analysis approach, in which each line seems to be continuous.

2) PERFORMANCE EVALUATION OF VVM COMPONENT

Each component is commonly evaluated by measuring its comprehensive performance rather than its internal details. For this reason, we explore the overall transient queue length at the VVM component for different request rates; see Fig. 10. From the figure, it can be observed that the transient queue length tends to be stable after a sharp increase. A larger request rate means faster service demand of a specified vehicular node; the queue length corresponding to a larger rate, as a result, increases faster than the others and reaches a higher stable level. Fig. 11 reveals the same trend with increasing number of vehicular nodes (VN).

Hence, an increase of either the request rate or the number of instances can boost upward gradient of the queue length; normally, the larger the rate (or number of instances) is, the faster the rise.

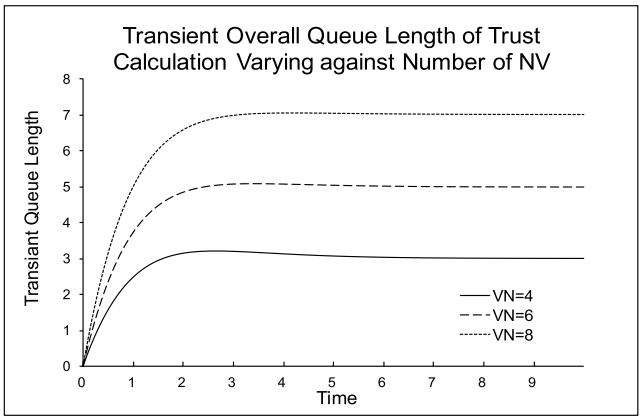


FIGURE 11. Transient overall queue length of trust calculation at VVM varying against number of VN.

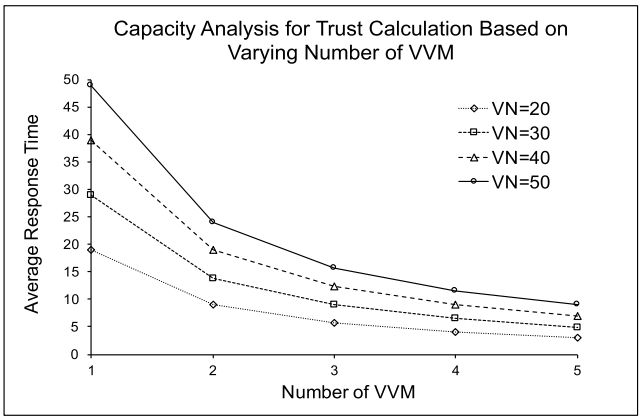


FIGURE 13. Capacity planning for trust calculation at VVM based on varying number of VVM.

3) CAPACITY ANALYSIS OF SYSTEMS

The response time is an alternative performance property, which represents the total time from generating a service request to completing the service. The response time can be observed from a steady state of the system representing a global time behavior of completing a task. In this case, we suppose each vehicular node generates only one service request per second, namely $r_{gt} = 1.0$ (Requests/Second); then the unit of response time is in seconds.

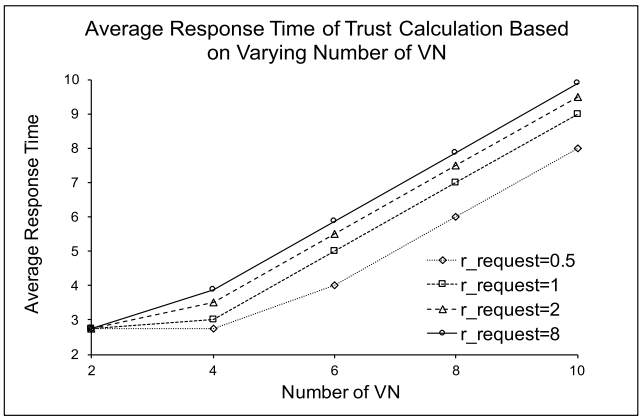


FIGURE 12. Average response time of trust calculation at VVM based on varying number of VN.

Fig. 12 depicts the change of the average response time at *VVM* based on varying request rates and numbers of vehicular nodes. Obviously, the average response time grows with increasing number of vehicular nodes. Meanwhile, a larger request rate also causes a longer average response time, according to the higher locations of the lines.

In capacity planning, the number of servers or the capability represented by the service rate will be considered in terms of the given customer requirements. For example, in Fig. 13, the given requirements are as follows: the number of vehicular nodes (VN), $m \in \{20, 30, 40, 50\}$ and the related request of each VN, $r_{gt} = 1.0$. Under these requirements, system capacity plan is investigated by measuring the average response time against varying numbers of *VVMs* and *VNs*. Fig. 13 represents the capacity analysis outcomes, in which the average response time decreases with increasing number of *VVM* servers. This means that a greater number of servers will result in a lower average response time. In capacity analysis, a system, for example, has a basic requirement for QoS, which is that the average response time must be less than 15 seconds. Thus, according to Fig. 13, at least 2 *VVM* servers are required when the number of *VNs* is 30; however, if the number of *VNs* increases to 40, the required number of *VVMs* should be at least 3.

4) SUMMARY

Fluid-flow analysis is able to directly generate a population distribution by solving ODEs. Thus, we can obtain throughput and queue length, such as in Figs. 8-11. The average response time can also be calculated using the queuing-network approximation. However, utilization cannot be obtained from the ODE-based fluid-flow analysis, which is a limitation of this form of analysis.

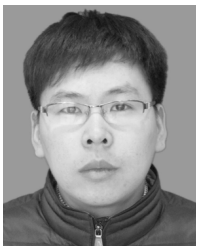
IX. CONCLUSION

This work primarily considers the trustworthiness issue in VSNs, which are a special type of social network joint with vehicular networks. Both humans and vehicles play main roles in such special networks. Moreover, individuals are able to break common rules to benefit themselves, and the traditional security mechanism (e.g., public key infrastructure) cannot guarantee the trustworthiness of authorized users in this situation. Based on this requirement, we proposed a cloud-based trust management framework and evaluated its performance through a formal method. The main achievements can be summarized as follows: this work, on the one hand, builds a framework that supports a trust management mechanism for a vehicular social network; on the other hand, we explore a novel performance modeling approach for this new application, as well as some related analysis methods. Our future research will be continued based on this work.

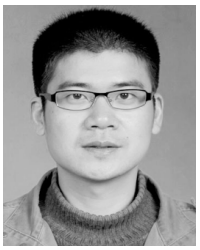
In the future, we will focus on developing a novel trust model to adapt requirements on the trustworthiness of VSNs by considering some psychological factors of VSN users. The trustworthiness issue is different from traditional security, as it is commonly affected by complex and uncertain human behaviors that are closely related to human psychology. Thus, it is necessary to consider these human psychological factors in the trust models to improve their applicability. Furthermore, experiments will be conducted by observing the number of malicious vehicular nodes in VSNs. The trust algorithm will be evaluated on the basis of *Recall* and *Precision*, which are both widely used in machine learning and information retrieval to assess the accuracy. Finally, new modeling techniques, for example a mobility model, will be explored to simulate dynamic network communications in VSNs.

REFERENCES

[1] F. Boccardi, R. W. Heath, Jr., A. Lozano, T. L. Marzetta, and P. Popovski, "Five disruptive technology directions for 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 74–80, Feb. 2014.
[2] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and devices for the 5G era," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 90–96, Feb. 2014.
[3] V. M. Vegni and V. Loscrí, "A survey on vehicular social networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2397–2419, 4th Quart., 2015.
[4] M. R. Schurgot, C. Comaniciu, and K. Jaffres-Runser, "Beyond traditional DTN routing: Social networks for opportunistic communication," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 155–162, Jul. 2012.
[5] A. Festag, "Standards for vehicular communication—From IEEE 802.11p to 5G," *E&I Elektrotech. Informationstechnik*, vol. 132, no. 7, pp. 409–416, Nov. 2015.
[6] S. K. Bhoi and P. M. Khilar, "Vehicular communication: A survey," *IET Netw.*, vol. 3, no. 3, pp. 204–217, Sep. 2014.
[7] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for Internet of Things," *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.
[8] T. H. Luan, X. Shen, F. Bai, and L. Sun, "Feel bored? Join verse! Engineering vehicular proximity social networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 3, pp. 1120–1131, Mar. 2015.
[9] Q. Yang and H. Wang, "Toward trustworthy vehicular social networks," *IEEE Commun. Mag.*, vol. 53, no. 8, pp. 42–47, Aug. 2015.
[10] W. Jiang, J. Wu, F. Li, G. Wang, and H. Zheng, "Trust evaluation in online social networks using generalized network flow," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 952–963, Mar. 2016.
[11] X. Meng and D. Liu, "GeTrust: A guarantee-based trust model in Chord-based P2P networks," *IEEE Trans. Depend. Sec. Comput.*, to be published, doi: 10.1109/TDSC.2016.2530720.
[12] R. Hussain, W. Nawaz, J. Y. Lee, J. Son, and J. T. Seo, "A hybrid trust management framework for vehicular social networks," in *Proc. CSoNet*, 2016, pp. 214–225.
[13] W. Li and H. Song, "ART: An attack-resistant trust management scheme for securing vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 960–969, Apr. 2016.
[14] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *J. Netw. Comput. Appl.*, vol. 40, pp. 325–344, Apr. 2014.
[15] S. Bitam, A. Mellouk, and S. Zeadally, "VANET-cloud: A generic cloud computing model for vehicular Ad Hoc networks," *IEEE Wireless Commun.*, vol. 22, no. 1, pp. 96–102, Feb. 2015.
[16] X. Chen and L. Wang, "Exploring trusted data dissemination in a vehicular social network with a formal compositional approach," in *Proc. IEEE 40th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Atlanta, GA, USA, Jun. 2016, pp. 616–617.
[17] X. Chen and L. Wang, "Exploring fog computing based adaptive vehicular data scheduling policies through a compositional formal method—PEPA," *IEEE Commun. Lett.*, to be published, doi: 10.1109/LCOMM.2016.2647595.
[18] J. Hillston, *A Compositional Approach to Performance Modelling*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[19] Y. Zhao and N. Thomas, "Efficient solutions of a PEPA model of a key distribution centre," *Perform. Eval.*, vol. 67, no. 8, pp. 740–756, Aug. 2010.
[20] J. Hillston, "Fluid flow approximation of PEPA models," in *Proc. IEEE 2nd Int. Conf. Quant. Eval. Syst.*, Sep. 2005, pp. 33–42.
[21] I. Mitrani, *Probabilistic Modelling*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[22] X. Chen, L. Wang, J. Ding, and N. Thomas, "Patient flow scheduling and capacity planning in a smart hospital environment," *IEEE Access*, vol. 4, pp. 135–148, 2016.



XIAO CHEN received the master's and Ph.D. degrees of Computing Science from Newcastle University, U.K., in 2008 and 2013, respectively. He is currently a Lecturer with Jiangsu University, China. His research interests include formal modelling based on stochastic process algebra and performance evaluation for large scale systems, such as vehicular networks, cloud systems, and cyber-physical systems. His current research is in modelling of vehicular cloud systems and trust issues of vehicular social networks.



LIANGMIN WANG received the B.S. degree in computational mathematics from Jilin University, Changchun, China, in 1999, and the Ph.D. degree in cryptology from Xidian University, Xi'an, China, in 2007. He is currently a Full Professor with the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. He has published over 60 technical papers at international journals and conferences. His current research interests include security protocols and Internet of Things.