International Journal of Science and Research (IJSR)
ISSN (Online):
Index Copernicus Value (2016): 6.14 | Impact Factor (2015): 6.391

Rider: A Multi Layer Framework For Road Condition Estimation

Sona P[1], Divya M[2]

[1, 2]AWH Engineering College, Calicut University, Department of Computer science & Engineering, Kuttikkatoor, Kozhikode, India

Abstract: ***Rider: A multilayer framework for road condition estimation, is an agent based multi-layer framework to estimate road condition details and to support various applications for vehicular social networks (VSNs)formed by in-vehicle or mobile devices used by drivers, passengers, and pedestrians. The programming model of the framework incorporates features that support collaborations between mobile agents to provide communication services on behalf of owner applications, and service (or resident) agents to provide application services on mobile devices. Built on top of the mobile devices operating systems, the framework architecture consists of framework service layer, software agent layer and owner application layer. By using this application developed on the framework can autonomously and intelligently self-adapt to rapidly changing network connectivity and dynamic contexts of VSN users.***

Keywords: VANET, VSN, software agent, mobile agent, agent migration

1. Introduction

Rider, a multi-layer framework for road condition estimation that is able to utilize road condition information and provide a high level software platform for VSN applications. Rider hides the complexity of dealing with changing network connectivity and varying user services requirements in VSNs, by providing a programming model with extensibility support. Furthermore, Rider supports dynamic agent collaborations and service matching during runtime of mobile devices. Thus, more autonomous, intelligent and adaptive applications can be developed for the dynamically changing environments of VSNs. It can perform well under dynamically changing connectivity, and achieve good time efficiency with low computation and communication overhead.

2. Related works

Literature survey deals with the related techniques which contribute to the development of Rider system architecture. Here present ten most important papers for developing the problem definition.

Information interaction is a crucial part of modern transportation activities. The idea of Vehicle-to-Passenger communication (V2P)[1], which allows direct, instant, and exible communication between moving vehicles and roadside passengers. With pocket wireless devices, passengers can easily join VANETs as roadside nodes, and express their

travel demands, e.g., taking a free ride or calling a taxi via radio queries over VANETs. Once a matched vehicle is found through the disseminated queries, the driver can decide
whether to provide corresponding services, especially the carrying of passengers and goods.

Here investigate the main challenges in vehicle calling; establish a trip history model to predict vehicle movement, and develop typical query dissemination schemes to match the target vehicle in vehicular networks. With V2P over VANETs, vehicle transportation is capable of open and efficient P2P information interaction, and thus benefits from relevant efficiency improvement. Based on a realistic travel survey and simulation, and prove that vehicle calling is effective and efficient in casual carpooling and taxi calling. In V2P, both vehicles and passengers need necessary hardware equipment to support the communication. When it comes to vehicles, GPS and electric maps are widely deployed, as well as wireless devices.

Since there's no standard human-machine interface, assume that some signal lights, buzzers, or graphical or phonetic interfaces can inform the driver of vehicle calling, allowing the driver to answer yes or no by pushing buttons on the device. To individuals, previous wireless mobile devices, such as cell phones, i-Pods, PDAs, and laptops, are not designed for vehicular communication. So, here assume that passengers have pocket wireless devices as vehicle callers in order to perform this task, which can be regarded as a cheap device with an electric map and simple input/output.

In the field of ubiquitous computing, a class of applications called context-aware services attracted great interest especially since the emergence of wireless technologies and mobile devices. Context aware application [2] can dynamically capture a range of information from its environment and this information represents a context, the application adapts its execution according to this context. An important challenge in ubiquitous computing is dealing with context. Ontologies presents the most promising instrument for context modelling and managing due to their high and formal expressiveness and the possibilities for applying ontology reasoning techniques.

To build context aware services, here need to define mechanisms for the adaptation of their behaviour according to the current context situation. Such mechanisms will Favourite loosely coupling between the core services and its adaptations seen as transversal preoccupations. The adaptations are eventually conditioned by the existence of relevant situations to the current context. The Request

Notifier notifies, in a synchronous or asynchronous mode, the Service Identifier with the active situation and Id of Service in order to adapt this service to the given situation. The interpretation mechanism operated by Reconfigure Service, recover situations and weave the necessary adaptation aspects, following a set of adaptation rules, and user context in services to produce a semantic contextual service.

In the last few years telecommunications and internet have spread all over the world, in a pervasive way, connecting millions of devices, people, sensors and services without a planned strategy. In such scenario the discovery of services represent still an open challenging research field. To address that problem [3] proposes context aware semantic service discovery architecture designed to perform distributed Service Discovery in heterogeneous networks. This novel architecture is technology independent and compatible with most of the existent service discovery protocols; it inherits and extends the results of the last research groups in the field of context-aware service discovery based on the use of semantic languages.

There exist distributed scenarios [4] in which the need for dynamism, mobility, and adaptivity, has to be addressed with highly dynamical approaches. These scenarios present different challenges and difficulties: efficient access to heterogeneous and distributed data sources, dynamic load balancing, unstable connections and communication failures, etc. So, different approaches and middleware have appeared to tackle these challenges and help the developer of distributed applications. In particular, mobile agent technology can provide significant advantages for the development of applications in these contexts. In this paper, here emphasize the benefits that mobile. Agents can provide to distributed systems by illustrating them with real distributed Systems.

LOQOMOTION (Location dependent Queries On Moving ObjecTs In mObile Networks) is a [5] distributed location-dependent query processing system whose architecture is based on mobile agents. The system deploys a network of agents to perform the query processing over a distributed set of computers, called proxies, which manage information about objects moving within different geographic areas: each proxy is in charge of one proxy area and so the data management tasks are distributed. Moving objects (e.g., cars) are represented as circles in the figure, and the dashed ellipses represent the different proxy areas. Each proxy has a Data Management System or DMS (e.g., a Database Management System) that stores information about the moving objects within its proxy area and can answer queries about them.

In this scenario, assume for illustration purposes that the user submits a query that must retrieve the white objects located within a certain distance of each black object, that is, a query that retrieves the white objects within the circular relevant areas shown around the black objects. As the objects are continuously moving, the answer to the query must be updated continuously, with a certain refreshment frequency.

Currently there are no Internet access authentication protocols available that are lightweight, can be carried over arbitrary access networks, and are exible enough to be reused in all the likely future ubiquitous mobility access contexts.[6] article proposes the PANA/UMTS authentication protocol for heterogeneous network access as a step towards filling this gap. A security analysis of the PANA/UMTS protocol is also provided. This article aims primarily at contributing to the design of authentication protocols suitable for use in future heterogeneous Internet access environments supporting ubiquitous mobility

Agilla [7] is a mobile agent middleware that facilitates the rapid deployment of adaptive applications in wireless sensor networks (WSNs). Agilla allows users to create and inject special programs called mobile agents that coordinate through local tuple spaces, and migrate across the WSN performing application-specific tasks. This utility of code and state has the potential to transform a WSN into a shared, general-purpose computing platform capable of running several autonomous applications at a time, allowing us to harness its full potential.

Mobile agents are special processes that can autonomously migrate across nodes. Agilla provides two forms of migration, strong and weak, to support diverse application needs for self- adaptation. Strong migration transfers both the code and state, allowing the agent to resume execution at the destination. It is useful for performing computations that span multiple nodes. Weak migration only migrates the code. It exhibits less overhead since the state does not need to be transferred, but resets the agent. When an agent migrates, it can either clone or move. If an agent is cloned, a copy of it arrives and starts executing at the destination while the original one resumes on the original node. If an agent is moved, it will no longer exist on the original node after it arrives at the destination. An agent's life cycle begins when it is either injected into the network from a base station or cloned from another agent already in the network. Each agent executes autonomously performing application-specific tasks, and multiple agents may reside on the same node. When an agent completes its tasks, it dies freeing the computational resources it used.

3. Supporting Multilayer Framework

Rider is an agent based multi-layer framework to support the various applications of for vehicular social networks (VSNs) formed by in-vehicle or mobile devices used by drivers, passengers, and pedestrians. The programming model of the framework incorporates features that support collaborations between mobile agents to provide communication services on behalf of owner applications, and service (or resident) agents to provide application services on mobile devices. Built on top of the mobile devices operating systems, the framework architecture consists of framework service layer, software agent layer and owner application layer. By using this application developed on the framework can autonomously and intelligently self-adapt to rapidly changing network connectivity and dynamic contexts of VSN users. A practical implementation and experimental evaluations of Rider are presented to demonstrate its

reliability and efficiency in terms of computation and communication performance on popular mobile

3.1 System Architecture

Rider consist of following layers 1.Framework service layer 2. Resident agent layer 3. Mobile agent layer 4. Owner application layer

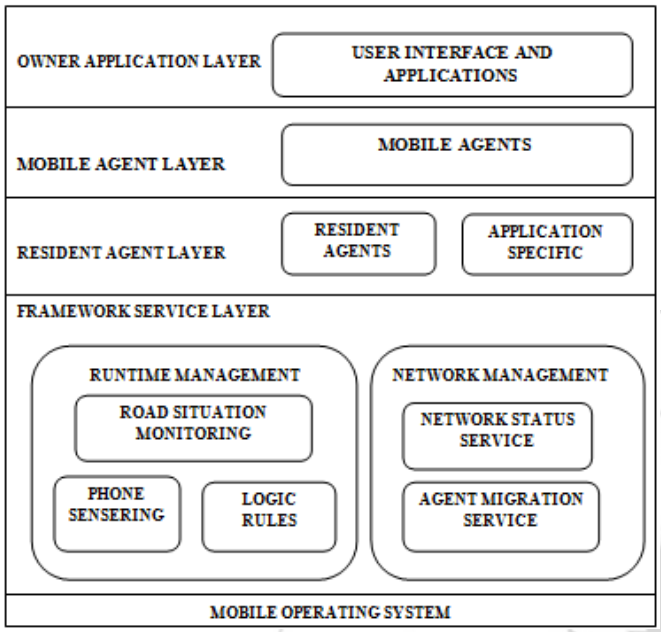

Figure 1: System Architecture

Framework service layer: This layer initializes the runtime environments of Rider, extracts the context information from the VSN, and provides the core functions and generic services as framework services to the upper layer. The framework services are only accessed by the resident agent layer but not by other layers. In order to meet the unique Requirements of VSN applications beyond basic services like time service and ID name service. It has two functions: Runtime management functions and network management Functions. The runtime management is responsible for the on time sensing of the traffic situations in the road. And provide latest information given to the other users in the VSN. Network management is responsible for the dynamic network connectivity in VSN.

Resident agent layer: The resident agents provide all application services of Rider on each node to visiting mobile agents. Resident agents can be deployed automatically on all the services that the resident agents provide. Application services provided by the resident agent layer can be built on the services provided by the framework service layer. Framework services only implement basic and generic functions and services that are required by most VSNs applications. Thus, the services provided by this layer contain two parts: demand to any mobile node that participates in the same VSN. Once resident agents are deployed to a mobile node, they will remain in it to provide various application services simultaneously as long as the node is a part of the VSN. Mobile agents can directly use all the services that the resident agents provide. Application services provided by the resident agent layer can be built on the services provided by the framework service layer. Framework services only implement basic and generic functions and services that are required by most VSNs applications. Thus, the services provided by this layer contain two parts: (i) framework services provided by the framework service layer and (ii) application specific services developed by application developers. Framework services could be reused by every VSN application, while application specific services could be reused by a specific type of VSN application on the mobile devices during run-time.

Mobile agent layer: The mobile agents run on top of resident agents and are used to execute different application services provided by resident agents. They do not contain any application services in their codes. All the services a mobile agent needs come from the resident agents, such as the services for specific applications, migration service, etc. A mobile agent only contains basic information, such as its migration mode, processing scheme for executing results, as well as computation and communication results. Also, mobile agents are used for transferring necessary resources or application services to some mobile devices when they do not contain such resources or services. Moreover, the data of task execution result gathered from a mobile agent could be shared among multiple VSN applications in one device.

Owner application layer: The applications are owners of mobile agents. An application provides the user interface to its users who use mobile devices in their vehicles. Through the user interface, the user can select the developed functions he/she prefers, and the owner application of Rider could automatically initiate a mobile agent to execute the services of Rider to accomplish the corresponding function in the VSN system, or release multiple mobile agents to accomplish different tasks simultaneously. Rider supports multiple mobile agents with multiple owner applications working at the same time. In addition, the owner applications can monitor the status of mobile agents they release with the help of resident agents distributed on each node.

3.2. Adaptation to Dynamic Network Connectivity

Since the network connectivity of the underlying VANET is dynamic and frequently changes, each mobile node can join or leave the network anytime and anywhere, which may cause VSN application failures. For example, without knowing the failures of targeting nodes when mobile agents execute tasks around VANETs, they may get lost in such networks and fail to bring the expected results back to the users, while the lack of sufficient services and resource in a new mobile node joining a VSN may also impact the correctness of the execution results of mobile agents for VSN applications. Thus, Rider provides self healing and self-configuring capabilities over dynamic network connections as key framework services to VSN applications. In Rider, for VSN applications to self-heal, it enables the resident agents deployed on every mobile node to continue monitoring the status of a VANET (e.g., which nodes have just become unavailable in a VSN), and provide the information to the mobile agents to help them to adapt to node or link failures. Also, for self-configuration, Rider provides a service that can

dynamically and automatically deploy the core functions and services to a new mobile node when it joins a VSN upon VANETs, and if a mobile device does not have the necessary resources or services, mobile agents can also automatically transfer the necessary resources or services between mobile nodes.

3.3 Adaptation to Users Dynamic Contexts

In a VSN system, because of the opportunism of user connections, the changing contexts of the users may also result in users dynamic contexts. However, traditionally, the descriptive information of the service requester is compared to that of the service provider, and their similarity is measured using traditional service matching by simple string matching methods. Such an approach cannot work well as it is not realistic to require service requesters and service providers to use exactly the same contexts (e.g., words about their destinations in a rideshare application) in open and dynamically changing environments of VSNs.

Here adopt a hierarchical architecture for Rider, so as to provide a light-weight but scalable framework and facilitate agent collaborations and resource reuse for multiple VSN applications on mobile devices. The architecture of Riders four layers from the bottom up: the runtime management layer, resident agent layer, mobile agent layer, and owner application layer.

3.4 Dynamic Network Environment

In Rider, consider two situations when mobile agents are executing the applications while the network environment of VSN changes: First situation: Mobile nodes become disconnected when mobile agents are executing applications in a VSN system. Second situation: New mobile nodes join a VSN system when mobile agents are executing applications. In Rider, develop a novel network status service in its framework service layer, which makes use of the network APIs, where every node can listen and determine how many nodes are currently available in the VANET. In addition, Here adopt a scheme that enables resident agents to share all the IDs of currently connected nodes, as well as available service lists of mobile devices. By using this service, the framework service layer can inform the upper layer about the real-time networking status of the VSN, such as the current list of node IDs and available services. At the same time, considering the diverse VSN scenarios and specific performance concerns of different VSN applications, Rider does not put any constraint on the application behaviours like specific migration schemes of mobile agents. Instead, to ease the developers efforts, Rider provides three generic migration strategies for application developers to develop mobile agents.

4. Steps of operation

Procedure (Driver Calling)

Finding nearest vehicle

1. Set passenger latitude and longitude
   latitude=LocationService.latitude
   longitude=LocationService.longitude
2. Take matching latitude and longitude from database table driver
3. If there is a match available then return its vehicle no, name and phone no from the registration and driver table

- Procedure-Road Disruption

Finding road disruption

1. Set lastupadate=-1
   shakethreshold=1400
   tempplace=null
2. Get services from sensor manager and telephony manager for sensor reading and device id
3. If (curtime-lastupdate)>100ms)
   difftime=curtime-lastupdate
   lastupdate=curtime
4. Find the value of 3D co-ordinates
   (Y, 4)>10 then top Shake
   If(X,4)>-10 then left shake
   If(x, 4)>10 then Right Shake
   If(Y,4)>-10 then Bottom shake
5. Check the axis shake
   X=values[sensormgr.DATA_X]
   Y= values [sensormgr.DATA_Y]
   Z= values[sensormgr.DATA_Z]
6. Calculate speed value
   speed= [(X+Y+Z-LastX-LastY- LastZ)/difftime]*1000
7. if (speed >shakethreshold)
   Search data in db to find the place where the disruption is entered. If not entered update database table ‘shake’ in the same location
8. Set LastX=X
   LastY=Y
   LastZ=Z
9. Repeat the operation

5. Impacts and Cost

Considering the limited resource of mobile devices (i.e., computing power, storage capacity, and battery energy), unstable networking connections of VANETs, and time efficiency requirements of most VSN application scenarios, the impacts and costs of mobile VSN applications on mobile devices are always a concern. Thus, it adopt weak adaptation as the primary option in Rider, so as to decrease the resource requirements on mobile devices and ensure that VSN applications developed on Rider have considerable time efficiency in finishing tasks. Also, in order to reduce networking overhead of VSN systems, Rider divides data into two types: shared and nonshared. The shared data contain only the basic context information (e.g., ID name and available services) of the local device, which will be shared with the VSN system by the resident agents. On the other hand, non-shared data contain the agent codes, application specific services, existing data in local mobile devices, and execution results of mobile agents, etc. In addition, when developers develop mobile agents, they can decide whether the mobile agents should store the processing results locally in the mobile devices, and what data mobile agents should carry over to the next devices to which they migrate.

6. Conclusion

Rider, an agent based multi-layer framework to road condition estimation that is able to utilize the information from the sensor and provide a high level software platform for VSN applications. Rider hides the complexity of dealing with changing network connectivity and varying user services requirements in VSNs, by providing a programming model with extensibility support. Rider also provides a rich set of framework services to support application development using Java with the standard API format.

7. Future Scope

Rider provides a modelling paradigm which can facilitate the creation and deployment of different VSNs applications. Furthermore, Rider supports dynamic agent collaborations and service matching during runtime of mobile devices. Thus, more autonomous, intelligent and adaptive applications can be developed for the dynamically changing environments of VSNs.

References

[1] N. Liu, M. Liu, J. Cao, G. Chen, and W. Lou, When Transportation Meets Communication: V2P over VANETs, In Proc. IEEE ICDCS, 2010, pp. 567-576.
[2] H. Chen, T. Finin, and A. Joshi, An ontology for context-aware pervasive computing environments, The Knowledge Engineering Review, vol. 18, no. 03, pp. 197207, 2003.
[3] K. Fujii, and T. Suda, Semantics-based context-aware dynamic service composition, ACM Trans. on Autonomous and Adaptive Systems, 2009, vol. 4, no. 2, Article 12.
[4] A.R. El-Sayed and J. Black, "Semantic-Based Context-Aware Service Discovery in Pervasive-Computing Environments". Carlos Bobed, Sergio Ilarri, and Eduardo Mena IIS Department, University of Zaragoza, Zaragoza, Spain,
[5] "Distributed Mobile Computing: Development of Distributed Applications Using Mobile Agents", in IEEE conference, July 2010.
[6] Paulo S. Pagliusi and Chris J. Mitchell Information Security Group Royal Holloway, University of London Egham, Surrey TW20 0EX, UK,"Heterogeneous Internet Access via PANA/UMTS", August 31, 2004
[7] C. L. Fok, G. C. Roman, and C. Lu, Agilla: A Mobile Agent Middleware for Sensor Networks, ACM Trans. on Autonomous and Adaptive Systems, 2009, vol. 4, no. 3, Article 16.

Author Profile

Sona P received the B Tech degree in Computer Science and Engineering from university of Calicut in 2009 and she is currently persuading her M-tech in Computer Science and Engineering from Calicut University.

Divya M received the B Tech and M Tech degrees in Computer Science and Engineering from Calicut University in 2008 and 2013, respectively. She is working with AWH Engineering College since December 2010.