Providing healthcare services on-the-fly using multi-player cooperation game theory in Internet of Vehicles (IoV) environment

Neeraj Kumar\textsuperscript{a}, Kuljeet Kaur\textsuperscript{a}, Anish Jindal\textsuperscript{a}, Joel J.P.C. Rodrigues\textsuperscript{b,c,n*}

\textsuperscript{a}Department of Computer Science and Engineering, Thapar University, Patiala, India
\textsuperscript{b}Instituto de Telecomunicações, University of Beira Interior, Portugal
\textsuperscript{c}ITMO University, St. Petersburg, Russia

Received 31 December 2014; received in revised form 19 March 2015; accepted 15 May 2015
Available online 18 June 2015

Abstract

Internet of Vehicles (IoV) is a leading technology of the present era. It has gained huge attention with respect to its implementation in wide variety of domains ranging from traffic safety to infotainment applications. However, IoV can also be extended to healthcare domain, where the patients can be provided healthcare services on-the-fly. We extend this novel concept in this paper and refer it as “Healthcare services on-the-fly”. The concept of game theory has been used among the vehicles to access the healthcare services while traveling. The vehicles act as players in the game and tend to form and split coalitions to access these services. Learning automata (LA) act as the players for interaction with the environment and take appropriate actions based on reward and penalty. Apart from this, Virtual Machine (VM) scheduling algorithm for efficient utilization of resources at cloud level has also been formulated. A stochastic reward net (SRN)-based model is used to represent the coalition formation and splitting with respect to availability of resources at cloud level. The performance of the proposed scheme is evaluated using various performance evaluation metrics. The results obtained prove the effectiveness of the proposed scheme in comparison to the best, first, and random fit schemes.

© 2015 Chongqing University of Posts and Telecommunications. Production and Hosting by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

The advancements in Internet have led to the emergence of various technologies. Internet-of-Things (IoT) is one such technology in which numerous devices over the Internet are inter-connected and communicate with one another using
different protocols and standards. More than 24 billions devices are expected to be interconnected by 2020 [1]. These devices include laptops, desktops, powerful servers, sensors, and vehicles. If all the interconnected devices are considered to be vehicles, then it is called Internet-of-Vehicles (IoV). IoV is used for providing communication amongst vehicles in order to support various applications ranging from data transmission for traffic safety to infotainment services. Vehicles act as intelligent machines having on-board computing, communication, sensing and storage capabilities. Due to availability of advanced hardware and software resources nowadays, vehicular communication have affected many applications. This communication is provided through Vehicular Ad-hoc Networks (VANETs).

VANETs are being integrated with other technologies like cloud computing to enhance their reliability and scalability [2-6]. One of the prominent applications where VANETs are being implemented is healthcare domain. Use of VANETs in this domain means healthcare services could be extended to patients on-the-fly. Such an integration can be life-saving for millions of people who may require medical facilities during traveling. This paper highlights such a scheme which is referred to as “E-healthcare services on-the-fly in IoV environment”. In this scheme, patients’ physiological data are relayed to a cloud server in real-time. Based on the data received, the cloud server allocates appropriate number of resources to the vehicle under consideration on the basis of patients’ current health conditions. Implementing real-time E-healthcare services not only requires dedicated resources but also an appropriate resource allocation mechanism for accessing the available services. Hence, in order to fulfill these requirements, we propose a system coupled with cloud computing platform to handle healthcare related data in a seamless manner. On the other hand, resource allocation for the vehicles is done based on the proposed game theoretical model. Apart from this, there also arises a need to handle the available cloud resources efficiently. This can be gracefully handled with the help of proposed Virtual Machine (VM) scheduling mechanism. Thus, keeping these issues in mind, a novel model is proposed to cater the requests for accessing health services on-the-fly using cloud computing, game theoretical model and VM scheduling.

SRNs and game theory have been used for modeling various real world problems efficiently [7-12]. For proper utilization of resources at cloud level and to handle large number of requests from the users, an extra care is taken for Virtual Machine (VM) scheduling also, i.e., an efficient VM Scheduling algorithm is required which can efficiently handle the available resources while serving the requests from the users.

1.1. Motivation

In US, more than 30,000 causalities and 2 million other injuries occurred in 2009 due to motor vehicle crashes. The estimated loss due to these crashes was about $230 billion. In addition to it, the congestion at highways cost $78 billion annually [5] and 8.4 billion gallons of fuel was being wasted annually [6]. Lot of people die on roads even before reaching the hospitals due to congestion and traffic jams. So, an application like E-healthcare on-the-fly is the need of the hour. It provide healthcare services to the patients on-the-move. These services need to be handled in real-time. Thus, providing necessary medical services at the earliest, thereby save millions of lives.

Such an E-healthcare system would require distributed data repositories to store enormous amount of heterogeneous data generated from the vehicles. Traditional storage and computational resources would not be suffice for these requirements. Thus, distributed clouds at different levels are required to provide timely services to the end users. There are various implications involved in providing E-healthcare services with respect to IoV environment. Timely response to patients is one of the most important issues. The mobility of vehicles is also a crucial issue while providing the response to the vehicles. All these issues have to be handled in order to provide seamless services to the end users.

1.2. Contributions

In this paper, we propose a novel E-healthcare model coupled with cloud computing platform to provide health related services in IoV environment on-the-fly. The proposed model is based on game-theoretical approach to calculate payoff and VM scheduling mechanism to handle the resources on the cloud. The major contributions of the paper are summarized as follows.

- A system model for providing healthcare services in vehicular cloud environment is proposed. In this model, different modes of accessing various services from the cloud are highlighted.
- A game theoretical model has been presented to form a game amongst the vehicles to prioritize the patients for accessing the services. Thus, coalition formation and splitting algorithms are designed for each player in the game in which a unique payoff function is assigned to each player based upon the available resources in the game.
- A stochastic reward net (SRN)-based model is used for representing different states and transitions of the players. These players take adaptive decisions on the basis of proposed learning automata (LA)-based game model.
- Finally, a virtual machine (VM) scheduling algorithm is designed for efficient resource utilization to serve vehicles’ requests at cloud level.

1.3. Organization

The rest of the paper is organized as follows. Section 2 highlights the related work with respect to the proposed model. Section 3 provides preliminaries and background details. Section 4 describes the system model along with problem identification. Section 5 elaborates the proposed solution. Section 6 presents the results obtained. Section 7 concludes this paper with future directions.

2. Related works

There are a number of research proposals addressing the issues related to healthcare applications in IoV environment with support from the cloud. Some of the most prominent research proposals in this area are summarized as follows. Dua et al. [13] discussed various routing protocols according to their applicability in particular application. Different services use different
types of resources and these services can be integrated in vehicular environment to share resources so as to efficiently manage the resources [2, 14]. Vehicular clouds are used in many places and applications like in parking lots, malls, synchronizing traffic lights, and self-organizing high occupancy vehicle lanes [4, 15]. Gerla and Kleinrock [3] proposed a vehicle based emergency network which operated with intermittent connectivity using peer-to-peer content sharing among the nodes. Authors described four major characteristics for this scenario as content downloading (location dependent content), P2P location advertisement, P2P interaction, and sensing the environment. Authors also discussed how emergency routing was done when the grid failed. Information from the vehicles can be fetched in ad-hoc manner and routing of packets to their destination can be done dynamically using this information [16].

Various challenges such as security and privacy, sensor filtering, and secure networking [17] have been discussed in IoV with respect to modes of communications such as vehicle-to-vehicle (V2V), vehicle-to-sensors (V2S), vehicle-to-road infrastructure (V2R) and vehicle-to-Internet (V2I). Some of the wireless solutions for these types of communications have been reviewed in [18]. Also, usage of mobile clouds in healthcare services with respect to the data rate and routing in such scenarios have been discussed [19, 20]. Researchers have also studied various scheduling factors in virtual machines and clouds like VM migration, offloading, handoff and optimal VM allocation with respect to various available resources [21-24].

SRNs had been used to model various real-time systems addressing various problems as discussed in the following proposals. Ibe and Trivedi [7] presented stochastic petri nets for different types of polling systems. Dependability models were generated using stochastic petri nets based modeling in [8]. Maleki et al. [9] evaluated the performances of various grid environments using SRNs. Three approximation models namely - Exact, Folded and Fixed-point approximation were studied for analyzing the behavior of the system.

Many research proposals have used game theory-based approach to form coalition and Payoff Function (PF) for various players in the game. A Bayesian coalition game model with non-transferable utility was discussed in [10]. A belief update mechanism based on Bayes theorem was used to observe the behavior of other nodes in coalition. An approach was used to form coalition that was based on cooperative game theory in which distributed-agents were evaluated for smart grids [11]. Misra et al. discussed the necessity of bandwidth redistribution and used auction based mechanism to maximize the utility function so that revenue generated for each gateway is maximum [12].

There are a number of different techniques for scheduling the VMs as discussed in the following proposals. Shiraz et al. [25] used VM-based approach to outsource computation intensive applications to resource rich machines either partially or entirely. A genetic algorithm was used to schedule cloud resources based on number of VMs and number of tasks to be performed [26]. Ghribi et al. [27] proposed an energy-efficient VM allocation and migration algorithm by using bin-packing approach with best-fit algorithm. A decentralized VM migration approach was presented in [28] which uses load information on each physical node in order to make load migration decisions. Authors defined two threshold values called “lower threshold” and ‘upper threshold’. The aim of latter was to provide extra resources to unpredicted workload rise and former was used to switch nodes which were not utilized much into sleep mode. Wang et al. [29] predicted future bandwidth consumption of a VM with the help of Hidden Markov Model (HMM) for providing tele-health services in urgency and normal conditions. A bandwidth-aware task scheduling algorithm was proposed in [30] in which a non-linear programming model was formed which tried to minimize the total time needed to finish all the tasks. Kumar et al. presented a RFID-enabled Elliptic Curve Cryptography (ECC)-based authentication mechanism to enhance the security of healthcare application to provide medical services to the patients while traveling [31].

Apart from the above research proposals, there are a number of other proposals in the literature addressing the issues of applicability of computational intelligence techniques for solving various real world problems such as [32-41].

3. Background and preliminaries on LA and PN model

3.1. Learning automata

Learning automata (LA) is a piece of code that executes the learning algorithm to produce the best output. It gives input to the environment and based on the output received from the environment. It changes its action as shown in Fig. 1. In the proposed model, LA is deployed on the vehicles which helps them to interact with neighboring vehicles in IoV environment.

LA are represented as follows: \( \text{LA} = (Q, K, P, V, G) \) where \( Q = \{q_1, q_2, \ldots, q_n\} \) is a finite set of states of LA, \( K = \{k_1, k_2, \ldots, k_q\} \) is a finite set of actions performed by it, \( R = \{r_1, r_2, \ldots, r_p\} \) is a finite set of responses received from its environment, \( V : Q \times R \rightarrow Q \) maps the current state and input from environment to the next possible state of automaton and \( G \) is a function which maps the current state with respect to input [32-41].

On the basis of vehicle interaction with its environment, an automaton either gets awarded or penalized. The prime objective of an automaton is to interact with the environment such that it gets minimum penalty from environment and produces best output.

3.2. PN model

The brief explanation about the PN modeling is as follows. PNs are directed bipartite graphs that are used for representing the control and flow of information in the system [7]. These are

![Fig. 1 Learning automata.](image-url)
specified by quadruple $[8]$ which contains Places, Transitions, Arcs and Initial Markings. $PN = (P, T, A, M_0)$, where $P = \{P_1, P_2, P_3, \ldots\}$ is set of places, $T = \{t_1, t_2, t_3, \ldots\}$ is set of transitions, $A \subseteq (P \times T) \cup (T \times P)$ is set of arcs, $M_0 = \{m_0, m_1, m_2, \ldots\}$ is initial markings, where $m_i = 0, 1, 2, \ldots; i \in [1, n]$.

PNs usually consist of two types of nodes: Places and Conditions, connected via directed arcs. Places are further categorized into two types: Input and Output places and are used to represent various states of the underlying system. Graphically, these places are represented in the form of circles. On the other hand, transitions are represented graphically by solid bars and are refer to the events that cause changes in the states of the concerned systems. Places and transitions are connected via directed lines called arcs. These arcs are of two types: Input and Output arcs. Input arc connects an input place with its respective transition, and is used to depict the state that needs to be satisfied in order to trigger the related transition. While, the output arc connects a transition with its corresponding output place and is used to depict an occurrence of an event. Place(s) might contain one or more token(s). Tokens are depicted graphically by dots. The number of tokens that belongs to specific state is referred to as marking. Arcs have an associated weight/cardinality referred to as multiplicity (default value 1). An event is said to be enabled if the number of tokens at input place is equal to the multiplicity of the corresponding arc $[7]$. These enabled events get fire and tend to change the state of the system. During this process, they tend to deposit tokens at the output place equivalent to the multiplicity of the respective arc. The basic architecture of a simple Petri Net is shown in Fig. 2 $[31]$.

4. System model

The system model for providing healthcare services in IoV environment is shown in Fig. 3.

4.1. Layered architecture

The system is divided into two layers namely - Acquisition layer, and Communication and computation layer. The working of these layers is described as follows.

4.1.1. Acquisition layer

The system is based on the assumption that the body sensors are deployed in the vehicles and patients’ physiological data are captured using these sensors. This assumption can easily be satisfied because the captured data can easily be preprocessed on vehicles as they have on-board computing and storage functionality. The vehicles form various coalitions based on the concept of game theory. Each vehicle acts as a player in the game whose objective is to seek the services from the cloud. Every player is assigned a unique PF based on parameters such as - service priority index, resource and bandwidth capacity of the vehicle.

LA are assumed to be deployed on the vehicles and are responsible for interacting with the environment or with the surrounding vehicles. In the proposed solution, we have
assumed that mobile cloud is the environment where all the players perform actions based upon the feedback from the environment. The environment can be defined by the parameters such as number of inputs, reinforcement signal, and penalty probabilities associated with respect to all the actions taken by the players in the game. The environment may also be stochastic or probabilistic, based upon the feedback given by it to LA, which may be a random or probabilistic variable. According to the response received from the environment, LA decide their actions by taking the reinforcement signal to the stage where they have to move. There are two types of feedback that an environment can give to all the players with respect to the actions to be taken. These actions are reward and penalty. Each player has an action probability vector associated with it which is updated after each action performed by a player in the game.

Let us assume that \( p_j(n) \) represents the action probability vector at an instant \( n \) for taking an action \( j \). For each pair of \( i \) and \( j \), there are two sets of equations; one set is for reward, and other set is for penalty. For Reward set, one combination is for the scenario where source and destination are equal, i.e., \( i=j \), and second combination is where source and destination are not equal, \( i \neq j \). In the current solution, we assumed a Linear Reward-Inaction scheme (\( LRI_{i,j} \)) in which if the LA receives reward from the environment, then the action probability is updated according to Eq. (1). Otherwise, the probability remains the same as described in Eq. (2) [32-41]. Following equations are used for both rewards and penalty in the proposed scheme: For reward

\[
p_j(n+1) = \begin{cases} p_j(n) + \alpha (1 - p_j(n)) & j = i \\ (1 - p_j(n)) & j \neq i \end{cases}
\]

(1)

For penalty

\[
p_j(n+1) = \begin{cases} p_j(n) & j = i \\ p_j(n) & j \neq i \end{cases}
\]

(2)

where \( \alpha \) is the learning parameter. Based on reward and penalty, vehicles form various coalitions to access the services from the cloud. The vehicles can join and leave these coalitions as discussed in Section 5.1.

4.1.2. Communication and computation layer

The collected data from the sensors is sent to vehicles using very short range communication techniques such as Bluetooth, Zigbee, Passive Radio Frequency Identifier (RFID), UWB (Ultra WideBand) and 60 GHz Millimeter Wave [18]. There are various types of communication used by the vehicles in the proposed system. Table 1 shows the comparison of existing techniques and standards which are used in different communications used in interconnected vehicles in VANETs [4]. Road Side Units (RSUs) are placed along the roads which form RSU clouds among one another. The vehicles that lie in the range of particular RSU cloud forms a coalition. These vehicles can form a Peer-to-Peer (P2P) or Vehicle-to-Vehicle (V2V) cloud to exchange the information regarding road condition, accident or congestion on the route, nearby places, and alternative route. For V2V interaction, short range communication techniques have been used such as, Dedicated Short-Range Communication/Wireless Access in Vehicular Environment (DSRC/WAVE) and Dynamic Spectrum Access (DSA). For interaction between Vehicle-to-RSU Infrastructure, DSA and DSRC/WAVE techniques are used. For communication of RSU clouds with central cloud, long range communication techniques like Wi-Fi, Worldwide Interoperability for Microwave Access (WiMax), Long term Evolution (LTE) (frequency bands range from 1.4 MHz to 20 MHz and provides data rate of 75 Mbps for uploading and 300 Mbps for downloading) and its variant LTE Advanced (maximum frequency of 100 MHz and provides peak data rate upto 3 Gbps for downloading and 1.5 Gbps for uploading), are used.

In this system, different RSUs combine together to form RSU clouds, where hardware, computational and storage resources are shared for processing of data. An intelligent Hypervisor is used for monitoring the state of system and for scheduling VM which is used to process the data sent by the vehicle. RSU clouds provide various functionalities to vehicles such as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), and STorage-as-a-Service (STaaS). IaaS and PaaS are used to provide proprietary services to the users, but in case of healthcare services mostly SaaS and STaaS are used to provide an application interface and store data onto the RSU cloud respectively. Coalition-as-a-Service (CaaS) can

<table>
<thead>
<tr>
<th>Communication</th>
<th>Alternatives</th>
<th>Protocols used</th>
<th>Frequency bands</th>
<th>Data rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicle-to-sensor (V2S)</td>
<td>Bluetooth</td>
<td>IEEE 802.15.1</td>
<td>2.4 GHz</td>
<td>Upto 3 Mbps</td>
</tr>
<tr>
<td></td>
<td>Zigbee</td>
<td>IEEE802.15.4</td>
<td>868 MHz, 915 MHz</td>
<td>250 Kbps at</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.4 GHz</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td></td>
<td>Passive RFID</td>
<td>IEEE 802.15.4f</td>
<td>915 MHz</td>
<td>&lt;4 Mbps</td>
</tr>
<tr>
<td></td>
<td>UWB</td>
<td>IEEE 802.15.4a</td>
<td>3.1-10.6 GHz</td>
<td>53.3-480 Mbps</td>
</tr>
<tr>
<td></td>
<td>60 GHz Millimeter</td>
<td>IEEE 802.15.3c</td>
<td>57-64 GHz</td>
<td>&gt;1 Gbps</td>
</tr>
<tr>
<td></td>
<td>Wave</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vehicle-to-vehicle (V2V)</td>
<td>DSRC/WAVE</td>
<td>IEEE 802.11p</td>
<td>5.850-5.925 GHz</td>
<td>3-27 Mbps</td>
</tr>
<tr>
<td></td>
<td>DSA</td>
<td>IEEE 802.11af</td>
<td>476-494 MHz</td>
<td>1 Mbps</td>
</tr>
<tr>
<td></td>
<td>WAVE</td>
<td>IEEE 802.11p</td>
<td>5.850-5.925 GHz</td>
<td>3-27 Mbps</td>
</tr>
<tr>
<td>Vehicle-to-road infrastructure (V2R)</td>
<td>Wi-Fi</td>
<td>IEEE 802.11 2.4-5 GHz</td>
<td>1.54 Mbps</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WIMAX</td>
<td>IEEE 802.16</td>
<td>1.25-20 MHz</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Existing communication technologies used in VANETs.
also be provided to vehicles to form coalition amongst themselves for accessing shared services.

4.2. Working methodology

Data is collected from the hospitals and is stored at a central cloud where large number of computational and storage resources are available. The advantage of doing this is that any doctor/hospital can access any patient’s data from anywhere to prescribe diagnosis. Also, the solution or decision support system, that is being used to generate the results, is deployed at the central cloud. Solutions are also deployed at RSU clouds, so that faster response to requesting entities can be provided. To fulfill this requirement, some of the data for chronic diseases is stored at RSU clouds, so that whenever a vehicle requests for service, RSU clouds would be able to differentiate whether the normal service is required or emergency services are required. The data coming from vehicles is stored at RSU clouds and after some time, it is copied to the centralized cloud where it is permanently stored for future references.

The flow of information in this model is as follows. Vehicles which want services form coalitions and request the nearby RSU Cloud which checks the payoff of the vehicle (or assigns a payoff value to the vehicles if not previously assigned), and based on the payoff value, it provides the services and resources to the vehicles. Higher payoff value indicates that the patient is chronic and requires emergency services, i.e., dedicated bandwidth is required for faster response. Lower payoff value indicates that normal services are required and hence resources are allocated accordingly. Payoff calculation algorithm is discussed in detail in Section 5. If RSU cloud is able to generate the results based on the received data, then, it passes the results back to vehicles, otherwise it sends the data to the central cloud where more sophisticated solutions are deployed. The priority to vehicles is assigned based on their payoff values. So, vehicles with more priority are served faster than normal vehicles. A VM Scheduling algorithm for scheduling VMs at the cloud level is required to handle the requests coming from the vehicles, so that, the resources are properly utilized. The designed algorithm is explained in the coming Sections. When resources on RSU cloud are not enough to provide services to the vehicles, it borrows resources either from other RSU clouds or from the central cloud. If vehicle goes out of range of one RSU cloud after requesting the service then, response to that vehicle is sent via another RSU cloud using MIPv6 until the handoff between vehicle and new RSU cloud is completed. Table 2 presents various symbols and their meaning used in the paper.

5. The proposed solution

The proposed solution is based on the concept of Bayesian coalition game and PN-based modeling in which all the moves and actions taken by the players are represented as the state space representation using PN and all the actions are executed using conditions probability among the players of the game. Each player in the game is assigned a unique payoff function (PF) based upon which it executes its actions by watching the strategies and PF of the other players in the game. Players are having the flexibility to

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF</td>
<td>Payoff function</td>
</tr>
<tr>
<td>$N$</td>
<td>Denotes the set of players consisting of $N$ number of mobile nodes, $N \in [1, N]$</td>
</tr>
<tr>
<td>$u$</td>
<td>Denotes the set of payoff utilities of mobile nodes at an instant of time, $u = {u_1, ..., u_n}$</td>
</tr>
<tr>
<td>$U$</td>
<td>Denotes the payoff utilities of coalition structures at an instant of time, $U = {U_{S_1}, ..., U_{S_k}}$</td>
</tr>
<tr>
<td>$S$</td>
<td>Denotes the set of coalitions at an instant of time</td>
</tr>
<tr>
<td>$P_I$</td>
<td>Denotes the service priority index of the mobile entity, $I$, such that $P_I \in [0, 1]$. This is variable value and could vary from low, medium and high priority</td>
</tr>
<tr>
<td>$B_{CI}$</td>
<td>Denotes the bandwidth capacity of the mobile entity $I$</td>
</tr>
<tr>
<td>$R_{CI}$</td>
<td>Denotes the resource capacity of the mobile entity $I$</td>
</tr>
<tr>
<td>$B_{R,l}$</td>
<td>Bandwidth required by the mobile node $l$</td>
</tr>
<tr>
<td>$R_{A}$</td>
<td>Bandwidth available with the RSU cloud</td>
</tr>
<tr>
<td>$R_{RA}$</td>
<td>VM Resources required by the mobile node $l$</td>
</tr>
<tr>
<td>$S_l$</td>
<td>Total VM resources available with RSU cloud</td>
</tr>
<tr>
<td>$P_R$</td>
<td>Place depicting the incoming request of mobile nodes for coalition formation</td>
</tr>
<tr>
<td>$P_P$</td>
<td>Place depicting that request are being processed to calculate the utility parameter of the respective node</td>
</tr>
<tr>
<td>$P_U$</td>
<td>Place depicting the estimation of cumulative utility function of the coalition</td>
</tr>
<tr>
<td>$P_C$</td>
<td>Place depicting the deciding utility comparison function</td>
</tr>
<tr>
<td>$P_G$</td>
<td>Place depicting the request generation phase</td>
</tr>
<tr>
<td>$P_V$</td>
<td>Place depicting respective RSU cloud with $n$ VMs</td>
</tr>
<tr>
<td>$T_A$</td>
<td>Transition depicting the arrival of incoming request of mobile nodes for coalition formation</td>
</tr>
<tr>
<td>$T_P$</td>
<td>Transition depicting the processing of requests for estimation of utility parameter of the respective node</td>
</tr>
<tr>
<td>$T_U$</td>
<td>Transition depicting that the estimation of cumulative utility function of the coalition</td>
</tr>
<tr>
<td>$T_C$</td>
<td>Transition depicting utility comparison function</td>
</tr>
<tr>
<td>$T_G$</td>
<td>Transition depicting the request generation phase</td>
</tr>
<tr>
<td>$T_V$</td>
<td>Transaction depicting end of the service request</td>
</tr>
<tr>
<td>$\lambda_A$</td>
<td>Arrival rate of the requests of mobile nodes for coalition formation</td>
</tr>
<tr>
<td>$\lambda_P$</td>
<td>Processing rate of the respective RSU cloud</td>
</tr>
<tr>
<td>$l_i$</td>
<td>Load parameter of node $i$</td>
</tr>
<tr>
<td>$id$</td>
<td>Index information of source mobile node that requires the services</td>
</tr>
<tr>
<td>$util$</td>
<td>Presents the current VM resource utilization by the source mobile node</td>
</tr>
<tr>
<td>$U_i$</td>
<td>Utility function of VM for current request</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Cost Function of node $i$</td>
</tr>
</tbody>
</table>
move from one coalition to another with an intention to increase their PF. Various steps used in the proposed solution are described as follows.

### 5.1. Coalition formation and splitting

Coalition formation and splitting amongst the mobile nodes are described in Algorithms 1 and 2 respectively. PF for a mobile node which needs services from the cloud is defined in Eq. (3). PF increases for the emergency services and it remains same for normal set of services required by mobile node. The Service Priority Index is used to identify the critical nodes. It categorizes the service requirement into emergency, normal, and low based on the data received at clouds which was generated by the sensors. The concept of coalition formation and splitting follows the Bayesian coalition game concept defined as follows.

Bayesian Game $g$ is defined by the following quadruple:

$$G = (N, u, U, S)$$

where,

- $N$ denotes the set of players consisting of $N$ number of mobile nodes, $N \in [1, N]$.
- $u$ denotes the set of payoff utilities of mobile nodes at an instant of time, $u = \{u_1, ..., u_N\}$.
- $U$ denotes the set of payoff utilities of coalition structures at an instant of time, $U = \{U_{S_1}, ..., U_{S_n}\}$.
- $S$ denotes the set of coalitions at an instant of time, $S = \{S_0, ..., S_L\}$.

Based upon the above parameters, payoff function $u_i$ for each player in the game is defined as follows:

$$u_i = p_T \cdot BC_T \cdot RC_T$$  \hspace{1cm} (3)

where,

- $p_T$ denotes the service priority index of the mobile entity, $l$, such that $p_T \in [0, 1]$. This is a variable value and could vary from low, medium, and high priority.
- $BC_T$ denotes the bandwidth capacity of the mobile entity $l$.
- $RC_T$ denotes the resource capacity of the mobile entity $l$.

**Algorithm 1.** Coalition Formation [10].

Input: $G = \{u, U, S\}$  
Output: $S$

1: Initialize $t = 0$.
2: A new mobile node $i$ requests to join $S_j$, such that $S_j \in S$.
3: Corresponding RSU cloud $j$ computes $i$’s payoff function $u_i$ at time $t$ as given in Eqs. (1), (2).
4: $j$ also computes new payoff function of $S_j$ after joining node $i$ at $t$ as $U_{S_j} = U_{S_j} + u_i$.
5: if ($U_{S_j} > U_{S_j}$) then
6: $i$ joins $S_j$
7: else
8: Do not join coalition and retry after some time
9: end if

**Algorithm 2.** Coalition Splitting [10].

Input: $G = \{u, U, S\}$  
Output: $S$

1: Initialize $t = 0$.
2: Mobile node $i$ is selected from the set $N$ which wants to leave its coalition $S_L$ to join $S_J$, such that $S_L, S_J \in S$ and $L \neq J$.
3: Corresponding RSU cloud $j$ gets $i$’s payoff function $u_i$ at time $t$ as given in Eqs. (1), (2).
4: $j$ computes new payoff function of $S_j$ at $t$ after adding node $i$ as, $U_{S_j} = U_{S_j} + u_i$.
5: IF ($U_{S_j} > U_{S_j}$) then
6: $i$ leaves $S_L$ and joins $S_J$
7: ELSE
8: Stays in current coalition structure
9: END IF

Table 2 (continued)

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_i$</td>
<td>Revenue Function of node $i$</td>
</tr>
<tr>
<td>$LT$</td>
<td>Lower Threshold</td>
</tr>
<tr>
<td>$UT$</td>
<td>Upper Threshold</td>
</tr>
</tbody>
</table>

Table 2 (continued)

$BC_T$ varies for the mobile node as different services have different bandwidth requirements. Emergency services may need to send multimedia data of patients like the video of patient, thus, require a high bandwidth. Normal services need bandwidth just for transmitting the data to the cloud, thus, the bandwidth requirement is less. It depends upon how much bandwidth is available with the RSU cloud. If bandwidth requirement for a mobile node is not available on RSU cloud, it can take these from the centralized cloud or other RSU clouds. It can also alter bandwidth of other mobile nodes in its coalition depending on the service priority index. $RC_T$ indicates resources needed by the mobile node to receive the requested service. VMs are present in each RSU cloud which handles requests coming from different mobile nodes. Emergency service might need more than one VM to get seamless service. The algorithm for coalition formation is inspired and motivated by the work presented by authors in [10], and is described as follows.
it adds node $i$, otherwise node $i$ is not added to current coalition. Algorithm 2 explains the process of a mobile node leaving current coalition and joining the other. Similar to Algorithm 1, RSU cloud gets the PF value of the mobile node. It computes the new PF for the cloud if mobile node joins the coalition, if new PF value is greater than the older one, it adds the mobile node to its coalition, otherwise coalition structure remains same. For the mobile nodes which goes out of the current RSU cloud and are not added to another coalition, services to these mobile nodes will be provided through MIPv6.

5.1.1. An example
Initially, when the requests will go to RSU cloud (where a small Decision support System and VMs are available), values of $\mathcal{P}_T$, $\mathcal{B}_T$ and $\mathcal{R}_T$ are computed at RSU cloud and equal values of $\mathcal{P}_I$, $\mathcal{B}_I$ and $\mathcal{R}_I$ are assigned to every mobile node considering that every node require normal service. After providing the services, RSU cloud updates the value of $\mathcal{P}_I$ based on the results of first service (if patient is identified as critical, then value of $\mathcal{P}_I$ is increased). From next time, $\mathcal{B}_I$ and $\mathcal{R}_I$ are computed based on the $\mathcal{P}_I$ value of the mobile node.

For example, suppose the data rate available with RSU cloud is 5 Mbps and number of VMs available are 50. Initial value of $\mathcal{P}_I$ is fixed as 0.2 and for normal service, value of $\mathcal{B}_I$ is fixed as 50 Kbps and value of $\mathcal{R}_I$ is fixed as 1. For sensitive cases (say $\mathcal{P}_I \geq 0.5$), value of $\mathcal{B}_I$ is fixed as 500 Kbps and value of $\mathcal{R}_I$ is fixed as 3. For extremely critical services (say $\mathcal{P}_I \geq 0.8$), value of $\mathcal{B}_I$ is fixed as 1 Mbps and value of $\mathcal{R}_I$ is fixed as 5. For other values of $\mathcal{P}_I$, values of $\mathcal{B}_I$ and $\mathcal{R}_I$ are fixed accordingly. Now, suppose a device $i$ sent the request to access the services and no value of $\mathcal{P}_I$ was assigned to it. So, now $\mathcal{P}_I$ is assigned as 0.2 and $\mathcal{B}_I$ is assigned as 50 Kbps and $\mathcal{R}_I$ is assigned as 1. After providing the services to node $i$, RSU learned that the data received from node $i$ was of a patient in critical situation, then, value of $\mathcal{P}_I$ is changed accordingly. Suppose the value of $\mathcal{P}_I$ of node $i$ is now changed to 0.6, now, next time when node $i$ will request for the service, $\mathcal{B}_I$ will be 500 Kbps and $\mathcal{R}_I$ will be 3.

Now, if the VMs or the bandwidth available with RSU cloud is utilized above Upper Threshold (UT), then, RSU cloud will either request centralized cloud or the other RSU cloud for VM resources as discussed in Section 5.3. If still the requirement of the new mobile nodes is not met, then, these nodes have to wait in a queue for accessing the cloud. Whenever requirement of current mobile node is met and it free RSUs resources, a node is picked up from the queue and services are provided to it.

5.1.2. SRN model for coalition formation and splitting
SRNs are variants of PNs which have been extensively used by the researchers to address various systems and issues [7-9]. In this paper, we have used a SRN model to represent coalition formation and splitting using LA-based component to access the healthcare services. The major objective of the current model is to represent the coalition formation process with respect to the availability of resources (VMs) on RSU cloud to cater the incoming requests from vehicles. Consider the following input parameters for the current model as $\lambda_A$ denotes the arrival rate of the requests from mobile vehicles that need to leave coalition $S_i$ and join coalition $S_j$, $n$ represents the number of available virtual machines in the respective RSU cloud under reference, $\lambda_p$ represents the processing rate of the respective RSU cloud. The SRN model is described as follows:

$$\text{SRN} = \{P, T, A, M_0\}$$

where,

- $P = \{P_R, P_F, P_U, P_G, P_V\}$
- $T = \{T_A, T_F, T_U, T_C, T_G, T_V\}$
- $A$ Matrix representing the set of all input and output arcs.
- $M_0 = \{0, N, 0, 0, 0, 0\}$

As depicted in the Fig. 4, transition $T_A$ depicts the arrival of requests from mobile nodes to join a new coalition $S_i$. This transition upon firing, places a token in place $P_R$ depicting the requests ready to be serviced by the RSU cloud. The cloud contains $n$ number of VMs machines. For the transition $T_F$ to be fired, at least one token each should be presented on $P_F$ and $P_R$ respectively. Once $T_F$ is fired, the related request is processed to estimate the payoff utility function of the related node and finally depositing a token on $P_V$. After this, the availability of a token at $P_R$, further triggers the transition $T_U$ that computes the cumulative utility function of the entire coalition $S_j$. Thus, a token is deposited at $P_U$. After this, transition $T_C$ is fired. This transition does the comparison between $U_{S_j}$ and $U_{S_i}$ and finally deposits a token at $P_C$. Following this activity, a message needs to be generated by the RSU cloud whether the mobile node is eligible to join coalition $S_i$ or not. This task is initiated by $T_G$ that finally deposits a token at $P_G$.

With this, the coalition formation task comes to an end that requires the resources to be released. Therefore, this task is performed by transition $T_V$ that finally deposits the token back at $P_V$.

5.2. VM-scheduling algorithm
VM scheduling is an important task which is performed by the intelligent hypervisor in RSU cloud so as to map different requests onto various VMs. The number of VMs created by the cloud varies with the network load. The cloud resources are distributed between various VMs. Depending upon the type of service the mobile node wants (emergency or normal), VMs are allocated to the mobile node by the cloud. It is possible to assign more than one VMs to one mobile node (which is considered as critical by the cloud based on received data) that requires more network resources in order to get the service quickly.

Various proposals have been used in the literature using different techniques to allocate VMs to the devices as discussed above. The design of VM scheduling algorithm in this paper is inspired from [28]. The difference between algorithm presented in [28] and our algorithm is that our scheme used a SRN-based modeling for state and actions representation by the various players in the game. The proposed learning algorithms are based upon the individual PF assigned to each player in the game.
We assume that the VMs within same RSU cloud can be shared. If the VMs in RSU cloud are already overloaded, then, VMs of different RSU clouds could be shared or central cloud can lend the resources.

5.2.1. Load estimation
The load information on each VM needs to be collected first in order to make VM scheduling decision. A load parameter \( l_i \) is used to represent the load information for each node

\[
l_i = (id, util)
\]

where, \( id \) is index information of source mobile node that requires the services and \( util \) presents the current VM resource utilization by the source mobile node. VM \( i \) will receive load information from all the mobile nodes present in its coalition. This information contains the ID of the source mobile node and VMs required to process the request that mobile node. After all the information is received, load estimation can be done which is further used for VM scheduling.

5.2.2. VM migration
As different types of applications need different resources to process the requests, the number of VMs required to process each request can change with time. Critical response may need more resources than normal response. In a scenario, where the mobile vehicles constantly request for services, the number of VMs which are required for processing the requests may not available in particular RSU cloud. In such a case, the requests are migrated to other VMs which are presented in the same coalition (intra) or to VMs of other RSU clouds (Inter). There are two possibilities that lead to VM migration, either the resources are ‘over-utilized’ or ‘under-utilized’. In both the cases, the requests need to be migrated to other VMs so that the load can be balanced in case of former and the load can be freed so as to save energy in case of latter.

Two thresholds are used to specify over and under utilization of resources: ‘upper-threshold’ and ‘lower-threshold’ respectively. The upper-threshold depicts that maximum resources are being used for a particular VM and remaining resources are reserved for the case of emergency and lower-threshold depicts that a particular VM is not utilized properly and the requests that processed by that VM can be migrated to other VMs to put that VM into sleep mode and save energy.

Algorithm 3. VM Scheduling for migrating requests of mobile nodes [28]

Input: \( I = \{U_i, R_i, C_i, UT, LT\} \)

Output: \( U_i \)

Assumptions: It is assumed that Revenue \( (R_i) \) and cost \( (C_i) \) function are calculated at RSU level depending on the resources that are being used.

1: Calculate utility of VM for the current request as

\[
U_i = (R_i - C_i) \times u_i,
\]

where \( u_i \) is calculated from Eq. (1).

2: if \( (U_i > UT \land U_i < LT) \) then

3: VM needs to migrate requests to other VMs.

4: Get VM utilization of other VMs that lies between their respective UT and LT.

5: Store the current utility, UT and LT for these VMs in a list.

6: Compute the \( U_i \) of the VMs for the current request.

7: Update the utility field of the list according to result gathered from previous step.

8: Discard the VMs whose \( (U_i > UT \lor U_i < LT) \) respectively.

9: Sort the list according to increasing values of \( U_i \).

10: Migrate the request onto first VM in the list.

11: end if

12: Migrate the next few requests to the same VM before refreshing itself or till the destination VM returns a STOP status to the current VM.

For a particular VM, upper and lower thresholds are calculated by RSU cloud itself depending upon the number of resources that are available in that RSU cloud. If for a VM, the utilization of resources is less than the lower-threshold or greater than the upper-threshold, then, the requests from that VM are migrated onto other VMs which is explained in Algorithm 3. Firstly, the utility function \( (U_i) \) is calculated for current request using the formula given in line 1 of algorithm. It uses a revenue function \( (R_i) \), cost function \( (C_i) \), and payoff function \( (u_i) \). \( R_i \) computes the revenue value that will be generated if the request is served, \( C_i \) computes the cost of resources that will be used to serve the request and \( u_i \) is payoff of the current request which is calculated in Eq. (1). If \( U_i \) lies between lower threshold \( (LT) \) and upper threshold \( (UT) \), then the request is served by VM, else, it gets the utilization of resources of other VMs along with their \( UT \) and \( LT \) values and stores it in a list. A new utility for all the VMs is calculated for the current request and its value is updated in the list. Only those VMs are retained in the list \( U_i \) lies between its respective \( UT \) and \( LT \).

The request is migrated onto the VM whose updated value is minimum from the list. The next few requests are also migrated onto the same VM until it returns a STOP status which indicates that processing of requests is not possible due to lack of resources. STOP status is sent if \( UT \) for a VM is reached. Resources above \( UT \) are reserved for emergency requests which may require more resources when served. If no suitable VM is found to migrate the requests, then, VM will require to wake up sleeping VMs and transfer requests onto these VMs. In case of a tie, RSU can randomly assigns one VM to process requests or it use first fit strategy. In first fit strategy, the VM which communicated its utility value to current VM earlier is chosen.

6. Performance evaluation

6.1. Simulation settings

The performance of the proposed scheme is evaluated using ns-2 with SUMO [42]. Starting with the initial configuration of the system, the algorithm is iterated several times to get the desired results. The results obtained are averaged over these finite number of iterations of the designed algorithm. Initially, all the actions taken by the players are executed randomly, so all actions are having equal probability initially. After finite number of iterations, the probability of execution of all the subsequent actions is increased or decreased depending upon the outcome of the previous actions, i.e., a reward or a
penalty. After finite number of moves, the solution converges to finite value. Parameters such as probability of execution of all the actions, % number of successful messages transmissions, and delay by varying the speed and arrival rate of jobs are used to test the effectiveness of the scheme. The results are obtained at 95% confidence interval by varying the speed of the vehicles, job arrival rate, and number of actions performed.

6.2. Results and discussion

6.2.1. Impact of vehicles speed on VM-scheduling parameters

Figs. 5–7 show the impact of speed of the vehicles on delay, throughput, and packet delivery ratio. The proposed scheme is compared with best fit, first fit, and random selection by varying the speed of the vehicles. As observed from Figs. 5–7, the proposed scheme that incurred less delay has higher throughput and packet delivery ratio as compared to the first fit, best fit and random selection algorithms. With an increase in the speed of the vehicles, it is difficult to maintain the route stability among the vehicles which results a higher delay, less throughput and less packet delivery ratio, but as the proposed scheme uses the SRN-based approach among the players for taking the actions, so there are less chances of failure of the transmitted messages which results an increase in the throughput and packet delivery ratio with a decrease in the delay as observed from the Figs. 5–7.

6.2.2. Impact of job arrival rate on VM-scheduling parameters

Figs. 8–10 show the impact of job arrival rate on delay, throughput, and packet delivery ratio. The proposed scheme is compared with best fit, first fit, and random selection by varying the velocity of the vehicles. With an increase in the job arrival rate, there is an extra load on the nearest access point to satisfy the request with in the defined threshold of time which may increase the delay incurred. But, as the proposed scheme has used an intelligent SRN-based approach with constant learning from the environment, so the incoming requests are satisfied from the nearest and less overloaded access points resulting a decrease in the delay incurred, increase in the overall throughput and packet delivery ratio as observed from the Figs. 8–10. None of the other existing schemes have an intelligent mechanism for satisfying the users requests in an intelligent manner, so these schemes have more delay, and less throughput and packet delivery ration as observed from Figs. 8–10.

6.2.3. Impact of number of actions on probability of successful transmissions

Figs. 11–13 show the impact of number of actions on probability of successful packet delivery, utility and successful transitions by varying the learning rates of the players in the game. As we increase the learning rates of the players, there are chances of an increase in the number of messages exchanged among the players of the game which may result an increase in the overhead generated. With an increase in the learning rate of the players in the game, there is an increase in the probability of successful delivery of the packets as observed in the Fig. 11. This is because at the higher learning rate, it is difficult to maintain the synchronization with respect to all the moves taken by the players in the game. Similarly, there is an increase in the utility of the players at lower learning rate as observed in Fig. 12. Moreover, there is an increase in the successful transmissions at lower learning rate as observed in Fig. 13. This is due to the fact as explained above that with an increase in the learning rate, more overhead may be generated with respect to the moves taken by the players in the game (Fig. 13).
7. Conclusions

Providing healthcare services on-the-fly is one of the biggest challenges in the dynamic environment such as vehicular networks as vehicles are having high velocity and varying density. In this paper, we propose a new multi-player cooperative coalition game theory to address this issue in which states of the players are represented as a stochastic reward nets (SRNs). In SRNs, moves of the players are...
conditioned by a learning rate, with respect to the moves of other players in the game. A new payoff function is designed for the players in the game such that coalition among the players is formed using the conditional probability of execution of all the actions taken by the players in the game. Also, an algorithm for VM-scheduling is proposed for execution of various moves taken by the players in case of overloading. The performance of the proposed scheme is evaluated using various performance metrics in comparison to other existing proposals. The results obtained prove the effectiveness of the proposed scheme in comparison to other schemes. In the future, we will explore various strategies for VM-scheduling by varying the load on the service providers.

Acknowledgments

This work was partially supported by Instituto de Telecomunicações, Next Generation Networks and Applications Group (NetGNA), Covilhã Delegation, by Government of Russian Federation, Grant 074-U01, and by National Funding from the FCT - Fundação para a Ciência e a Tecnologia through the UID/EEA/500008/2013 Project. We would like to thank all the anonymous reviewers for their constructive comments which improved the quality, presentation and content of the paper.

References

Providing healthcare services on-the-fly using multi-player cooperation game theory


[42] (www.isi.edu/nsnam/ns).