Design, implementation and performance analysis of a vehicular cloud using Hadoop over wireless links

by Athanasios Oikonomou, Supervisor: Dimitrios Katsaros, University of Thessaly

***Abstract*—In the 20th century vehicular clouds has become a significant research area due to its specific features such as efficient traffic management, transport safety improvement, traffic congestion relieve and infotainment. Vehicles carry more communication systems and have more computing and storage power than ever. A single vehicle has limited computation and storage resources, which may result in low data processing capability. The idea of all the vehicles connected together creating a big cloud so that can collect, storage and analyze a large amount of traffic related data is a big challenge. In this cloud vehicles must share computation resources, storage resources and bandwidth resources. Several technologies have been deployed to maintain and promote Intelligent Transportation Systems (ITS). This paper presents the design and implementation of a hadoop based vehicular cloud. The proposed architecture is a layer above hadoop which combines all the advantages of hadoop framework with the advantages of wireless links between vehicles.**

***Index Terms*—cloud computing; hadoop; mapreduce; mobile cloud; vehicle; wireless link;**

I. Introduction

THE last years a lot of researches have taken place on how to make the vehicles on our roads smarter and the driving experience safer and more enjoyable. By supporting traffic related data gathering and processing, vehicular networks are able to notably improve transport safety, relieve traffic congestion, reduce air pollution, and enhance driving comfortability [1]. The huge fleet of energy-sufficient vehicles that crisscross our roadways, airways, and waterways, most of them with a permanent Internet presence, featuring substantial on-board computational, storage and sensing capabilities can be thought of as a huge farm of computers on the move.

Many vehicles are being equipped with components that someone would classify them as 'intelligent vehicles'. Such components include sensors and actuators with intra-vehicle communication, and electronic control units for data processing and operation control. A typical car or truck today is likely to contain at least some of the following devices: an on-board computer, a GPS device, a radio transceiver, a short-range rear collision radar device, a camera, supplemented in high-end models with a variety of sophisticated sensing devices. According to the automotive sensors market growth, the average number of sensors per vehicle has reached 70 in 2013 [2].

Also vehicles in near future will be equipped with wireless communication modules for supporting three types of communication: between vehicles, known as vehicle-to-vehicle (V2V) communication, between vehicles and infrastructure (V2I), or between vehicles and any neighboring object (V2X). A new type of networks, referred as Vehicular Ad-hoc Networks (VANET) employ a combination of V2V and V2I communications and have been proposed to give drivers advance notification of traffic events. A VANET is a set of moving vehicles in a wireless network that apply the Information Communication Technology (ICT) to provide state-of-the-art services of traffic management and transport. VANET has received significant consideration because of the prospect of enabling novel and attractive solutions in areas such as vehicle and road safety, traffic efficiency and Intelligent Transportation Systems (ITS). Dedicated Short Range Communications (DSRC) [3] is specifically designed for Vehicle-to-Vehicle (V2V) and Vehicle-to-Roadside (V2R) communications. The IEEE 802.11p, called Wireless Access in Vehicular Environments (WAVE) [4], is currently a popular standard for DSRC. Besides, the Long-Term Evolution (LTE), LTE-Advanced and Cognitive Radio (CR) [5] [6] are all fairly competitive technologies for vehicular networking [7] [8].

In this paper we will design and implement our cloud manager which is a layer above hadoop framework. Hadoop was designed to analyze data using nodes connected by Ethernet cables locally. So this layer will make hadoop to work over wireless links and will give solutions to several fundamental problems in vehicle networks.

The rest of the paper is organized as follows. We describe the vehicular basic idea in Section II, and present some application scenarios in Section III. The way hadoop frameworks analyzes data can be seen in Section IV. In Section V, we present the vehicular network architecture and in section VI our system architecture and how we used hadoop's basic principles. Also, how to adjust them to our manager. Finally, we present performance statistics in Section VII, and conclude this paper at the end.

Dimitrios Katsaros and Athanasios Oikonomou are with Department of Electrical & Computer Engineering, University of Thessaly, Volos. Email: dkatsar@inf.uth.gr, oikonathan@inf.uth.gr

II. VEHICULAR CLOUD

Today, the term "cloud computing" is used to describe the concept off running remote service over the network. The backend are usually located in the data center and services can be provide to massive amount of users concurrently in a distributed manner. The idea behind cloud computing is that computing, storage, and bandwidth are packaged as a service and can be utilized over a network, which is the Internet. The vehicular cloud differs from today's Internet cloud in the nature of its hardware and sensor platform. The vehicle cloud builds its platform opportunistically in a centralized or distributed control, from spare and idle resources made available by participating vehicles. Another major difference is the mobility nature of vehicular cloud. Vehicle cloud enables application to run on a particular location at a particular time, which is not possible for traditional cloud service. In fact, the majority of services provided by the vehicle cloud are expected to be location aware applications.

Vehicular Clouds (VCs) utilize the concept of Cloud Computing and VANETs together. There are two types of VCs. Infrastructure-based VC and Autonomous VC (AVC). In the Infrastructure based VC, the driver accesses roadside infrastructure for services. In AVC, the vehicles are organized in VC to support emergencies and ad hoc events. The VC services can be categorized into three levels, application, platform and infrastructure.

Infrastructure as a Service (IaaS): Several types of virtualization occur in this layer. Among the other resources, computing, network, hardware and storage are also included. In the bottom layer of the framework, infrastructure devices and hardware are virtualized and provided as a service to users to install the operating system (OS) and operate software applications. Therefore, this layer is named Infrastructure as a Service (IaaS). Amazon Web Services (AWS) is a very good example of this category where Amazon provides its customers computing resources through its Elastic Compute Cloud (EC2) service and storage service through both Simple Storage Service (S3) and Elastic Book Store (EBS).

Platform as a Service (PaaS): In PaaS, Mobile operating systems such as Android, iPhone, Symbian and other OS, as well as database management and IMS are included in this section. This layer contains the environment for distributing storage, parallel programming design, the management system for organizing distributed file systems and other system management tools for cloud computing. Program developers are the primary clients of this platform layer. Google AppEngine and Microsoft Azure are good example of that category.

Software as a Service (SaaS): Analytical, interactive, transaction and browsing facilities are included in the Application layer. SaaS delivers several simple software programs and applications as well as customer interfaces for the end users. Thus, in the application layer, this type of services is called Software as a Service (SaaS). By using the client software or browser, the user can connect services from providers via the internet and pay fees according to their consumed services, such as in a pay as you go model. IBM is a good example of this category.

Olariu et al. [9] argued that it is only a matter of time before the huge vehicular fleets on our roadways, streets and parking lots will be recognized as an abundant and under-utilized computational resource that can be tapped into for the purpose of providing third-party or community services. With this in mind, we have coined the term AVC to refer to: a group of largely autonomous vehicles whose corporate computing, sensing, communication and physical resources can be coordinated and dynamically allocated to authorized users.

Our cars spend significant time on the road and may face dynamically fluctuating locations. In this case, the vehicles will help local consultants resolve traffic incidents in a timely fashion which is not possible with the municipal traffic management centers alone due to the lack of adequate computational resources. We expect that, the vehicles are capable of solving problems in many situations that may require an indefinite time for a centralized system.

The cloud infrastructure consists of two parties: cloud storage and cloud computation. The data gathered by the inside-vehicle layer will be stored in the geographic information system (GIS), a road traffic control device or a storage system based on the type of applications. The computation part is used to calculate the computational tasks which provide faster performance, for example, the health recognition sensors send data to driver behavior database in cloud storage.

Despite of the well-developed information technologies, there is a significant challenge that hinders the rapid development of vehicular networks. Vehicles are normally constrained by resources, including computation, storage, and radio spectrum bandwidth. Due to the requirements of small-size and low-cost hardware systems, a single vehicle has limited computation and storage resources, which may result in low data processing capability. On the other hand, many emerging applications demand complex computation and large storage, including vehicle multimedia entertainment, vehicular social networking, and location based services. It becomes increasingly difficult for an individual vehicle to efficiently support these applications. A very promising solution is to share the computation and storage resources among all vehicles or physically nearby vehicles. Another important characteristic of AVCs is the ability to offer a seamless integration and decentralized management of cyber-physical resources; an AVC can dynamically adapt its managed vehicular resources allocated to applications according to the applications' changing requirements and environmental and systems conditions. Routing in vehicular networks has been considered as a challenging task, with consideration of various factors, including vehicles mobility patterns, the ability of vehicles to carry messages to different locations, and the availability of Internet connections at roadside infrastructure. Moreover, variant communication channels have different levels of reliability in a location, which leads to various end-to-end delays and communication costs. A good routing strategy should take all these for an optimized and dynamic decision.

One of the main characteristics of VC is the mobility of the nodes which directly affects on the available computational capabilities and storage resources, for example, the number of parked vehicles in the parking is not constant. Therefore, to provide fluctuating application requirements and resource accessibility on the move, the necessary related protocol architecture and VC networking must be developed.

The existing layered network architecture, for example TCP-IP stack, is not adequate to support ongoing evolving technologies and applications. It needs to use the service-oriented and component based network architecture [10] with sufficient learning opportunities and monitoring facilities in order to cope with reusable and extensible applications and resources, which require to be largely deployed as common services available in VC environments.

The fundamental building blocks and structures that compose VC should be engineered and designed to face the structural stress of the unstable working situation. Olariu et al. [11] was the first to propose a robust dynamic architecture for VC based on Eucalyptus cloud system [12] and virtualization approach to aggregate the computational and storage resources. Greater emphasis and more researches are necessary for the migration of virtual machines among cars and efficient vehicle virtualization.

III. Application scenarios

The main goal of this section is to illustrate the power of the AVC concept. VCs offer cost-efficient way of services. Smart combinations of storage, price, and communication abilities of VC can be chosen. We touch upon several important scenarios illustrating various aspects of AVCs that are extremely important and that, under present-day technology are unlikely to see a satisfactory resolution.

A. Scenario I: augmenting the capabilities of small businesses.

Consider a small business employing about 250 people and specializing in offering IT support and services. It is not hard to imagine that, even if we allow for car-pooling, there will be up to 150 vehicles parked in the company's parking lot. Day in and day out, the computational resources in those vehicles are sitting idle. We envision harvesting the corporate computational and storage resources in the vehicles sitting in the parking lot for the purpose of creating a computer cluster and a huge distributed data storage facility that, with proposer security safeguards in place, will turn out to be an important asset that the company cannot afford to waste.

B. Scenario II: dynamic management of parking facilities.

Anyone who has attempted to find a convenient parking spot in the downtown area of a big city or close to a university campus where the need for parking by far outstrips the supply would certainly be interested to enlist the help of an automated parking management facility. The problem of managing parking availability is a ubiquitous and a pervasive one, and several solutions were reported recently. However, most of the existing solutions rely on a centralized solution where reports from individual parking garages and parking meters are aggregated at a central (city-wide) location and then disseminated to the public. The difficulty is with the real-time management of parking availability since the information that reaches the public is often stale and outdated.

This, in turn, may worsen the situations especially when a large number of drivers are trying to park, say, to attend a down-town event. We envision that by real-time pooling the information about the availability of parking at various locations inside the city, an AVC consisting of the vehicles that happen to be in a certain neighborhood will be able to maintain realtime information about the availability of parking and direct the drivers to the most promising location where parking is (still) available.

C. Scenario III: autonomous mitigation of recurring congestion.

In face of traffic congestion some drivers often pursue detours and alternate routes that often involve local roads. Making the decision behind the steering wheel is often challenging. The driver does not know whether the congestion is about to ease or is worsening. In addition, when many vehicles decide to execute the same travel plan, local roads become flooded with traffic that exceeds its capacity and sometimes deadlocks take place. Contemporary ITS and traffic advisory schemes are both slow to report traffic problems and usually do not provide any mitigation plan. An AVC-based solution will be the most appropriate and effective choice. Basically, vehicles in the vicinity will be able to query the plan of each other and estimate the impact on local roads. In addition, an accurate assessment of the cause of the congestion and traffic flow can be made by contacting vehicles close to where the bottleneck is. In addition, appropriate safety precautions can be applied to cope with the incident, e.g. poor air quality due to the smoke of a burned vehicle.

D. Scenario IV: sharing on-road safety messages.

The trend in the car manufacturing industry is to equip new vehicles with major sensing capabilities in order to achieve efficient and safe operation. For example, Honda is already installing cameras on their Civic models in Japan. The cameras track the lines on the road and help the driver stay in lane. A vehicle would thus be a mobile sensor node and an AVC can be envisioned as a huge wireless sensor network with very dynamic membership. It would be beneficial for a vehicle to query the sensors of other vehicles in the vicinity in order to increase the fidelity of its own sensed data, get an assessment of the road conditions and the existence of potential hazard ahead. For example when the tire pressure sensor on a vehicle reports the loss of air, vehicles that are coming behind on the same lane should suspect the existence of nails on the road and may consider changing the lane. The same happens when a vehicle changes lane frequently and significantly exceeds the speed limit; vehicles that come behind, and which cannot see this vehicle, can suspect the presence of aggressive drivers on the road and consider staying away from the lanes and/or keeping a distance from the potentially dangerous driver. The same applies when detecting holes, unmarked speed breakers, black ice, etc.

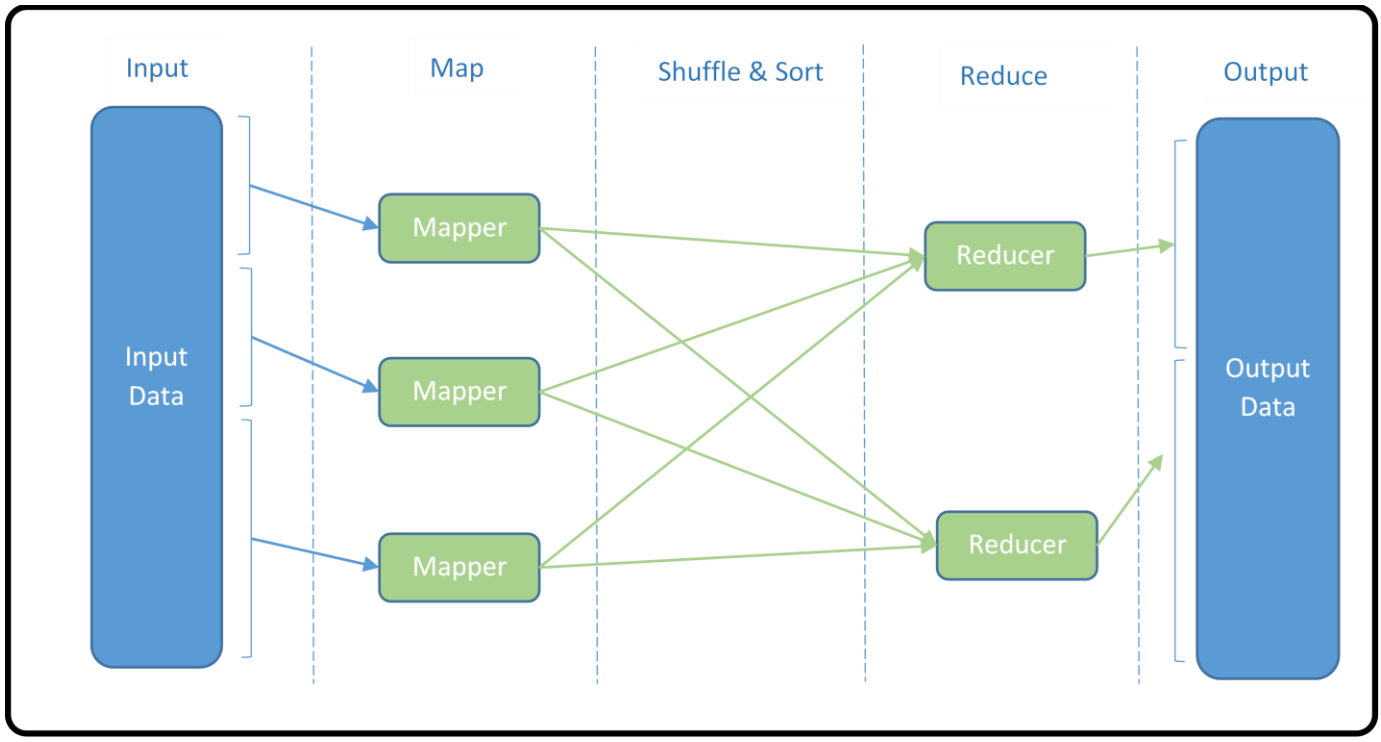


Fig. 1. MapReduce Architecture.

Contemporary VANET design cannot pull together the required solution and foster the level of coordination needed for providing these safety measures.

E. Scenario V: Self-organized high occupancy vehicle (HOV) lanes.

For precise and predefined travel time, HOV lanes carry a vast number of cars, which carry many passengers especially during periods of high traffic congestion. However, the authorities know about the congestion and have the official power to setup HOV lanes, but they do not have sufficient resources to compute and assess the situation to establish the time frame to use the HOV lane to ease the effects of traffic jam. VCs could setup HOV lanes dynamically by stimulating the flow of traffic and reducing the travel times for HOV lanes. VCs can dynamically provide the solution by gathering data from onboard vehicle sensors, and this type of solution is not possible with the current technology.

F. Scenario VI: VC in developing countries perspective.

Due to lack of sophisticated centralized decision support systems and infrastructure, the concept of VC will be very important in developing countries as well. Moreover, VCs will Play a vital role in making a vast number of computing resources accessible through a vehicular network by using many computing applications dynamically, which are not possible to use with the current infrastructure.

IV. Analyzing Data With Hadoop

Hadoop [13] is a well-adopted, standards-based, open-source software framework built on the foundation of Google's MapReduce framework and a distributed file system called Hadoop Distributed File System (HDFS) based on Google's File System papers [14]. MapReduce is a parallel programming technique derived from the functional programming concepts and is proposed for large-scale data processing in a distributed computing environment.

The focus is on scanning hundreds or thousands of files in parallel to run a set of repetitive actions in a batch-oriented processing fashion. Hadoop provides tools for the analysis and transformation of very large data sets using the MapReduce paradigm. It is very popular among academic institutions and real industries. It is suitable for fields such as web search, email spam detection, genome manipulation, social networks, economic computations and for analysis of unstructured data such as log and text. An important characteristic of Hadoop is the partitioning of data and computation across many of hosts, and executing application computations in parallel close to their data. A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers.

A. MapReduce

Hadoop leverages a cluster of nodes to run MapReduce programs massively in parallel. The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the MapReduce library expresses the computation as two functions: Map and Reduce. Fig.1 shows what is happening when a hadoop job is running. The first step when running a Hadoop job is to process input data. Data are broken into data blocks and stored across the local files of different nodes. Data's replication is very important for reliability. Each input split contains several lines of data, each line is read as a record and then will be wrapped as key objects and value objects.

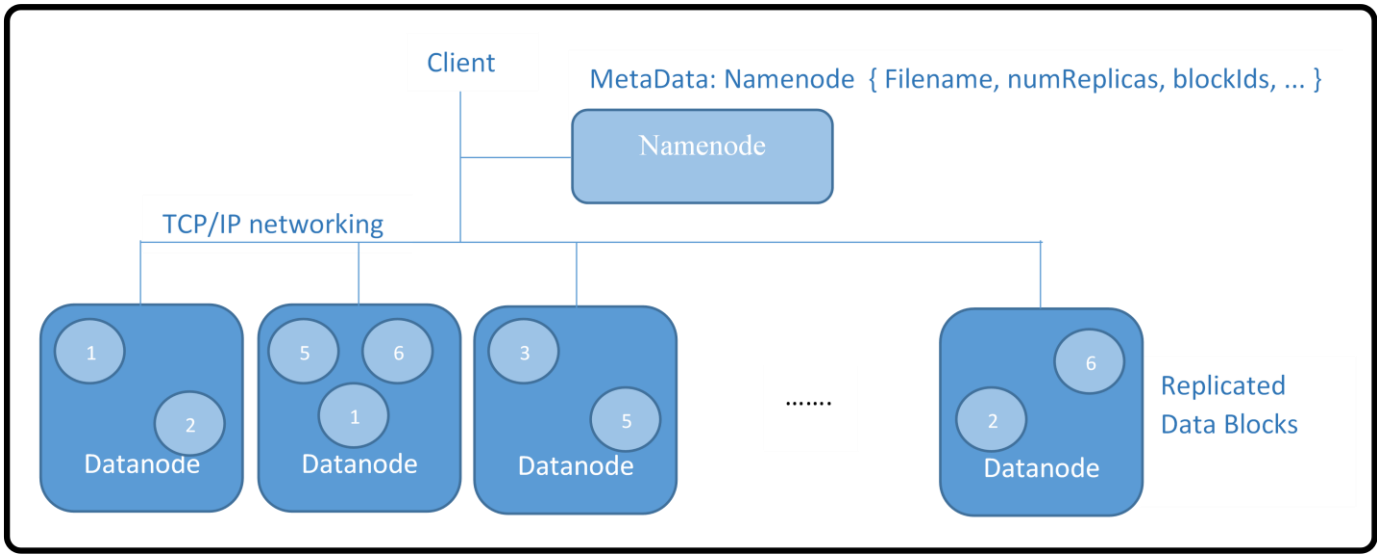


Fig. 2. HDFS Architecture.

Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The map function will consume a key and a value object and emit a key and value object as well. During this process, all the records will be executed by the same map function. Each node in order to run a job uses local system resources like CPU, memory and storage. The local files that constitute the file system are called HDFS. The number of nodes in each cluster varies from hundreds to thousands of machines. Hadoop can also allow for a certain set of fail-over scenarios. After all the map tasks finish, the reduce tasks will pull the corresponding partitions from the output of the map tasks. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function.

The Reduce function, also written by the user, accepts an intermediate key and a set of values for that key. The framework groups all the pairs, which have the same key and invokes the reduce function passing the list of values for a given key. It merges together these values to form a possibly smaller set of values. Typically, just zero or one output value is produced per Reduce invocation. All these data will be sorted and merged in the reduce tasks to make sure that all the values with the same key will be put together.

Finally reduce tasks will run and produce the output data. This last step is important to assemble intermediate results into a final result.

B. HDFS

HDFS is the storage component of Hadoop. While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favor of improved performance for the applications at hand. It is a file system optimized for high throughput and works best when reading and writing large files. To support this throughput HDFS leverages unusually large for a filesystem block sizes and data locality optimizations to reduce network input/output (I/O).

Like in a file system for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units. Unlike a filesystem for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage. HDFS stores file system metadata and application data separately. As in other distributed file systems, like GFS [13] [14], Lustre [15] and PVFS [16] [17], HDFS stores metadata on a dedicated server, called the NameNode. Application data are stored on other servers called DataNodes. All servers are fully connected and communicate with each other using TCP-based protocols.

In Fig.2 we can see HDFS architecture. HDFS maps all the local disks to a single file system hierarchy allowing the data to be dispersed at all the data/computing nodes. Scalability and availability are also key traits of HDFS. In order to provide data reliability HDFS uses block replication. Initially, each block is replicated by the client to three data-nodes. The block copies are called replicas. A replication factor of three is the default system parameter, which can either be configured or specified per file at creation time. For HDFS, the most important advantage of the replication technique is that it provides high availability of data in high demand. This is actively exploited by the MapReduce framework, as it increases replications of configuration and job library files to avoid contention during the job startup, when multiple tasks access the same files simultaneously.

Once the block is created, its replication is maintained by the system automatically. The name-node detects failed data-nodes, or missing or corrupted individual replicas, and restores their replication by directing the copying of the remaining replicas to other nodes. Hadoop schedules the MapReduce computation tasks depending on the data locality and hence improving the overall I/O bandwidth. This setup is well suited for an environment where Hadoop is installed in a large cluster of commodity machines. Hadoop stores the intermediate results of the computations in local disks, where the computation tasks are executed, and then informs the appropriate workers to retrieve them for further processing.

Although this strategy of writing intermediate result to the file system makes Hadoop a robust technology, it introduces an additional step and a considerable communication overhead, which could be a limiting factor for some MapReduce computations. Different strategies such as writing the data to files after a certain number of iterations or using redundant reduce tasks may eliminate this overhead and provide a better performance for the applications.

C. NameNode

The HDFS namespace is a hierarchy of files and directories. The NameNode is the master of HDFS that directs the slave DataNode daemons to perform the low-level I/O tasks. The NameNode is the bookkeeper of HDFS; it keeps track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed filesystem. The function of the NameNode is memory and I/O intensive.

Files and directories are represented on the NameNode by inodes, which record attributes like permissions, modification and access times, namespace and disk space quotas. The file content is split into large blocks and each block of the file is independently replicated at multiple DataNodes. The NameNode typically doesn't store any user data or perform any computations for a MapReduce program to lower the workload on the machine. The NameNode maintains the namespace tree and the mapping of file blocks to DataNodes.

When a HDFS client wanting to read a file first contacts the NameNode for the locations of data blocks comprising the file and then reads the specific block contents from the DataNode closest to the client. Then, the client writing data, requests the NameNode to nominate a suite of three DataNodes to host the block replicas. Finally, the client then writes data to this three DataNodes.

The current design has a single NameNode for each cluster. The cluster can have thousands of DataNodes and tens of thousands of HDFS clients per cluster, as each DataNode may execute multiple application tasks concurrently. HDFS keeps the entire namespace in RAM. The inode data and the list of blocks belonging to each file comprise the meta-data of the name system called the image. The NameNode also stores the modification log of the image called the journal in the local host's native file system. During restarts the NameNode restores the namespace by reading the namespace and replaying the journal. The locations of block replicas may change over time and are not part of the persistent checkpoint, which is the persistence record of the image.

D. DataNodes

Each slave machine in your cluster will host a DataNode daemon to perform the grunt work of the distributed filesystem, reading and writing HDFS blocks to actual files on the local filesystem. When you want to read or write a HDFS file, the file is broken into blocks and the NameNode will tell your client which DataNode each block resides in. Each block replica on a DataNode is represented by two files in the local host's native file system. The first file contains the data itself and the second file is block's metadata including checksums for the block data and the block's generation stamp.

During startup each DataNode connects to the NameNode and performs a handshake. The purpose of the handshake is to verify the namespace ID and the software version of the DataNode. If either does not match that of the NameNode the DataNode automatically shuts down. The namespace ID is assigned to the file system instance when it is formatted. The namespace ID is persistently stored on all nodes of the cluster. Nodes with a different namespace ID will not be able to join the cluster, thus preserving the integrity of the file system. The consistency of software versions is important because incompatible version may cause data corruption or loss, and on large clusters of thousands of machines it is easy to overlook nodes that did not shut down properly prior to the software upgrade or were not available during the upgrade. A DataNode that is newly initialized and without any namespace ID is permitted to join the cluster and receive the cluster's namespace ID. After the handshake the DataNode registers with the NameNode. DataNodes persistently store their unique storage IDs. The storage ID is an internal identifier of the DataNode, which makes it recognizable even if it is restarted with a different IP address or port. The storage ID is assigned to the DataNode when it registers with the NameNode for the first time and never changes after that. A DataNode identifies block replicas in its possession to the NameNode by sending a block report. A block report contains the block id, the generation stamp and the length for each block replica the server hosts. The first block report is sent immediately after the DataNode registration. Subsequent block reports are sent every hour and provide the NameNode with an up-to-date view of where block replicas are located on the cluster.

During normal operation DataNodes send heartbeats to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available. The default heartbeat interval is three seconds. If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable. The NameNode then schedules creation of new replicas of those blocks on other DataNodes. Heartbeats from a DataNode also carry information about total storage capacity, fraction of storage in use, and the number of data transfers currently in progress. These statistics are used for the NameNode's space allocation and load balancing decisions. The NameNode does not directly call DataNodes. It uses replies to heartbeats to send instructions to the DataNodes. The instructions include commands to replicate blocks to other nodes, remove local block replicas, reregister or to shut down the node and send an immediate block report.

These commands are important for maintaining the overall system integrity and therefore it is critical to keep heartbeats frequent even on big clusters. The NameNode can process thousands of heartbeats per second without affecting other NameNode operations.

V. VEHICULAR NETWORK DESIGN

In this section, we introduce the design of our system. Fig.3 shows the cloud architecture for vehicular networks [19]. It is a hierarchical architecture that consists of three interacting

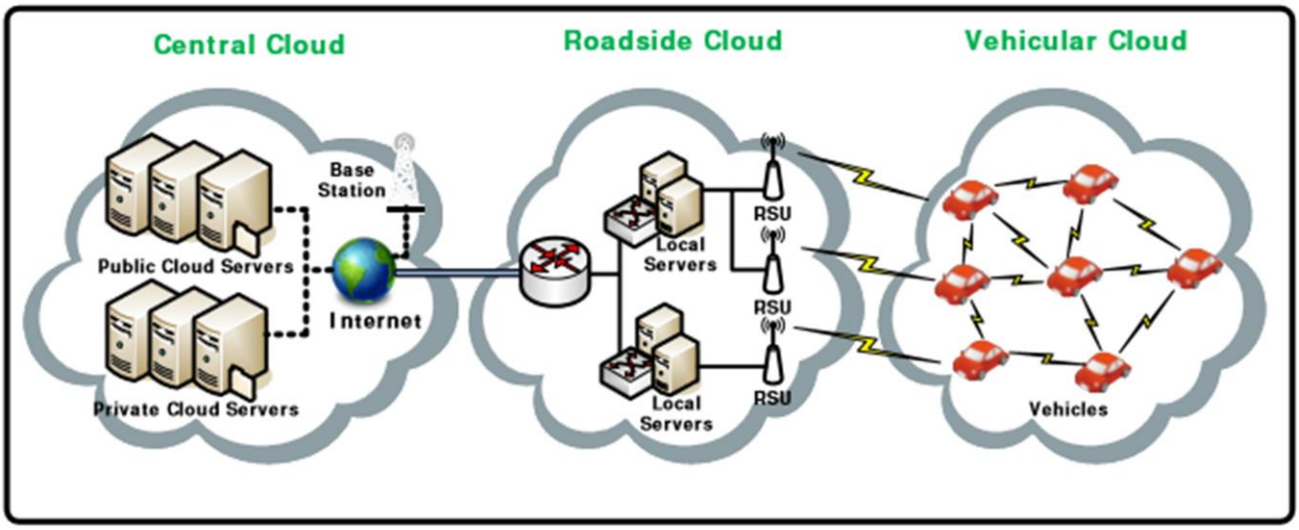


Fig. 3. Proposed cloud-based vehicular networks architecture.

layers: vehicular cloud, roadside cloud, and central cloud. Vehicles are mobile nodes that exploit cloud resources and services.

- Vehicular Cloud: a local cloud established among a group of cooperative vehicles. An inter-vehicle network, a VANET, is formed by V2V communications. The vehicles in a group are viewed as mobile cloud sites and they cooperatively create a vehicular cloud.
- Roadside Cloud: a local cloud established among a set of adjacent roadside units. In a roadside cloud, there are dedicated local cloud servers attached to Roadside Units (RSUs). A vehicle will access a roadside cloud by V2R communications.
- Central Cloud: a cloud established among a group of dedicated servers in the Internet. A vehicle will access a central cloud by V2R or cellular communications.

This architecture has several essential advantages. First, the architecture fully utilizes the physical resources in an entire network. From vehicles to roadside infrastructures and data center, the computation and storage resources are all merged into the cloud. All clouds are accessible to all vehicles. Second, the hierarchical nature of the architecture allows vehicles using different communication technologies to access to different layers of clouds accordingly. Hence, the architecture is flexible and compatible with heterogeneous wireless communication technologies, e.g., DSRC, LTE/LTE-Advanced and CR technologies. Third, the vehicular clouds and the roadside clouds are small-scale localized clouds. Such distributed clouds can be rapidly deployed and provide services quickly.

A. Vehicular Cloud

In a vehicular cloud, a group of vehicles share their computation resources, storage resources, and spectrum resources. Each vehicle can access the cloud and utilize services for its own purpose. Through the cooperation in the group, the physical resources of vehicles are dynamically scheduled on demand. The overall resource utilization is significantly enhanced. Compared to an individual vehicle, a vehicular cloud has much more resources. Due to vehicle mobility, vehicular cloud implementation is very different from a cloud in a traditional computer net- work. We propose two customization strategies for vehicular clouds: Generalized Vehicular Cloud Customization (GVCC) and Specified Vehicular Cloud Customization (SVCC). In GVCC, a cloud controller is introduced in a vehicular cloud. A cloud controller is responsible for the creation, maintenance, and deletion of a vehicular cloud. All vehicles will virtualize their physical resources and register the virtual resources in the cloud controller. All virtual resources of the vehicular cloud are scheduled by the cloud controller. If a vehicle needs some resources of the vehicular cloud, it should apply to the cloud controller. In contrast to GVCC, SVCC has no cloud controller. A vehicle will specify some vehicles as candidate cloud sites, and directly apply for resources from these vehicles. If the application is approved, the corresponding vehicles become cloud sites, which will customize virtual machines (VMs) according to the vehicle demand. These two strategies, GVCC and SVCC, are quite different. With respect to resource management, GVCC is similar to a conventional cloud deployment strategy in which cloud resources are scheduled by a controller. A vehicle is not aware of the cloud sites where the VMs are built up. The cloud controller should maintain the cloud resources. During a cloud service, if a cloud site is not available due to vehicle mobility, the controller should schedule a new site to replace it. In SVCC, since there is no cloud controller, a vehicle has to select other vehicles as cloud sites and maintain the cloud resources itself. In terms of resource utilization, GVCC is able to globally schedule and allocate all resources of a vehicular cloud. GVCC has higher resource utilization than SVCC. However,

the operation of the cloud controller will need extra computation. Therefore, SVCC may be more efficient than GVCC in terms of lower system overhead.

B. *Roadside Cloud*

A roadside cloud is composed of two main parts: dedicated local servers and roadside units. The dedicated local servers virtualize physical resources and act as a potential cloud site. RSUs provide radio interfaces for vehicles to access the cloud. A roadside cloud is accessible only by the nearby vehicles, i.e., those located within the radio coverage area of the cloud site's RSU. This fact helps us recall the concept of a cloudlet. A cloudlet is a trusted, resource-rich computer or cluster of computers that is connected to the Internet and is available for use by nearby mobile devices [18]. In this article, we propose the concept of a roadside cloudlet. A roadside cloudlet refers to a small-scale roadside cloud site that offers cloud services to bypassing vehicles. A vehicle can select a nearby roadside cloudlet and customize a transient cloud for use. Here, we call the customized cloud a transient cloud because the cloud can only serve the vehicle for a while. After the vehicle moves out of the radio range of the current serving RSU, the cloud will be deleted and the vehicle will customize a new cloud from the next roadside cloudlet in its moving direction. When a vehicle customizes a transient cloud from a road- side cloudlet, it is offered by virtual resources in terms of virtual machine (VM). This VM consists of two interacting components: the VM-base in the roadside cloudlet and the VM-overlay in the vehicle. A VM-base is a resource template recording the basic structure of a VM, while a VM-overlay mainly contains the specific resource requirements of the customized VM. Before a cloud service starts, the vehicle will send the VM-overlay to the roadside cloudlet. After combining the VM-overlay with the VM-base, the roadside cloudlet completes the customization of a dedicated VM. During a cloud service, as the vehicle moves along the roadside, it will switch between different RSUs. For the continuity of cloud service, the customized VM should be synchronously transferred between the respective roadside cloudlets. This process is referred to as VM migration.

C. *Central Cloud*

Compared to a vehicular cloud and a roadside cloud, a central cloud has much more resources. The central cloud can be driven by either dedicated servers in vehicular networks data center or servers in the Internet. A central cloud is mainly used for complicated computation, massive data storage, and global decision. There already exists mature open source or commercial software platforms that could be employed for the deployment of a central cloud.

VI. Our Hadoop Based Manager Architecture

Programming models used by cloud-based distributed batch processing infrastructures such as Hadoop allow parallel processing of data. For example, with Hadoop, the location and sensor data analysis algorithms can be implemented as

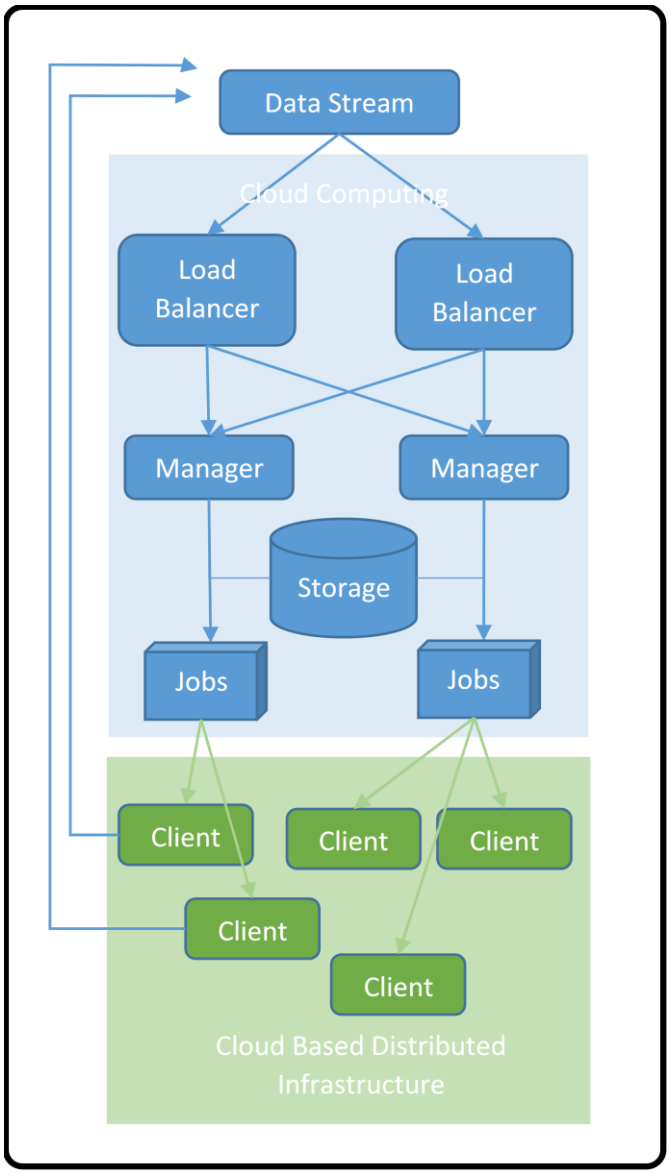


Fig. 4. Cloud-based architecture used by our proposed framework.

MapReduce jobs. Scaling out the computation on a large number of machines in a cluster is simple with Hadoop. The same computation that runs on a single machine can be scaled to a cluster of machines with few configuration changes in the program. Cloud-based distributed batch processing infrastructure such as Hadoop allows scaling the data analysis jobs up or down very easily which makes analysis flexible. With this flexibility in data analysis jobs, the frequency of analysis jobs can be varied.

In order to design our Hadoop based cloud framework we used basic Hadoop's ideas and change them so that can be suitable for modular Datanodes with wireless connectivity. Fig.4 shows a cloud deployment architecture used in our proposed framework. In this deployment architecture, tier-1 consists of the web servers load balancers, tier-2 consists of application servers and tier-3 consists of a cloud based distributed batch processing infrastructure such as Hadoop. Compute intensive tasks such as data processing are

formulated as MapReduce jobs which are executed on vehicles using Hadoop. This deployment is suitable for massive scale data analytics. Data is stored in a cloud based distributed storage such as Hadoop Distributed File System (HDFS).

Cloud-based deployment architecture leverages the dynamic scaling capabilities of computing clouds. Two types of scaling options are available for the cloud-based deployment, described as follows:

- Horizontal Scaling (scaling-out): Horizontal scaling or scaling-out involves launching and provisioning additional server resources for various tiers of the deployment.
- Vertical Scaling (scaling-up): Vertical scaling or scaling-up involves changing the computing capacity assigned to the server resources while keeping the number of server resources constant.

Our manager consist of two different type of nodes, master nodes and slave nodes. Master nodes works as managers. They keep track of how your files are broken down into file blocks, which nodes store those blocks, and the overall health of the distributed filesystem. The function of the NameNode is memory and I/O intensive. Each slave machine-vehicle in the cluster will host a slave node daemon to perform the grunt work of the distributed filesystem, reading and writing blocks to actual files on the local filesystem.

A. Master node

Master nodes work like servers. They wait for new connections from vehicles. The most important is that in our distributed system we can have more than one master node. Also during data analyses they store information about cpu and memory statistics from each slave node and data analyses results.

1) Data

The first thing that the master does is, like in a file system for a single disk, to break the files into block-sized chunks which are stored as independent files allowing the data to be dispersed at all the computing nodes. HDFS is built upon the single-node namespace server architecture. Since the name-node is a single container of the file system metadata, it naturally becomes a limiting factor for file system growth. In order to make metadata operations fast, the name-node loads the whole namespace into its memory, and therefore the size of the namespace is limited by the amount of RAM available to the name-node. In order to avoid this each time a vehicle downloads a file to analyze just appends to file's name info like what time the download happen and from whose vehicle.

In order to provide data reliability HDFS uses block replication. Initially, each block is replicated by the client to three data-nodes. The block copies are called replicas. A replication factor of three is the default system parameter, which can either be configured or specified per file at creation time. In our case vehicles do not have a lot of local storage, so we do not use this technique.

When slave nodes/vehicles want to get more data to analyze they just pull them from master node. This is very important because when we have a cloud with thousands of connected clients that request data or message exchange service processing load is very big.

2) Slaves management

Because we cannot know how many vehicles will be part of the cloud master, it keeps waiting for new connections. We can add at any time more wireless nodes for data processing.

In Hadoop during normal operation, data-nodes periodically send heartbeats to the name-node to indicate that the data-node is alive. Heartbeats also carry information about total and used disk capacity and the number of data transfers currently performed by the node, which plays an important role in the name-node's space and load-balancing decisions. The default heartbeat interval is three seconds. If the name-node does not receive a heartbeat from a data-node in 10 minutes, it pronounces the data-node dead and schedules its blocks for replication on other nodes. In our case because we use wireless moving nodes, the connections are terminated very often, so we do not use this heartbeat model. When a slave node is offline the data he analyzes is sent to other nodes.

3) Slaves communication

Master node undertakes message exchange between slave nodes when a connection is available for security reasons that are analyzed in the next section.

B. Slave node

Vehicular clouds have challenges that look alike to the difficulties faced in VANET and cloud computing. Slave node architecture based in a try to give solutions to challenges addressed by vehicular clouds.

1) Security and privacy

Security and privacy are very important aspects for the establishing and maintaining of the trust of users in VC. Privacy measures are required to ensure the VC communication and information in the isolated and trustworthy environment, while security procedures are needed to protect against network threats. Establishing trust relationships between several participants is a vital part of trustworthy communication and computation. It is impossible to provide a secure environment for vehicular cloud computing without considering some security requirements such as confidentiality, integrity and authentication.

- Confidentiality: Sensitive data should not be disclosed by unauthorized users,
- Integrity: Data should not be tampered with or modified. The messages must be reliable and valid,
- Availability: Data should be available whenever they are needed.
- Authentication: Determining whether someone or something is, who or what it is claimed to be,
- Privacy: The user's privacy should be preserved, and

- Real-time constraints: Some applications, such as accident alerts, require real-time or near real-time communication.

Large numbers of papers have addressed security issues related to Vehicular ad hoc network (VANET). Active and Passive location security has been proposed by Yan et al. [20]. The users are authenticated and validated using digital signatures. The messages are encrypted so as not to disclose the message contents. GeoEncrypt has been proposed in VANET by Yan et al. [21]. In the recent times, cloud security problems have been addressed. The solutions were to restrict the hardware accessibility to ensure minimal risks from insiders [22]. New platform was proposed by Santos et al. [23] to have a trustworthy conventional cloud. Santos proposed a new platform to achieve trust in conventional clouds. A trust coordinator maintained by an external third party is imported to validate the entrusted cloud manager, which makes a set of virtual machines (VMs) such as Amazon's E2C (i.e., Infrastructure as a Service, IaaS) available to users. Garfinkel et al. [24] proposed a solution to prevent the owner of a physical host from accessing and interfering with the services on the host. Berger et.al [25] and Murray et.al [26], adopted similar solution to prevent the interference in the host services by the physical host owner. Jensen et al. [27] stated technical security issues of using cloud services on the Internet access.

The vehicular cloud has to authenticate the highly mobile vehicles. It has to verify the identity of users and check for the integrity of messages received from them. A few authentication metrics that can be adopted [28] are:

- Ownership: Unique identity like security token, identity card and user owned software token.
- Knowledge: Security questions, passwords, private identification numbers.
- Biometrics: Human biometrics like fingerprints, signature, eye scan, voice.

In VANET different attacks are possible such as, Denial of Service attack, Fabrication Attack, Alteration Attack, Replay Attack, Message Suppression Attack, Replay Attack, Sybil [29] Attack. Again, different attackers in VANET are such as, Selfish Driver, Malicious Attacker [30]. To overcome that some security requirements should be considered such as Authentication, Availability, Non-repudiation, Privacy, Real-time constraints, Integrity, Confidentiality [31].

The quality of information (QoI) retrieved from vehicles is also a concern. Data/information retrieved needs to be verified and validated before making decisions and/or publishing to the public. Authentication and authorization of the nodes accurately in the intermittent short-range communication relates to the integrity of data. The complexity increases with the increase of nodes. User identity, spoofing and tempering data could be the main threat to the network where the attacker pretends to be another user of same priority level. As a solution to this problem is suggested the use of SSH File Transfer Protocol (also Secure File Transfer Protocol, or SFTP). The SFTP is a network protocol that provides file access, file transfer, and file management functionalities over any reliable data stream. It was designed by the Internet Engineering Task Force (IETF) as an extension of the Secure Shell protocol (SSH) version 2.0 to provide secure file transfer capability. This protocol is assumed to run over a secure channel, such as SSH, that the server has already authenticated the client, and that the identity of the client user is available to the protocol. In this way, only authenticated users can connect and upload data to master node. Vehicular networks lacks the relatively long life context, so personal contact of user's device to a hot spot will require long life password and this will be impractical for securing VC. The generated public keys solves problems like that.

The attackers face many challenges as well. The high mobility has both pros and cons. The attackers will have to make repeated attempts to harm the vehicles. The access to each VM is transitory as the vehicles are bound to move across the states or districts. Moreover, the attackers have to locate the machine on which targets would be lying as it's a distributed environment. Experiments have been done to catch and compare the memory of processors, and users can find co-residence in the same physical machine [32]. The attackers must be physically co-located with the target user on the same physical machines. This will require attackers to be physically present at the same region with the target vehicles or shadow with the target vehicles at the same speed and have to collect valuable information with certain privileges or with security tokens.

2) Data collection

Vehicle-generated data or data obtained from neighboring vehicles can be stored until the vehicle reaches a dedicated data collector or kept in the vehicle until retrieved as a reply to queries sent by data-seeking vehicles. This communication paradigm is considered a case of delay/disruption-tolerant networks. Each vehicle keeps the sensed data in local storage, and send its stored data to the master node every time a connection is available. The vehicles are not able to access or alter the stored data on other vehicles. Therefore, the sensitive data that are stored in each vehicle should not be encrypted to protect it against the unauthorized access. The vehicles are authorized only to send to other vehicles and never to read. They can write only to master node.

3) Messages exchange

Wireless message exchange between vehicles is known as inter-vehicle communication and a network of communicating vehicles is known as a Vehicular Ad-Hoc Network (VANET). A VANET is a sub-category of wireless multi-hop networks in which a source depends on intermediate nodes to relay messages to a destination. In this communication paradigm, vehicles can be considered a resource for relaying data to other nodes out of the communication range of the source node. A vehicle can be used not only for relaying data to other neighboring vehicles but also to/from Road Side Units (RSUs).

All nodes, such as vehicles and road-side infrastructures are able to communicate with each other based on the V2V or V2I communication models in vehicular cloud computing. Furthermore, the vehicles and road-side infrastructures are required to communicate with the cloud to store or process their data. An OBU is responsible for establishing communication between vehicles or between vehicles and infrastructures. A RSU is an access point that is connected to a

location server that records or processes all location data forwarded by RSUs. The location server sends the data to the cloud for processing or storage. Furthermore, a trusted certificate authority (CA) is in charge of providing authentication services for vehicles and location-based service providers. The VC's message is constructed from some fields such as: vehicle id or pseudonym to identify the driver, time, message type, length of message, data, and direction. The type of message can be:

- Short message: to send an alert message or warning messages
- Media message: to get an environment services from other vehicles or a cloud
- Priority message: to end the alert messages or urgent messages
- Acknowledge message: to confirm the delivery of messages.

Providing secure communication in a vehicular cloud plays a central role in creating safer and more efficient driving. There are several security holes in vehicular communication, which make it vulnerable against attackers, for example, preventing communication (jamming), forging messages and transmitting false hazard warnings by the attacker (forgery), dropping or modifying a message by intermediate nodes (traffic tampering), and privacy violations. In order to make safer the communication the message exchange between nodes is checked by master node.

4) Mobility

One of the main characteristics of VC is the mobility of the nodes which directly affects the available computational capabilities and storage resources, for example, the number of parked vehicles in the parking is not constant. Therefore, to provide fluctuating application requirements and resource accessibility on the move, the necessary related protocol architecture and VC networking must be developed. We design our nodes so that can request data to analyze depending in their capabilities such as channel bandwidth and available free storage. Vehicles' mobility is considered as a plus that allows vehicles to cover wider areas compared to their static counterparts. However, while pooling resources from a vehicle, the vehicle may leave the area of interest or connectivity before reporting/relaying the desired data, or finishing the task in hand. Dynamic availability of resources may also be temporal stemming, for instance, from the high need for resources during the rush hours and their idleness during late nights. Effective resource management techniques are needed for proper task assignment and retrieval.

Interruption of data transmission between faster moving nodes in the highways becomes imminent as the connectivity lasts only for a few moments. On the contrary, during traffic jam congestion of transmitter nodes will create noise and interference [33]. Obstructed by large buildings and other infrastructures in the metropolitan areas the signal attenuates at a higher rate resulting in both deterioration of signal strength and signal quality. Obstructed by large buildings and other infrastructures in the metropolitan areas the signal attenuates at a higher rate resulting in both deterioration of signal strength and signal quality [34].

To solve these problems slave node check's signal level of wireless connection. If signal level is too low or it is decreasing very fast the node does not try to read or write data at master node.

5) Vehicles capabilities

The vehicles usually have a large number of different on-board devices, including GPS, wireless transceivers, and on-board radar devices. The different vehicles are able to have a different combination of these on-board devices with different capabilities such as speed of processor, volume of memory, storage, and CPU capacity. Therefore, providing a security for these heterogeneous vehicles in VC environment is difficult because most of the cryptographic algorithms are not lightweight and the vehicles need to have certain hardware conditions. In our case due to the restrictions vehicles have, encryption is optional.

One challenge in VC design is the ability to connect to the Internet and to its resources, including Cloud Services. While Vehicle-To-Infrastructure (V2I) communications (via DSRC, WiFi and 3G/4G) are becoming increasingly more reliable in civilian settings, in environments such as battlefields and urban emergence, infrastructure access will be very limited, suggesting that Vehicle-To-Vehicle (V2V) communications will be used for critical vehicle cloud services like internal data storage, searching and sharing.

6) Network Scalability

The vehicular clouds must be capable of addressing security schemes for dynamically growing number of vehicles. The dynamics of traffic produces dynamic demands on security. They should not only handle the usual traffic but ought to take care of extra fluctuation that may occur in case of emergencies or natural disasters. Recently, Content-Based Networking (CBN) has attracted much attention as a method for searching content. In CBN, a data object (viewed as a chain of chunks) is searched and retrieved based on its identity instead of the IP address of the node on which it resides. While several independent CBN designs exist, all designs have the following common attributes:

- receiver-oriented chunk based transport
- in-network per-chunk caching
- name-based forwarding
- uniquely identifiable content naming

Intuitively, in-network caching is beneficial to highly mobile vehicular scenarios. A data retrieval failure due to intermittent connectivity can be recovered more quickly by leveraging distributed caches.

Based at CBN idea we create name-based routing. Name-based routing is different from host IP-based routing in two aspects. First, the number of content names to consider in name-based routing is significantly larger than the number of IP addresses. Second, in host-based networks, each IP address is associated with one host interface. In contrast, copies of a data object chunks may exist at different locations in CBN. Only master node stores information about connected vehicles names.

7) Data processing

Hadoop can be run in one of the three following modes: Standalone (or local) mode where there are no daemons running and everything runs in a single JVM. Standalone mode is suitable for running MapReduce programs during development, since it is easy to test and debug them. Pseudo-distributed mode where the Hadoop daemons run on the local machine, thus simulating a cluster on a small scale. Lastly fully distributed mode where the Hadoop daemons run on a cluster of machines. By default, Hadoop is configured to run in a non-distributed mode, as a single Java process.

In standalone mode that is suitable for our platform, the local filesystem and the local MapReduce job runner are used. Each vehicle run Hadoop as standalone mode and waits for data to be analyzed.

VII. EVALUATION

The Wireless Signal Propagation Emulator developed by CMU [35] accurately emulates wireless signal propagation in a physical space. The emulator senses signals generated by known wireless sources through the antenna port, subjects the signals to the same effects that occur in a real physical space (e.g. attenuation, multi-path fading, etc.), and feeds the combined signals back into wireless cards. The emulator, however, has limitations in reproducing arbitrary motion patterns. In addition, although the propagation scenario is more realistic, it is still artificially created as opposed to real life measurement.

Testbeds play a key role in network research, because a wireless medium has physical characteristics that cannot easily be simulated. Over the years, many kinds of testbeds have been developed in wireless network. Performing testbed experiments in a wireless network (WNET), however, is challenging. In order to test our developed platform we use a WNET testbed with unmovable nodes. This testbed is available for use either via remote or on site access. As for mobility support, the outdoor testbed is grounded.

Testbed approaches are the only way to conduct VANET experiments with high fidelity, but the dynamic nature of VANETs results in experimental inconsistency. WNET allows researchers to repeat experiments with different protocol configurations to ensure consistency. The whole system is under full control and thus one can adjust a few parameters while keeping the rest unchanged. However, WNET assumes accurate modeling of physical characteristics such as mobility/traffic patterns, radio propagation models, and external interferences. In the case of mobility patterns the assumption that vehicles do not move is made. In order to test this system, the nodes are being put in different distances between them everytime the experiment take place.

A. NITOS Testbed

NITOS currently consists of 50 wireless nodes, deployed out-doors at the exterior of the University of Thessaly campus building. Users can perform their experiments by reserving slices (nodes, frequency spectrum) of the testbed through NITOS scheduler that together with cOntrol and Management Framework (OMF), support ease of use for experimentation and code development. Two Gigabit Ethernet switches interconnect the nodes with NITOS server, namely the Control switch that provide for control of experiment execution and measurement collection and the Experimental switch, which can be used for conducting wired experiments. A third Gigabit Ethernet, namely the Chassis Manager switch, is dedicated in controlling the operational status of the nodes through the transmission of custom http requests that control solid state relays on the Chassis Manager cards. NITOS nodes feature up to three wireless network interface cards (NICs), using the Atheros AR5424 and AR9380 chipsets. Each node use an Intel(R) Core (TM) 2 DUO processor in 2261MHz, 64bit, Ram 2Gb and 26GB Hard Disk.

B. Configuration

For the implementation of our mechanism, we used the MadWiFi open source driver. Thanks to the use of the MadWifi modules by the Linux kernel, it is possible to implement a Wireless Access Point with a Personal Computer or an Embedded Device that has a WiFi network card with an Atheros chipset. This feature is available starting with version 1.0.beta8 of Zeroshell, which introduces WiFi support in either AP (Access Point) or STA mode.

The following operational modes are supported:

- Sta: Station, a.k.a. infrastructure or managed. This device is acting as typical WLAN client station. This is the default mode if not otherwise specified. This is the mode that is used for slave nodes.
- Ap: Access Point, a.k.a. master. This device acts as the Access Point for other WLAN client stations and it is used by master nodes so that slaves can connect to them.
- Adhoc: Ad-hoc. a.k.a. IBSS mode. This device is in a peer-to-peer(s) WLAN without the need for an Access Point and it is used by the slaves when masters are not available.

The option to use the MadWifi drivers, combined with the use of wpa_supplicant and hostapd packages, is due to their ability to perform the functions of an AccessPoint with advanced features, for example:

Access authentication and wireless traffic encryption through WPA/WPA2 (RSN). It is supported either in WPA-PSK mode, in which the client, in order to be associated to an SSID, must know the Pre-Shared Key, or WPA-EAP mode, also known as WPA Enterprise, in which a user can become authenticated with Username and Password or a X.509 digital certificate validated by a Radius server. Both the TKIP encryption algorithm and the more secure CCMP, based on AES, are supported;

With management of the Multiple SSID mode, it is possible to create up to 4 virtual standalone Access Points for each WiFi network card in the system. It is clear that virtual SSIDs belonging to the same WiFi network card share the radio channel being used, and thus the available bandwidth. Moreover, for each virtual SSID it is possible to establish a standalone authentication and encryption scheme.

One of the four possible SSIDs can also work in Managed mode and associate to a WLAN as a client. For example, this is useful to extend the range of the Wireless network itself by

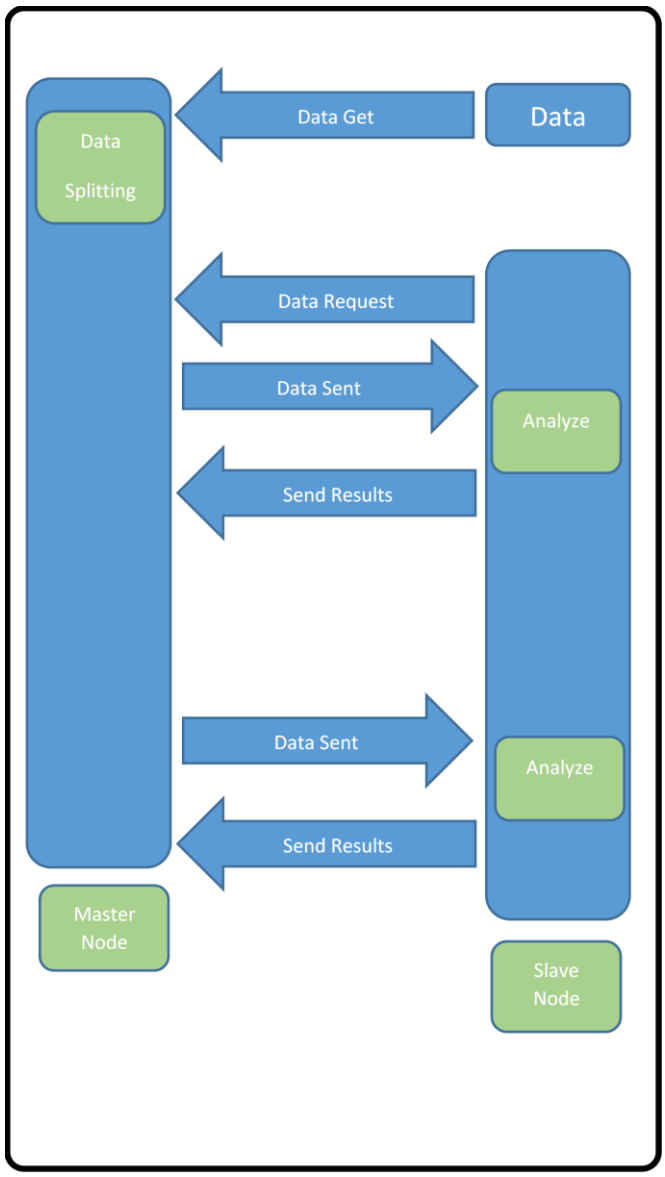


Fig. 5. Stages during data analyses.

implementing repeaters that work in WDS (Wireless Distribution System) but do not need to be interconnected by means of a Wired network.

Virtualization introduces an additional overhead, which may effect experiment results: Virtualization adds an extra software processing layer between application and hardware. In addition, hardware resources are used to run the virtual machines. Therefore, system performance is worse than a regular Linux system. For the experiments we created two Linux Distributions, one for master nodes and one for slave nodes.

In both masters and slaves Hadoop is installed in single mode with default configuration. When we start our manager in nodes additional information like file chunks size, data location, ports for message exchange and pseudo-name are required. Fig. 5 shows stages during data analyses with a single slave node. With green color we can see what is happening inside each node during time.

Benchmarking is an important issue for evaluating distributed systems, and extensive work has been conducted in this area. Various research and industry standard performance benchmarking solutions exist. TPC-C [36], TPCW [37], TPC-H [38], and SPEC OMP [39] are domain specific benchmarks: the first evaluates on-line transaction processing (OLTP) systems, the second applies to ecommerce web sites, the third evaluates decision support systems, and the fourth applies to parallel scientific computing applications that utilize OpenMP. SPECweb is another performance benchmark that evaluates web applications from different domains, such as banking, e-commerce and support [40].

Even though these benchmarks are useful in analyzing distributed systems in general, they are not applicable to MapReduce systems. The need of MapReduce benchmarking is motivated by the many recent works that have been devoted to the study and improvement of MapReduce performance and scalability. These include task scheduling policies in MapReduce, cost-based optimization techniques, replication and partitioning policies. MRBench also provides microbenchmarks in the form of a MapReduce implementation of TPC-H queries [41]. However, these microbenchmarks are not representative of full applications with complex workloads, and they do not provide automated statistics for performance. More recent work proposed HiBench [42], a benchmark which evaluates Hadoop in terms of system resource utilization, and Hadoop job and task performance. Although low-level resource monitoring may be useful in targeting specific system features, HiBench does not avoid the pitfalls of microbenchmarks, missing multiuser workloads for batch or interactive systems.

To evaluate our platform, we present four case studies. We investigate the scalability of the system with regard to the size of clusters and data input. Also we investigate, how chunks' size and number of files slave nodes get per request, affect data processing.

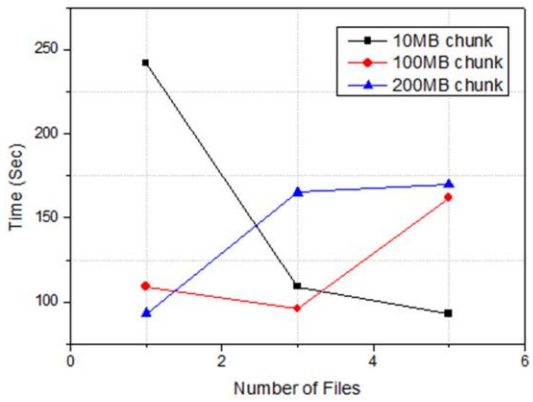
C. Chunks' file size and number of files get per request

After a lot of experiments we found that the chunks' file size and the number of files get by slave per request effect the amount of time for data processing. We conducted experiments running on one node for (a) 403MB, (b) 741MB and (c) 1.9GB data input size and 10MB, 100MB and 200MB chunk file size. Fig. 6 presents the performance results of the experiments. Fig. 6 shows the relationships between the amount of processed data and the execution durations for a given cluster. Response time results show that,

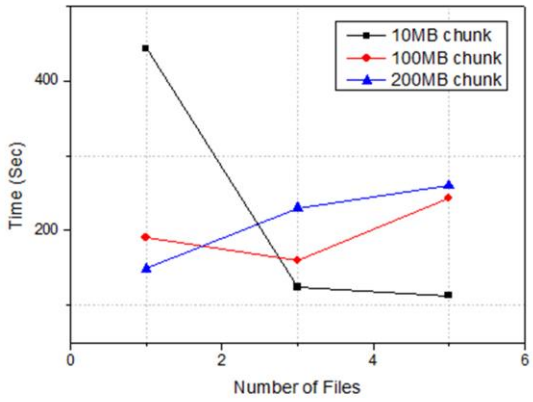
- when we can get a small number of files per request it is better to split data at bigger chunk size
- it is better to request bigger number of files when smaller chunks files are available
- in any other case the option of 100MB file chunk would gave the best results

In the case with wireless moving nodes the best scenario would be to use small chunks files, due to bandwidth limitations.

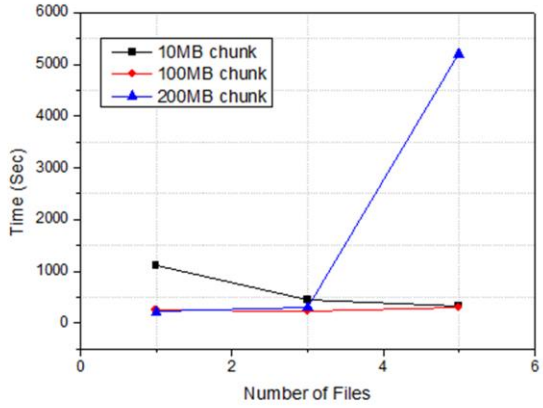
Fig. 7 represents the relationships between the amount of chunk size and the execution durations for (a) 1 file slave request and (b) 3 files slave requests. Response time results show that, the best scenario would be the use of 100MB chunks file size and 3 file get per request.



(a) 403MB Data size input.



(b) 741MB Data size input.

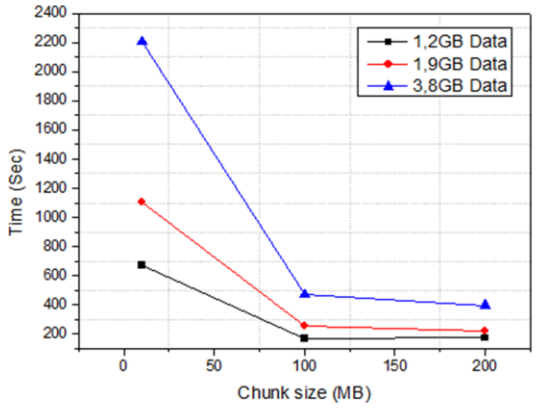


(c) 1.9GB Data size input.

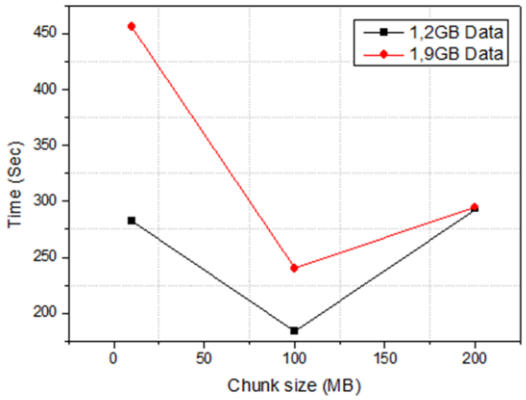
Fig. 6. The relationships between the amount of number of files per slave request and the execution durations for (a) 403MB, (b) 761MB and (c) 1.9GB data size input.

D. *Scalability With Regard To Cluster Size*

In this case study, we evaluate the scalability of our platform with regard to the size of clusters. We conducted experiments running clusters of different sizes: 3 and 7 nodes. We compare the results of these clusters with the results obtained when running on one node. We know that using more



(a) 1 file slave request.



(b) 3 files slave request.

Fig. 7. The relationships between the amount of chunk size and the execution durations for (a) 1 file slave request and (b) 3 files slave requests.

nodes every time we have and smaller processing times. So we run only one experiment just to see that this is true.

Here, response time results show that, using more nodes to process data the time is needed is smaller with 3.8GB data size input. We can see that in Fig. 8.

E. *Finding best case*

In this case study, we investigate how chunks' file size and number of files get by slave per request effect the amount of time for data processing by the cluster. We conducted experiments with 3.8GB data in the follow scenarios:

- Small number of files per request (1 file) and big chunk size (200MB)
- Big number of files (5 files) and small chunks files (10MB)
- The best option of 100MB file and 3 files per request

Fig. 8 presents performance results as functions of cluster size in the above 3 scenarios. As we can see the best option would be to use 100MB file chunks and getting 3 files each time we request data. We must notice that the differences are small because these three solutions are the best options we use.

F. *Scalability With Regard To Data Size*

In this case study, we investigate the scalability of our platform with respect to the size of input data. We conducted experiments with different sizes of input data: 403MB, 761MB, 1.2GB, 1.9GB and 3.8GB, 100MB file chunk size and 3 files get per slave request. The experiments were conducted on a 7 node clusters. Fig. 9 presents performance results as functions of input data size.

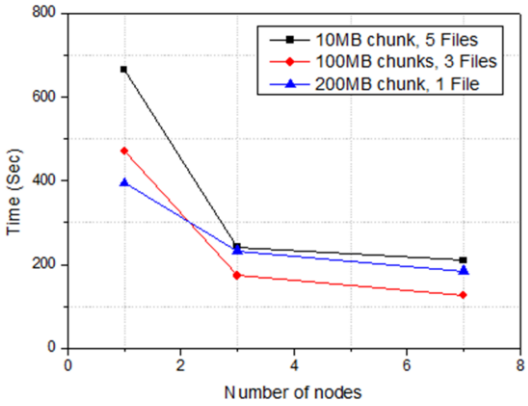


Fig. 8. The relationships between the amount of number of nodes and the execution durations for 3.8GB data size input.

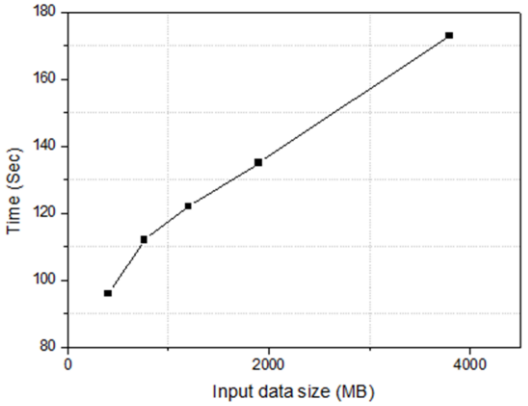


Fig. 9. The relationships between the data input size and the execution durations for 403MB, 761MB, 1.2GB, 1.9GB and 3.8GB data size input.

VIII. CONCLUSION

Running experiments in testbeds is essential to compare different parameters in realistic scenarios. However, the dynamic nature of VANET makes it difficult to faithfully reproduce the experiment environments. However, repeating experiments in a vehicle testbed is very costly and some scenarios are difficult to be precisely reproduced. The virtual machine, parallelized testbed, we proposed removes the aforementioned limitations by performing parallel runs under the same environmental conditions with a single set of hardware. The virtual machine also provides a complete operating system, so situations with different system requirement can still be processed in parallel. To sum up, the virtual machine parallelization provides consistent and reliable results from a single run with low cost.

REFERENCES:

[1] J. Chen, X. Cao, Y. Zhang, W. Xu, Y. Sun, "Measuring the performance of movement - assisted certificate revocation list distribution in VANET", Wireless Communications and Mobile Computing, 11 (7), 888-898, 2011.

[2] W.J. Fleming, "New automotive sensors—A review," in *IEEE Sensors Journal*, vol. 8, no. 11, 2008, pp. 1900-1921.

[3] US FEDERAL COMMUNICATIONS COMMISSION (FCC) (2003, September) "Standard Specification for Telecommunications and Information Exchange Between Roadside and Vehicle Systems—5 GHz Band Dedicated Short Range Communications (DSRC) Medium Access Control (MAC) and Physical Layer (PHY) Specifications" (Washington, DC).

[4] R. A. Uzcategui, and G. Acosta-Marum, "WAVE: A Tutorial", IEEE Communications Magazine, vol. 47, no. 5, pp. 126-133, May 2009.

[5] R. Yu, Y. Zhang, Y. Liu, S. Xie, L. Song and M. Guizani, "Secondary Users Cooperation in Cognitive Radio Networks: Balancing Sensing Accuracy and Efficiency", IEEE Wireless Communications Magazine, vol. 19, no. 2, April 2012, pp.2-9.

[6] S. Xie, Y. Liu, Y. Zhang, and Yu, "A Parallel Cooperative Spectrum Sensing in Cognitive Radio Networks", IEEE Transactions on Vehicular Technology, vol. 59, no. 8, pp.4079 C 4092, 2010.

[7] W. Tantisiriroj, S. Patil, G. Gibson. "Data-intensive file systems for Internet services: A rose by any other name ..." Technical Report CMU-PDL-08-114, Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh, PA, October 2008.

[8] S. Ghemawat, H. Gobioff, S. Leung. "The Google file system," in Proc. of ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp. 29–43.

[9] Brown A., Johnston S, Kelly K. "Using service-oriented architecture and component-based development to build web service applications", Rational Software Corporation, 2002.

[10] Olariu S, Hristov T, Yan G. "The next paradigm shift: from vehicular networks to vehicular clouds," in Mobile ad hoc networking: cutting edge directions, Basagni MC S, Giordano S, Stojmenovic I, 2nd ed. NJ, USA, John Wiley & Sons, Inc., Hoboken, 2013.

[11] T. Wang, L. Song, Z. Han, and B. Jiao "Popular Content Distribution in CR-VANETs with Joint Spectrum Sensing and Channel Access," to appear, IEEE Journal on Selected Areas in Communications.

[12] Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, et al., "The eucalyptus open-source cloud-computing system," in Proceedings of the 9th IEEE/ACM international symposium on cluster computing and the grid, Shang-hai, 2009, pp. 124–31.

[13] A. S. Foundation, "Apache hadoop." [Online]. Available: {http: //hadoop.apache.org/}.

[14] M. K. McKusick, S. Quinlan. "GFS: Evolution on Fast-forward," ACM Queue, vol. 7, no. 7, New York, NY. August 2009.

[15] Olariu, S., Khalil, I. and Abuelela. "Taking VANET to the Clouds". Pervasive Comput. Commun. 7, pp. 7–21.

[16] Lustre File System. http://www.lustre.org.

[17] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation, 2004, pp. 10–10. [Online].Available:http://portal.acm.org/citation.cfm?id=1251254.1251264.

[18] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. "The Case for VM-based Cloudlets in Mobile Computing", IEEE Pervasive Computing, vol. 8, no. 4, 2009.

[19] S. Olariu, M. Eltoweissy, and M. Younis, "Toward autonomous vehicular clouds," ICST Trans. Mobile Commun. Comput., vol. 11, no. 7–9, pp. 1–11, Jul.–Sep. 2011.

[20] G. Yan, S. Olariu, and M. C.Weigle, "Providing VANET security through active position detection," Comput. Commun., vol. 31, no. 12, pp. 2883–2897, Jul. 2008, Special Issue on Mobility Protocols for ITS/VANET.

[21] J. Sun, C. Zhang, Y. Zhang, and Y. M. Fang, "An identity-based security system for user privacy in vehicular ad hoc networks,"IEEE Trans. Parallel Distrib. Syst., vol. 21, no. 9, pp. 1227–1239, Sep. 2010.

[22] A. Friedman and D. West, “Privacy and security in cloud computing,” Center for Technology Innovation: Issues in Technology Innovation, no. 3, pp. 1–11, Oct. 2010.
[23] N. Santos, K. P. Gummadi, and R. Rodrigues, “Toward trusted cloud computing,” in Proc. HotCloud, Jun. 2009.
[24] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. B. Terra, “Virtual machine-based platform for trusted computing,” in Proc. ACM SOSP, 2003, pp. 193–206.
[25] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn, “VTPM: Virtualizing the trusted platform module,” in Proc. 15th Conf. USENIX Sec. Symp., Berkeley, CA, 2006, pp. 305–320.
[26] D. G. Murray, G. Milos, and S. Hand, “Improving XEN security through disaggregation,” in Proc. 4th ACM SIGPLAN/SIGOPS Int. Conf. VEE, New York, 2008, pp. 151–160.
[27] M. Jensen, J. Schwenk, N. Gruschka, and L. L. Iacono, “On technical security issues in cloud computing,” in Proc. IEEE Int. Conf. Cloud Comput., 2009, pp. 109–116.
[28] Fed. Fin. Inst. Examination Council, Authentication in an Internet banking environment 2009. [Online]. Available: http://www.ffiec.gov/pdf/authentication_guidance.pdf
[29] J. Douceur, “The sybil attack,” in Proc. Rev. Papers 1st Int. Workshop Peer-to-Peer Syst., 2002, vol. 2429, pp. 251–260.
[30] Levente Butty´an, Tam´as Holczer, and Istv´an Vajda, “On the Effectiveness of Changing Pseudonyms to Provide Location Privacy in VANETs” F. Stajano et al. (Eds.): ESAS 2007, LNCS 4572, pp. 129–141, 2007. c_Springer-Verlag Berlin Heidelberg 2007.
[31] Mousannif H, Khalil I, Al Moatassime H. Cooperation as a Service in VANETs. Journal of Universal Computer Science 2011; 17:1202-18.
[32] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, “Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds,” in Proc. 16th ACM Conf. CCS, 2009, pp. 199–212.
[33] Samara, G., Al-Salihy, W.A.H., Sures, R. “Security issues and challenges of Vehicular Ad Hoc Networks”, Proceedings of The 4th International Conference on New Trends in Information Science and Service Science (NISS), 2010, pp.393-398.
[34] Grilli,G., “Data Dissemination in Vehicular Networks”, PhD dissertation in Computer Science and Automation Engineering, June 2010. CMU wireless emulator, “http://www.cs.cmu.edu/emulator/”.
[35] Xen, http://www.xen.org/.
[36] “TPC-C: an on-line transaction processing benchmark”, http://www.tpc.org/tpcc/.
[37] “TPC-W: a transactional web e-Commerce benchmark”, http://www.tpc.org/tpcw/.
[38] “TPC Benchmark H - Standard Specification”, http://www.tpc.org/tpch/.
[39] Standard Performance Evaluation, “SPEC OpenMP Benchmark Suite”, http://www.spec.org/omp/.
[40] “SPECweb2009”, http://www.spec.org/web2009/.
[41] K. Kim, K. Jeon, H. Han, S.-g. Kim, H. Jung, and H. Y. Yeom, “MRBench: A Benchmark for MapReduce Framework”, in 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS ’08), 2008.
[42] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, “The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis”, in 22nd International Conference on Data Engineering Workshops (ICDE 2010), Los Alamitos, CA, 2010.