FIT Fukuoka Institute of Technology 福岡工業大学

GRADUATE SCHOOL OF ENGINEERING
FUKUOKA INSTITUTE OF TECHNOLOGY

Implementation of P2P Platforms for Vehicle Networks and Robot Control

by

Evjola Spaho

Adviser: Prof. Leonard Barolli

2013

Contents

List of Figures

List of Tables

Acknowledgements

Conducting research in doctoral level involves a great deal of collaboration with different people in various ways. For this reason, there are certain persons that I would like to convey my gratitude.

First and foremost, I would like to express my deepest gratitude to my advisor Prof. Leonard Barolli, for his orientation, support and guidance throughout the work.

I would like to thank Prof. Kazunori Uchida, Prof. Hiroyuki Fujioka and Prof. Valentin Claudiu Suciu for very good comments about this thesis as members of the inspection committee.

I am grateful to Prof. Fatos Xhafa, Technical University of Catalonia and Prof. Makoto Takizawa, Hosei University, Japan, for their continuous support and help.

I am also greatly indebted to Prof. Rozeta Miho and Dr. Vladi Kolici, Polytechnic University of Tirana, Albania for their deep and important teachings, which prepared me for the challenges of studies in Japan.

I would like to a give special thank to my friend Elis Kulla for his friendship and kind support.

I would like to thank Dr. Makoto Ikeda and Dr. Keita Matsuo for their continuous support and useful comments to improve the quality of this research and all members of INA Lab for their kind friendship.

Moreover, I have to thank my friends for their support and the great time that we have spent together.

I am very grateful to my family for their constant encouragement and love. This thesis would never have been possible without their continuous motivation, advice and support.

Abstract

Peer to peer (P2P) systems have become highly popular in recent times due to their great potential to scale and the lack of a central point of failure. Thus, P2P architectures will be very important for future distributed systems and applications. In such systems, the computational burden of the system can be distributed among peer nodes of the system. Therefore, in decentralized systems users become themselves actors by sharing, contributing and controlling the resources of the system. These characteristics make P2P systems very interesting for the development of decentralized applications. In this thesis, we consider the P2P paradigm to build applications for Inter-Vehicular networks and robot control in wireless and wired environment. We implement and investigate the performance of different routing protocols in V2V scenarios using CAVENET simulation system, NS2 and NS3. We propose and experimentally evaluate the performance of the application of secure robot control using JXTA-overlay P2P platform. The experimental results show that JXTA-overlay can be successfully used to control the robot in a smoothly way. P2P systems allow decentralized data sharing by distributing data storage across all peers in a P2P network. But, peers can join and leave the system at any time so the shared data may become unavailable. To cope with problem, we propose and implement a fuzzy-based P2P system for data replication over the JXTA-overlay P2P network and evaluate its performance by computer simulations. Replication techniques assure the availability since the same data can be found at multiple peers. The simulation results show that the proposed system have a good behavior.

The contribution of our research work is as follows.

- We proposed and implemented a new simulation tool for Inter-Vehicular networks and integrate it with NS2 and NS3.
- Evaluation and comparison of different routing protocols in vehicular networks in highway and crossroad scenarios using CAVENET, NS2 and NS3.

- Implementation and evaluation of P2P platform based on JXTA technology for secure robot control.
- Implementation and evaluation of a fuzzy-based system for data replication in P2P systems.

The thesis is organized as follows. In the first Chapter is described the background, purpose and contribution of this study. In Chapter 2 is presented the current state and problems of P2P systems. In Chapter 3 are described mobility models and routing in Vehicular networks. In Chapter 4 is explained the structure of our simulation system for V2V networks. Chapter 5 introduces JXTA and JXTA-Overlay platform. In Chapter 6 are described the design and implementation of our proposed applications based on JXTA-Overlay. In Chapter 7 are presented simulation results for V2V communication for different routing protocols using NS2 and NS3. In Chapter 8 are discussed the experimental results for secure robot control and the simulation results for the fuzzy based data replication system. In Chapter 9 are presented the conclusions of our research and the future work.

Chapter 1

Introduction

1.1 Background

Peer to Peer (P2P) Networks have become very popular in recent times due to the great potential to the high scalability, robustness and fault tolerance because there is no centralized server and the network self-organizes itself. A lot of research efforts in the field of P2P have mainly focused towards strictly functionality issues such as scalability, efficient message propagation across the network or access to distributed resources.

In a P2P network, peers communicate directly with each other to exchange information. One particular example of this information exchange, that has been rather successful and has attracted considerable attention in the last years, is file sharing. These kind of systems are typically made up of millions of dynamic peers involved in the process of sharing and collaboration without relying in central authorities. P2P systems are characterized by being extremely decentralized and self-organized. These properties are essential in collaborative environments. The popularity and inherent features of these systems have motivated new research lines in the application of distributed P2P computing. New problems have also been posed, such as scalability, robustness and fault tolerance, organization and coordination, adaptability, distributed storage, location and retrieval, reputation, and security. In particular, security advances have focused on anonymity, access control, integrity, and availability.

P2P Systems benefit from high scalability and fault tolerance. Unfortunately in the context of security this is also one of the main disadvantages. Usually there are no central authorities for the verification and enforcement of policies. Since the management of policies is inevitable, new security concepts for P2P Systems are needed. Security is a

fundamental quality criterion in P2P Systems. Fulfilling certain security goals is a vital precondition for the preparation of P2P Technology for larger business applications. This work has been motivated by the need of having secure applications based on P2P systems.

P2P is a very good approach to build efficient platforms for vehicular communication and robot control. The improvement of the network technologies has provided the use of them in several different fields. One of the most emergent applications of them is the development of the Vehicular Networks in which the communications are among the nearby vehicles. Vehicular communication have recently emerged as a platform to support intelligent inter-vehicle communication to improve traffic safety.

Vehicle Networks are composed for a set of communicating vehicles equipped with wireless network devices that are able to interconnect each other without any pre-existing infrastructure. The most important network technology available nowadays for establishing vehicular networks is the IEEE 802.11b (Wi-Fi) standard, nevertheless new standards as IEEE 802.11p or IEEE 802.16 (WiMax) are promising.

The exchange of information among the vehicles provides a great opportunity for the development of new driver assistance systems. These systems will be able to disseminate and to gather real time information about the other vehicles and the road traffic and environmental conditions. Such data will be processed and analyzed to facilitate the driving by providing the user with useful information.

The road-constrained characteristics of these networks and the high mobility of the vehicles, their unbounded power source, and the emergence of roadside wireless infrastructures make vehicular networks a challenging and promising research topic.

Robots are being steadily introduced into modern every day life and are expected to play a key role in the near future. Typically, the robots are deployed in situations where it is too dangerous, expensive, tedious and complex for humans to operate. The successful introduction of robots in human environments will rely on the development of competent and practical systems that are dependable, safe, and easy to use.

In this thesis, we consider the P2P paradigm to build applications for Inter-Vehicular networks and robot control in wireless and wired environment. We implement and investigate the performance of different routing protocols in V2V scenarios using CAVENET simulation system, NS2 and NS3. We propose and experimentally evaluate the performance of the application of secure robot control using JXTA-overlay P2P platform. The experimental results show that JXTA-overlay can be successfully used to control the robot in a smoothly way. P2P systems allow decentralized data sharing by distributing data stor-

age across all peers in a P2P network. But, peers can join and leave the system at any time so the shared data may become unavailable. To cope with problem, we propose and implement a fuzzy-based P2P systems for data replication over the JXTA-overlay P2P network and evaluate its performance by computer simulations. Replication techniques assure the availability since the same data can be found at multiple peers. The simulation results show that the proposed system have a good behavior.

1.2 Thesis Purpose and Contribution

In this thesis, we propose and implement P2P platforms for vehicle networks and robot control. We evaluate the performance of different routing protocols in vehicular networks considering different realistic scenarios and different parameters such as speed, number of nodes, number of connections, traffic type and transmission rate. Our study aims to identify the routing protocol that performs better in vehicle networks.

At present time, the most of the P2P research field has pushed through problems related with security. Security starts to become one of the key issues when evaluating a P2P system. For this reason, we built a new secure application for robot control in P2P and in the best of our knowledge, there is not any security platform implemented in real P2P architectures.

Our contributions are summarized in the following.

- We proposed and implemented a new simulation tool for Inter-Vehicular networks and integrate it with NS2 and NS3.
- Evaluation and comparison of different routing protocols in vehicular networks in highway and crossroad scenarios using CAVENET, NS2 and NS3.
- Implementation and evaluation of P2P platform based on JXTA technology for secure robot control.
- Implementation and evaluation of a fuzzy-based system for data replication in P2P systems.

1.3 Thesis Outline

This thesis is organized into nine chapters and its structure is given in Fig. 1.1.

Chapter 1 serves as an introduction to the thesis and its content. It describes the background, purpose and contribution of this study and the outline of this thesis.

Chapter 2 of this thesis presents an introduction in P2P systems. It describes the characteristics, requirements, the current state and problems of P2P systems. This chapter also introduce data replication in P2P systems and different replication techniques to assure availability in case of peer failure.

Chapter 3 introduces vehicular networks and different mobility models of these networks. In this chapter, different routing protocols for vehicular networks are classified in three categories: unicast approach, multicast and geocast approach and broadcast approach. This chapter also describes in details the characteristics and functionality of AODV, OLSR, DYMO, DSR and DSDV routing protocols that are used in simulations.

In **Chapter 4** is given the related work on VANET simulators. The structure of our simulation system for Vehicular networks called CAVENET is also described in this chapter.

Chapter 5 introduces JXTA components and JXTA-Overlay P2P distributed platform which we have used to develop our application for secure robot control.

In **Chapter 6** are described the design and implementation of our proposed applications based on JXTA-Overlay.

In **Chapter 7**, we give evaluation results for V2V communication for different scenarios using CAVENET, NS2 and NS3. We evaluate and compare the performance of different routing protocols using different metrics.

In **Chapter 8** are discussed the experimental results for secure robot control and the simulation results for fuzzy-based data replication system.

Chapter 9 concludes this thesis. The conclusions of our research and the future work are given in this chapter.

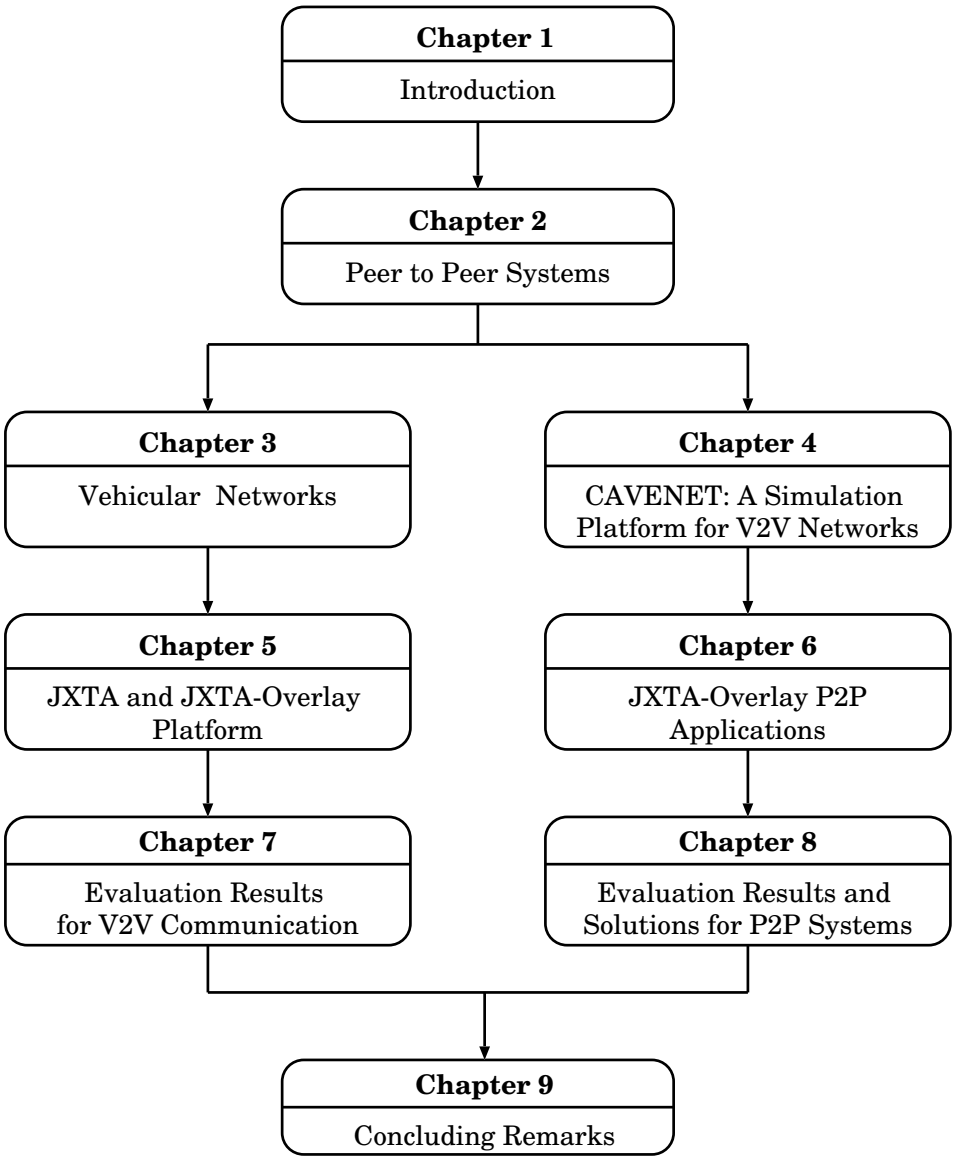


Figure 1.1: Thesis Structure.

Chapter 2

Peer to Peer Systems

2.1 Introduction

A P2P system is a self organizing system of equal and autonomous entities (peers) which aims for the shared usage of distributed resources in networked environment avoiding central services [1]. A P2P system is a network of peers that communicate with each other. A peer is an entity in the system, usually an application running on a device, or the user of such an application. All peers should be of equivalent importance to the system, no single peer should be critical to the functionality of the system.
Some of the essential features of P2P systems are:

- The peers should have autonomy, i.e. be able to decide services they wish to offer to other peers.
- Peers should be assumed to have temporary network addresses. They should be recognized and reachable even if their network address has changed.
- A peer can join and leave the system at its own disposal.

2.2 P2P Architectures

In the computer system taxonomy, P2P is in the category of distributed systems. From the architecture perspective, we can distinguish: “Pure” P2P systems, which are completely autonomous from any central control or component, and the “Hybrid” ones, which include some form of centralization, even for non-critical tasks. More commonly, P2P systems are

classified according to their purpose into distributed computing, file sharing, collaboration and platforms. These systems are briefly described with their representative applications.

2.2.1 Distributed Computing

This type of P2P system is aimed at solving complex computing problems by breaking them into smaller tasks and executing them in parallel on a number of peers. Such computing problems include market analysis, code breaking, searching for extraterrestrial life or bio-informatics (e.g. cure for cancer). The main representative of this category is SETI@home, a project aimed at analyzing radio signals from outer space in search for extraterrestrial life. This is a hybrid P2P system in which participating peers connect to the central server periodically to obtain the computing task and deliver results. The peers share processing power, but they do not interconnect [2].

2.2.2 File Sharing and Content Storage

File sharing P2P systems allow users to locate the files of interest according to their name and offer their own files to others. The files are classified as music, video, image, etc. The major feature of this type of system is that it allows quick location of files and a way to obtain copyrighted content at no cost. The file sharing started with the centralized Napster file sharing service, which was a server farm offering a directory of music files. Its purpose was to allow peers to find other peers that have the song of interest and then connect directly to one of them for download. Gnutella followed with a pure P2P architecture, creating a complex interconnected overlay network, with major clients being BearShare and LimeWire. The FastTrack network arrived next with Kazaa, the client that becames synonymous with the network itself, using super peers as small hubs that offer directory services for all peers connected to it. This approach facilitates much faster search and Gnutella network soon followed this model. Freenet is a system where anonymous peers provide information storage to each other, with anonymity being the primary feature.

2.2.3 Collaboration

Collaboration is the most user-involved type of activity that can be supported by a P2P approach. Virtual workspaces allow users to stay at their workstations, but at the same time collaborate and interact in real time without moving to a conference room. The P2P

approach enables shared workspaces without central servers and databases or any specific infrastructure. The simplest form of P2P collaboration is instant messaging, popularized by Jabber, AOL AIM, MSN Messenger, Yahoo! and ICQ messaging applications. Groove adds message board, file repository, calendar and custom modules in its collaborative application.

2.2.4 Platforms

The only "true" platform existent is JXTA, whose goal is to formulate standards and provide the P2P infrastructure. Vendors of other P2P solutions, such as Groove and Jabber, provide software development kits for creating customized applications, but they are still based on the underlying product. A P2P platform must provide the infrastructure for peer connectivity, messaging and organization. In some sense, .NET My Services can also be classified in this category, since it provides web service discovery using UDDI, service description with WSDL, and the development framework. However, web services strongly rely on centralized UDDI servers and organization in peer groups is not supported.

2.3 Goals, Benefits and Problems in P2P Networks

2.3.1 Goals of P2P Systems

Cost reduction: By reducing responsibilities and tasks of individual servers, can be reduced the need for costly centralized servers.
Eliminating single points of failure: The traditional client/server model is dependent on a static core. If the core fails, the whole system collapses. By distributing responsibility over a set of participants, the importance of any one participant is reduced.
Autonomy: Participants in the system are able to decide what resources should be available within the system.
Improved scalability: By distributing responsibility and tasks over a set of participants, resource usage can be distributed more evenly, typically resulting in more scalable systems.
Resource aggregation: Each participant contributes with some resource. By aggregating resources, resource demanding tasks can be performed.
Anonymity: In client/server systems, the server is able to keep track of actions of participants in the system. By distributing responsibility, various methods can be employed to

maintain the anonymity of peers.

Dynamism: Participants should be able to join and leave the system at every moment, and the overall productivity of the system should not be dependent on any single participant.

2.3.2 Benefits of P2P Systems

Use of the previously unused resources: On home and office computers, processing cycles are wasted constantly while the computer is on but underutilized/idle (generally overnight and during non-business hours). The disk storage is typically underutilized, as these computers are used mostly for simple non-intensive tasks. The P2P-based application can make use of these resources, thereby increasing utilization of an already paid resource.

Potential to scale: The resources of the server or server-cluster limit the capabilities of the client-server system. As the number of clients increases, it becomes very difficult to keep up with demand and maintain the performance and service at the required level. By distributing demand and load on the shared resources, the bottlenecks can be eliminated and a more reliable system achieved [3].

Self-organization: P2P systems build and organize themselves. Each peer dynamically discover other peers and build the network. They organize according to their preferences and current conditions within the peer group. For example, P2P systems such as file sharing networks offer a choice of file providers for download. If a popular peer is overloaded and poor performance occurs, consumer peers can switch to another provider, effectively re-balancing load and changing the network topology [4].

Increased autonomy and anonymity: In P2P systems, peers are all of equal status. They can autonomously decide when, for how long to participate and how much of their resources to share. Similarly, it is possible to achieve higher anonymity and privacy simply by having no central authority that can keep track of the activity in the system.

Cost distribution and reduction: This benefit actually comes through the achievement of all other goals. The cost of a powerful server or cluster can be avoided by distributing processing tasks over numerous low-powered computers. By letting the system self-organize into a large peer group, there is no need for central administration and maintenance, which reduces cost as well.

2.3.3 Problems of P2P Systems

The P2P also raises many concerns, mainly about security, manageability and performance. Several security aspects need to be considered before adopting a P2P solution for the system. In respect to data, once it is released into the peer community, it is almost impossible to control it. While it is possible to protect the integrity of the data with digital digests and hash values, it may not be possible to control copying or deletion of an object. **Secure communication** is another problem. Most of the P2P systems use insecure protocols for communication and among numerous participating peers it may be difficult to recognize the malicious ones. Another concern is **manageability of the system**. If any kind of control or tracking is required, P2P may not be the right way to go. Financial transactions are a prime example where regardless of the benefits of P2P, all involved parties would likely prefer a centralized system with strong authentication and tracking capabilities. Finally, achieving good performance is not trivial in P2P systems, although it is a major goal. The first steps in improving performance are distributing processing load and aggregating resources, such as processing power and storage from individual peers. For distributed computing, these steps may be enough, but in file sharing and storage systems, the amount of network traffic is a significant factor. In any P2P system where searching for resources is the core function, a search engine determines the performance and scalability of the system.

2.4 Data Replication in P2P Systems

P2P systems allow decentralized data sharing by distributing data storage across all peers in a P2P network. But these peers can join and leave the system at any time so the shared data may become unavailable. To cope with problem, P2P systems replicate data over the P2P network. Since the same data can be found at multiple peers, availability is assured in case of peer failure. In [5], the authors classify data replication techniques in three groups. We briefly describe them below.

2.4.1 Single-master vs Multi-master

In the Single-master approach, there is only a single primary copy for each replicated object. The single-master allows only one site to have full control over the replica (read and write rights) while the other sites can only have a read right over the replica. Advantage

of this model is the centralization of the updates at a single copy, simplifying the concurrency control. The disadvantage of this model is a single point of failure that can limit the data availability.

In the Multi-master approach, multiple sites holds primary copy of the same object. All these copies can concurrently updated. Multiple sites can modify their saved replicas. This approach is more flexible than single-master because in the case of one master failure, other masters can manage the replicas.

2.4.2 Full Replication vs. Partial Replication

There are two basic approaches for replica placement: full replication and partial replication. Full replication takes place when each participating site stores a copy of every shared object. Every site should have the same memory capacities in order to replace any other site in case of failure.

In partial replication, each site holds a copy of a subset of shared objects so the sites can take different replica objects. This approach requires less storage space because updates are propagated only towards the affected sites. But this approach limits load balance possibilities because certain sites are not able to execute a particular type transaction [6]. In partial replication, it is very important to find a right replication factor.

2.4.3 Synchronous vs. Asynchronous

In Synchronous replication, the node that initiates the transaction (set of update operations) propagates the update operations within the context of the transaction to all the other replicas before committing the transaction. There are several algorithms and protocols to achieve this behaviour [7, 8]. Synchronous propagation enforces mutual consistency among replicas. In [7] authors define this consistency criteria as one-copy-serializability. The main advantage of synchronous propagation is to avoid divergences among replicas. The drawback is that the transaction has to update all the replicas before committing.

The asynchronous approach do not change all replicas within the context of the transaction that initiates the updates. An advantage of asynchronous propagation is that the update does not block due to unavailable replicas, which improves data availability. The asynchronous replication technique can be classified as optimistic or non-optimistic in terms of conflicting updating [9, 10].

Chapter 3

Vehicular Networks

3.1 Introduction

Vehicular networks have recently emerged as a platform to support intelligent inter-vehicle communication to improve traffic safety. The road-constrained characteristics of these networks and the high mobility of the vehicles, their unbounded power source, and the emergence of roadside wireless infrastructures make vehicular networks a challenging and promising research topic.

Vehicular networks can be seen as an effort to combat real-life transportation problems such as accidents, traffic jams, fuel consumption and pollutant emissions.

Advances in wireless, global positioning systems and sensor technologies serve as the ground work for the development of innovative Vehicle to Vehicle (V2V) and Vehicle to Roadside (V2R) applications and services with great potential for improving the safety and comfort (see Fig. 3.1).

Due to the high cost of deploying and implementing vehicular systems in a real environment, most of research is concentrated on simulations.

3.2 Vehicular Traffic Models

The Random Waypoint (RW) model has been the earliest mobility model for ad-hoc networks. Basically, in RW every node picks up a random destination and a random velocity at certain points called waypoints. This model has been extended in a number of ways in order to take into account more realistic movements.

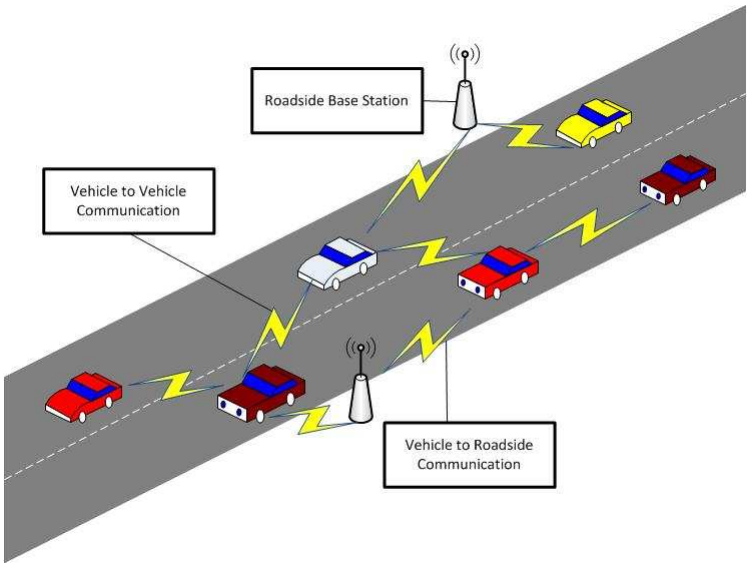


Figure 3.1: Communication in vehicular networks.

The simulation of such models has shown the problem of velocity decay. That means that the simulation variable slowly decays towards a steady state value as the simulation time proceeds. This is problematic, because we do not know when this transient ends. Consequently, we do not know precisely how to remove the transient values. The root of this phenomenon has to be attributed to the underlying mobility model, which has been assumed random. Every node randomly picks a velocity from a continuous uniformly distributed random variable between $[v_{\min}, v_{\max}]$. The velocity is changed at particular points called waypoints. In this way, the system has an infinite (but countable) number of states. The general solution to this problem consists in finding the steady state distribution of the simulation variable and let the system starts with that distribution.

The problem has been solved by several authors, in particular by Le Boudec [11], who used Palm distributions, and Noble [12]. However, all mobility models considered so far are Short Range Dependent (SRD). This means that every mobile chooses its velocity independently by the others.

In the particular case of deterministic traffic models, the average velocity is SRD and the transient state depends on the density of the vehicles. In general, the mobility model of VANETs for the simulated variable of interest (e.g. the average velocity) can be Long Range Dependent (LRD) in some cases. This fact poses some problems on how long the simulation should be and how many samples from the starting time should be discarded.

Random Waypoint model, the Random Walk model, the Random Direction model, the Reference Point Group (or Platoon) model, the Node Following mode, the Gauss-Markov

model, all involved generation of random linear speed-constant movements within the topology boundaries. So, using this models would produce completely useless results.

The Random Waypoint City Model [13] approach combines aspects of the Random Waypoint Mobility Model with the vector street maps. It has a high granularity as exact user locations are available. User movement is independent of other users and past trips, so that individual homes and workplaces of users are not modeled.

Cellular Automaton (CA) [14] is a discrete time model of the vehicular traffic. It is a stochastic CA model based on some pertinent rules. It is defined on a one-dimensional array of L sites and with open or periodic boundary conditions. It contains important aspects of fluid-dynamical approach to traffic flow such as the transition from laminar to start-stop traffic in a natural way. This model does not consider the effect of random acceleration and deceleration on the traffic flow.

STRAW (STreet RAndom Waypoint) [15], is another mobility model for VANETs that constrains node movement to streets defined by map data for real US cities and limits their mobility according to vehicular congestion and simplified traffic control mechanisms. This mobility model provides reasonable runtimes and memory consumption that scales fairly well with the size of the simulation. But this model did not take into consideration the lane changing.

Manhattan model is a generated-map-based model introduced in [16] to simulate a urban environment. The simulation area is represented by a map (generated before the simulation start) containing vertical and horizontal roads made up of two lanes, allowing the motion in the two directions (north-south for the vertical roads and east-west for the horizontal ones). Contrary to the freeway model, a vehicle can change a lane when it passes a crossroads with absolutely no control mechanism, thus continuing their movements without stopping. This makes this model unrealistic.

In City Section Mobility Model, the simulation area is a street network that represents a section of a city where the ad hoc network exists. The streets and speed limits are based on the type of city being simulated. This model can be seen as a hybrid model between RWP and Manhattan, as it introduces the principle of RWP like the pause-time and random selection destination, within a generated-map-based area. This model is only for small simulation area.

Stop Sign Model [17] is the first model that integrates a traffic control mechanism. This model is based on real maps of the TIGER/Lines database, but all roads are assigned a single lane in each direction and a vehicle should never overtake its successor. In the

Traffic Sign Model model, stop signals are replaced by traffic lights. A vehicle stops at a crossroads when it encounters a red stoplight, otherwise it continues its movement. When the first vehicle reaches the intersection, the light is randomly turned red with probability p (thus turned green with probability $1 - p$). If it turns red it remains so for a random delay (pause-time), forcing the vehicle to stop as well as the ones behind it.

To determine the mobility patterns of vehicles, geographic, sociological and external factors should be taken in consideration.

A realistic mobility model should include:

- **Number of Lanes and Their Directions:** From the point of view of protocol operations, these parameters can affect the connectivity of the network. Also intersection of lanes affect the traffic behaviour on the whole lane, because the crosspoint is the bottleneck for the lane. The traffic pattern differs depending on the kind of road on which the vehicles are passing: rural road, urban road, city road, highway. Also acceleration and deceleration of vehicles should be considered.

- **Obstacles:** The mobility model should take in consideration obstacles of mobility and wireless communication.

- **Traffic and Weather Conditions:** Traffic density is not uniformly spread around the day. A heterogeneous traffic density is always observed at some peak time of days, such as rush hours, weekends, holidays or special events also in bad weather condition and unexpected situations.

- **Drivers Behaviour:** Drivers interact with their environments, not only with respect to static obstacles, but also to dynamic obstacles, such as neighboring cars and pedestrians. Accordingly, the mobility model should control vehicles mutual interactions such as overtaking, traffic jam, preferred paths, or preventive action when confronted to pedestrians.

3.3 Routing Protocol Types

Routing Protocols for vehicular communication need to take into account the nature of node movements, vehicle mobility, which is higher than in traditional ad-hoc networks, direction of vehicle movement and direction of information movement. The end-to-end delay is another important issue due to emergency situations since wireless channels have

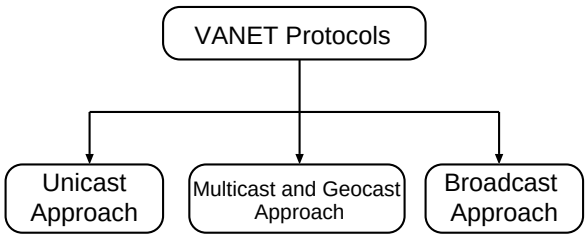


Figure 3.2: Taxonomy of routing protocols for vehicular networks.

unique problems like limited bandwidth, location-dependent channel errors and varying channel capacity.

Routing in VANETs has been studied recently and many protocols are proposed. We classify this protocols based on the routing category in:

- Unicast Approach
- Multicast and Geocast Approach
- Broadcast Approach

The taxonomy of VANET protocols is shown in Fig. 3.2.

3.3.1 Unicast Approach

Unicast protocols provide information delivery between two nodes via multiple wireless hops. Some VANET comfort applications (including Internet connectivity, multi-media access, inter-vehicle communications) use unicast routing. There are many unicast protocols proposed for VANETs. Anchor-based Street and Traffic Aware Routing (A-STAR) adopts the anchor-based routing approach with street awareness. A-STAR is proposed for metropolis vehicular communications. A-STAR features the novel use of city bus route information to identify anchor paths of higher connectivity so that more packets can be delivered to their destinations successfully. But the traffic awareness in A-STAR shall be extended to include data traffic to provide vehicular nodes with higher performance paths in terms of connectivity as well as delay.

Mobility-Centric Data Dissemination Algorithm (MDDV) [18], is a mobility-centric approach for data dissemination in vehicular networks designed to operate efficiently and reliably despite the highly mobile, partitioned nature of these networks. MDDV is designed to exploit vehicle mobility for data dissemination, and combines the idea of opportunistic forwarding, trajectory based forwarding and geographical forwarding. Since

no end-to-end connectivity is assumed, intermediate vehicles must buffer and forward messages opportunistically. As an opportunistic algorithm, MDDV answers the questions about who can transmit, when to transmit, and when to store/drop messages. Using a generic mobile computing approach, vehicles perform local operations based on their own knowledge while their collective behavior achieves a global objective. Message delivery reliability is improved by allowing multiple vehicles to actively propagate the message.

Vehicle-assisted Data Delivery (VADD) [19] protocol adopted the idea of carry-and-forward for data delivery from a moving vehicle to a static destination. The most important issue is to select a forwarding path with the smallest packet delivery delay. To keep the low data transmission delay, VADD protocol transmits packets through wireless channels as much as possible, and if the packet has to be carried through roads, the road with higher speed is chosen firstly. VADD protocol assumes that vehicles are equipped with pre-loaded digital maps, which provide street-level map and traffic statistics such as traffic density and vehicle speed on roads at different times of the day. According to the information provided by digital maps, VADD protocol proposed a delay model to estimate the data delivery delay in different roads.

Position Based Multi-hop Broadcast (PMBP) [20] is a protocol for emergency message dissemination in inter-vehicle communications. The highlights of this scheme includes: 1) by a cross layer approach, the current relaying node selects the neighboring node with the farthest distance from the source node in the message propagation direction as the next relaying node, which ensures emergency messages can be delivered to remote nodes with the least time latency; 2) At each hop, the emergency message is only broadcasted once, therefore, redundant broadcast messages are greatly reduced; 3) by adopting revised RTS/CTS handshake, there is no hidden terminal problem in PMBP, and it ensures every node could correctly receive the emergency message, which makes the scheme more reliable; and 4) the emergency message has the highest priority to access the channel, and it guarantees the emergency message be broadcasted as soon as possible.

3.3.2 Multicast and Geocast Approach

Geocast routing, is basically a location-based multicast routing. The objective of a geocast routing is to deliver the packet from a source node to all other nodes with a specified geographical region (Zone of Relevance, ZOR).

Distributed Robust Geocast (DRG) and RObust VEhicular Routing (ROVER) are geocast routing protocols for vehicular ad hoc networks. Distributed Robust Geocast (DRG) [21], is a geocast protocol that is completely distributed, without control overhead and state information. The Distributed Robust Geocast (DRG) protocol is developed for applications that require a fast and reliable transmission, though without any end-to-end QoS requirements. The protocol uses geographical addressing to form a multicast tree within a zone of relevance. The tree is formed on-demand and can be used to forward multiple data packets from the same source. Therefore, it can be used by a reliable transport protocol to ensure end-to-end QoS. Each vehicle is assumed to have a unique Vehicle Identification Number (VIN). Also, the vehicles are assumed to have a GPS receiver and access to a digital map. The objective of the protocol is to transmit a message, M, from an application, A, to all other vehicles within an application-specified ZOR, Z. The ZOR is defined as a rectangle specified by its corner coordinates. Thus, a message is defined by the triplet [A, M, Z]. When a vehicle receives a message, it accepts the message if, at the time of the reception, it is within the ZOR. Similar to geocasting protocols authors also define a Zone Of Forwarding (ZOF) as a zone including the source and the ZOR. All vehicles in the ZOF are part of the routing process, although only vehicles in the ZOR deliver the message to their corresponding application layer (specified by A).

DG-CastoR (Direction-based GeoCast Routing) [22], is a geocast routing protocol for Vehicular Ad hoc NETworks. It creates a virtual community of nodes able to communicate during a certain period of time. The Rendezvous geocast region is created based on the prediction of future locations to estimate the link availability between mobile nodes. The main advantage of DG-CastoR protocol is that it reduces the network congestion by avoiding the unnecessary packets transmission on the whole network.

3.3.3 Broadcast Approach

Broadcast is an effective approach for safety-related information exchange such as emergency accident, traffic information services, announcements and advertisement, to achieve cooperative driving in VANET. However, it suffers from several fundamental challenges such as message redundancy, link unreliability, hidden terminal and broadcast storm, that degrade the efficiency of the network.

UMB (Urban Multi-Hop Broadcast Protocol for Inter-Vehicle Communication Systems) [23] is designed to operate without exchanging location information among neigh-

boring nodes. This protocol has a mechanism for decrease the effect of hidden nodes and avoid collisions. However, usage of omni-directional antennas suffer from several drawbacks: redundant traffic, small coverage in sparse network, contention and collision especially at the intersection.

The Dedicated Omni-purpose inter-vehicle communication Linkage Protocol for HIghway automatioN (DOLPHIN) [24] implements a variation of flooding where information dissemination is done in the reverse direction of vehicle movement. In particular the nodes that broadcast the information are reselected in every communication hop, being in the rear position of the previous one. DOLPHIN, specifically is designed for cooperative driving, that is a message goes through all the vehicles in the highway. As a result it cannot be considered as a general purpose VANET Protocol.

Another VANET Protocol is the GPS-Based Message Broadcasting [25]. This protocol is based on a broadcast algorithm similar to the single cast routing protocol, Zone Routing Protocol (ZRP) and it is shown to perform much better than the flooding based ones. However, since the forwarding nodes are selected in every hop, it still has routing overhead. Furthermore, since it is a broadcast protocol, it is not well suited for the point-to-point communications.

BROADCOMM protocol [26] is a routing protocol that improves the quality of emergency broadcast communication in highways. It is based on Geographical Routing and has a hierarchical structure. The nodes are organized in two level of hierarchy: in the first level, nodes in the same cell can communicate with each-other and also with nodes of neighboring cell and the second level are cell reflectors that are a few nodes usually located closed to the geographical center of cell and can communicate with cell members or members of neighboring cells that are in the communication range of the cell reflector. This protocol outperforms similar flooding based routing protocols in the message broadcasting delay and routing overhead. It makes possible a real time communication between distant vehicles and group member vehicles. However this protocol works only in simple highway networks.

Distributed Vehicular Broadcast (DV-CAST) [27] is designed for safety and transport efficiency applications in VANET. It uses a per-hop routing based approach which uses only local connectivity information to make a routing decision. This protocol uses local connectivity to ensure the maximum reachability of the broadcast message. The designed protocol addresses how to deal with extreme situations such as dense traffic conditions during rush hours, sparse traffic during certain hours of the day and low market

Table 3.1: Comparision of VANET routing protocols.

Protocols	Routing Mechanism	Specific Usage	Drawbacks
AODV	Unicast	Performance evaluation in urban environments	Decrease the packet delivery ratio
OLSR	Broadcast	Performance evaluation in urban environments	Decrease the packet delivery ratio
DSR	Unicast	Compare the performance with other VANET specific protocols	Decrease the packet delivery ratio
GPRS	Unicast	Compare the performance with other VANET specific protocols	Low packet delivery ratio
A-STAR	Unicast	Routing in city environment	Routing paths are not optimal and cause large delay of packet transmission
MDDV	Unicast	Data dissemination	Large delay if the traffic density varies by time
VADD	Unicast	QoS routing protocol for VANET	Large delay due to varying topology and varying traffic density
PMBP	Unicast	Emergency messages exchange	Higher delay than simple flooding
DRG	Geocast	Fast communication across a large area	Unsuitable for safety data dissemination in highly dynamic environments
ROVER	Geocast	Can be used by a reliable transport protocol to ensure end-to-end QoS	Do not consider the size of data that can be transmitted
DG-CastoR	Geocast	Reduce network conggestion by avoiding transmission of unnesesary packets on the whole network	Neighbors may hinder the proper forwarding of messages
UMB	Geocast	Decrease the effect of hidden nodes and avoid collisions	Usage of omni-directional antennas cause redundant traffic and small coverage
DOLPHIN	Broadcast	Cooperative driving	Long communication delays and high network loads
GPS-based Message Broadcasting	Broadcast	Use GPS information to enhance the performance of broadcast service in VANET	Not well suited for point to point communication
BROADCOMM	Broadcast	Emergency broadcast communication in highways	Works only in simple highway networks
DV-CAST	Broadcast	Is designed for safety and transport efficiency applications	Assume that each vehicle can accurately detect the local connectivity

penetration rate of cars using Dedicated Short Range Communication technology. But, DV-CAST protocol design assumes that each vehicle can accurately detect the local connectivity, in fact in a real VANET this assumption may not be always valid as there could be many uncontrollable factors that could cause the neighbor detection mechanism to fail.

As each node receives and broadcasts the message almost at the same time, it causes contention and collisions, broadcast storms and high bandwidth consumption. In Table 3.1 is shown the comparison between VANET routing protocols. From the analysis of the existing routing protocols, we conclude that VANETs protocols should have the following features.

- **Low Latency:** Low end-to-end network delay, which meets application requirements.
- **High Reliability:** A major challenge in protocol design in VANET is to improve reliability of protocols and to reduce delivery delay time and the number of packet retransmission.
- **Scalability:** Ability to uphold service requirements over a range of vehicle densities, receiver count, and network diameters.
- **Driver Behavior:** Driver behavior should be considered for designing of delay bounded routing protocols since carry and forward is the mainly approach to deliver packets.
- **Comfort Messages:** Geocast routing for comfort applications should also considered. Comfort messages are usually tolerant of delay, Network bandwidth is generally reserved for emergency messages. It is worth to develop an efficient geo cast routing protocol for comfort applications with delay tolerant capabilities with low bandwidth utilization.
- **Hierarchical Routing:** For a hierarchical routing algorithm the size of routing tables will be much smaller as compared to the link state type of routing. Furthermore, the routing overhead and latency in hierarchical routing will be smaller. However hierarchical routing has drawbacks, including the need to maintain longer (hierarchical) addresses and the cost of continuously updating the cluster hierarchy and the hierarchical addresses as the nodes move.

3.4 Routing Protocols Description

Since VANETs are a specific class of ad-hoc networks, the commonly used ad-hoc routing protocols initially implemented for MANETs have been tested and evaluated for use in a VANET environment.

VANETs share some common characteristics with MANETs. They are both characterized by the movement and self organization of the nodes. In the following, we will describe AODV, OLSR, DYMO, DSDV and DSR, which are protocols used in this work.

AODV

The AODV [28] is an improvement of DSDV to on-demand scheme. It minimize the broadcast packet by creating route only when needed. Every node in network maintains the route information table and participate in routing table exchange. When source node wants to send data to the destination node, it first initiates route discovery process. In this process, source node broadcasts Route Request (RREQ) packet to its neighbours. Neighbour nodes which receive RREQ forward the packets to its neighbour nodes. This process continues until RREQ reach to the destination or the node who know the path to destination.

When the intermediate nodes receive RREQ, they record in their tables the address of neighbours, thereby establishing a reverse path. When the node which knows the path to destination or destination node itself receives RREQ, it sends back Route Reply (RREP) packet to source node. This RREP packet is transmitted by using reverse path. When the source node receives RREP packet, it can know the path to destination node and it stores the discovered path information in its route table. This is the end of route discovery process. Then, AODV performs route maintenance process. In route maintenance process, each node periodically transmits a Hello message to detect link breakage.

OLSR

The OLSR [29] protocol is a pro-active routing protocol, which builds up a route for data transmission by maintaining a routing table inside every node of the network. The routing table is computed upon the knowledge of topology information, which is exchanged by means of Topology Control (TC) packets.

OLSR makes use of `HELLO` messages to find its one hop neighbours and its two hop neighbours through their responses. The sender can then select its Multi Point Relays (MPR) based on the one hop node which offer the best routes to the two hop nodes. By this way, the amount of control traffic can be reduced. Each node has also an MPR selector set which enumerates nodes that have selected it as an MPR node. OLSR uses TC messages along with MPR forwarding to disseminate neighbour information throughout the network. Host Network Address (HNA) messages are used by OLSR to disseminate network route advertisements in the same way TC messages advertise host routes.

DYMO

DYMO [30] is a new reactive (on demand) routing protocol. DYMO builds upon experience with previous approaches to reactive routing, especially with the routing protocol AODV. It aims at a somewhat simpler design, helping to reduce the system requirements of participating nodes, and simplifying the protocol implementation. DYMO retains proven mechanisms of previously explored routing protocols like the use of sequence numbers to enforce loop freedom. At the same time, DYMO provides enhanced features, such as covering possible MANET-Internet gateway scenarios and implementing path accumulation.

Besides route information about a requested target, a node will also receive information about all intermediate nodes of a newly discovered path. There is a major difference between DYMO and AODV. AODV only generates route table entries for the destination node and the next hop, while DYMO stores routes for each intermediate hop. To efficiently deal with highly dynamic scenarios, links on known routes may be actively monitored, e.g. by using the MANET Neighbourhood Discovery Protocol or by examining feedback obtained from the data link layer. Detected link failures are made known to the MANET by sending a route error message (RERR) to all nodes in range, informing them of all routes that now became unavailable. Should this RERR in turn invalidate any routes known to these nodes, they will again inform all their neighbours by multicasting a RERR containing the routes concerned, thus effectively flooding information about a link breakage through the MANET.

DSDV

DSDV protocol is a proactive routing protocol which is a modification of conventional Bellman-Ford routing algorithm. In DSDV, each node maintains an entry of the table contains the address identifier of a destination, the shortest known distance metric to that destination measured in hop counts and the address identifier of the node that is the first hop on the shortest path to the destination. A sequence number is also associated with each route to the destination. The route labeled with the highest sequence number is always used. The routing table updates can be sent in two ways: a full dump or an incremental update. A full dump sends the full routing table to the neighbours and could span many packets whereas in an incremental update only those entries from the routing table are sent that have a metric change since the last update and it must fit in a packet.

When the network is relatively stable, incremental updates are sent to avoid extra traffic and full dump are relatively infrequent. In a fast changing network, incremental packets can grow big so full dumps will be more frequent.

DSR

DSR is an on-demand routing protocol designed specifically for use in multi-hop wireless ad-hoc networks of mobile nodes. It allows the network to be completely self-organizing and self-configuring and does not need any existing network infrastructure or administration.

The DSR protocol uses two main mechanisms: route discovery and route maintenance which operate entirely “on demand” and work together to allow nodes to discover and maintain source routes to arbitrary destinations in the network.

- Route Discovery
  DSR is an on-demand routing protocol, so it looks up the routing during transmission of a packet. At the first phase, the transmitting node searches its route cache to see whether there is a valid destination exists and if so, then the node starts transmitting to the destination node and the route discovery process end here. If there is no destination address, then the node broadcasts the route request packet to reach the destination. When the destination node gets this packet, it returns the learned path to the source node.

- Route Maintenance
  It is a process of broadcasting a message by a node to all other nodes informing the network or node failure in a network. It provides an early detection of node or link failure since wireless networks utilize hop-to-hop acknowledge.

DSR does not use periodic routing advertisements, thereby saving bandwidth and reducing power consumption.

Chapter 4

CAVENET: A Simulation Platform for V2V Networks

4.1 Related Work on VANET Simulators

In the recent years, a lot of simulators for VANETs have been emerging [31]. In VANETs, three types of simulators are developed:

- Mobility Simulators (software environments that generate vehicle movement in trace files).
- Network Simulators (used to test the performance of networking protocols).
- Integrated Simulators (integrate the traffic simulator and network simulator).

4.1.1 Mobility Simulators

The IMPORTANT framework has been one of the first attempt to understand the dependence between vehicular traffic and communication performance [16, 32]. The authors analyzed the impact of the node mobility on the duration of communication paths. However, the author implemented the code in C, which is difficult to debug and extend without the support of a detailed documentation. Moreover, it seems that the Freeway model that they use is not very realistic.

In [33], the authors present VanetMobiSim simulator written in Java, which can generate mobility traces in several formats. It is an open source vehicular mobility generator tool based on CanuMobiSim which has been extended to achieve realistic simulations of

vehicular mobility. But the details of the implementation are not open. There are also other powerful mobility simulators, like TranSim [34], which makes use of a cellular automaton for simulating the interaction of vehicles. Unfortunately, the code is not conceived for network protocols simulation, and the software is commercially licensed. Also, SUMO is another powerful mobility simulator, intended for traffic planning and road design optimization. It is written in C++. MOVE [35] is an extension to SUMO that adds a GUI for describing maps and defining vehicle movement and allows the user to import Google Earth maps. MOVE also includes a visualization tool that allows users to view the generated mobility trace.

4.1.2 Network Simulators

NS-2 [36] is an open-source discrete event network simulator that supports both wired and wireless networks, including many MANET routing protocols and an implementation of the IEEE 802.11 MAC layer. It is the most widely used simulator for academic networking research. The core of NS-2 is written in C++ and users interact with NS-2 by writing TCL scripts.

OPNET is another network simulator used for simulations of both wired and wireless networks but it is commercial. OMNeT++ is an object-oriented modular discrete event network simulator. OMNeT++ has a component-based design, new features and protocols can be supported through modules. OMNeT++ supports network and mobility models through the independently developed Mobility Framework and INET Framework modules.

The NS3 [37] simulator is developed and distributed completely in the C++ programming language, because it better facilitated the inclusion of C-based implementation code. The NS3 architecture is similar to Linux computers, with internal interface and application interfaces such as network interfaces, device drivers and sockets. The goals of NS3 are set very high: to create a new network simulator aligned with modern research needs and develop it in an open source community.

4.1.3 Integrated Simulators

Integrated simulators usually consist of two sub-simulators: mobility simulator and network simulator which communicate with each other like shown in Fig. 4.1. This sim-

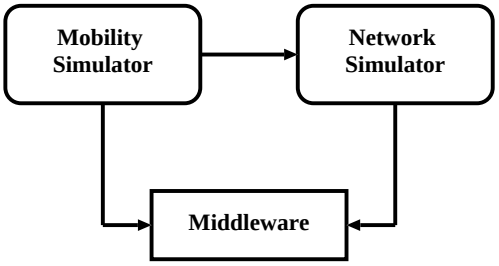


Figure 4.1: The structure of integrated simulators.

ulators offers a high level of maturity in both areas: mobility simulation and network simulation.

GrooveNet [38] is an integrated simulator that supports multiple models that characterize communication, travel and traffic control to enable large scale simulations in street maps of any US city. The current limitations are that map database does not indicate one-way streets and the altitude of the street. GrooveNet is implemented in C++ and Qt graphics cross-platform library in Linux. GrooveNet is based on the US Census Bureaus TIGER/Line 2000+ database format and is able to dynamically load counties at run-time. On startup GrooveNet reads map database text files and converts the topology data into a binary encoded file with a graph structure.

SWANS++ extends the network simulator SWANS by adding a GUI to visualize the scenario and a mobility model, STRAW for the vehicles movement in street scenarios. STRAW uses the simple RW mobility model, but it restricts the vehicles movement to real street boundaries, loaded from TIGER/Line data files. The mobility model implemented in this simulator does not support lane changing and also it does not provide feedback between the mobility and networking modules.

TraNS (Traffic and Network Simulation Environment) is a GUI tool that integrates traffic and network simulators to generate realistic simulations of Vehicular Ad hoc NETworks (VANETs). TraNS allows the information exchanged in a VANET to influence the vehicle behavior in the mobility model. There is an attempt to interface SUMO with NS2 [39]. However, in our opinion, it is very expensive to understand the SUMO language and also unnecessary, because the communications engineer needs only a parsimonious model, easy to extend and/or modify.

NCTUns 1.0 was developed only as a network simulator, but the most recent version, integrates some traffic simulation capabilities, such as designing maps and controlling vehicles mobility. A large variety of maps can be designed using different types of supported road segments. The best feature available in NCTUns is that its network protocol stacks

Table 4.1: VANET integrated simulators.

Name of the Simulator	GrooveNet	SWANS++	TraNS	CAVENET	NCTUns
Open source	Yes	Yes	Yes	No	Yes
Language	C	Java	SUMO	MATLAB	Java
Modular	Yes	Yes	Yes	Yes	Yes
Mobility Simulator	GrooveNet	JiST/SWANS	SUMO	MATLAB	Tightly integrated with network simulator
Network Simulator	Supports multiple models	SWANS	NS2	NS2 & NS3	Tightly integrated with mobility simulator
Mobility Model	Many	Many	Many	1-dimensional CA	Many

includes the Linux kernel protocol stack, including TCP/IP and UDP/IP, and the user level protocol stack and the MAC and PHY layer protocols. In this simulator, the code for the vehicles movement logic is integrated with the network simulation code, which makes it difficult to extend.

Veins (Vehicles in Network Simulation) is another simulator that couples a mobility simulator with a network simulator: SUMO is paired with OMNeT++ by extending SUMO to allow it to communicate with OMNeT++ through a TCP connection. In Veins, there is a manager module that is responsible for synchronizing the two simulators. This simulator has two separate events queues. At regular intervals, the manager module triggers the execution of one time-step of the traffic simulation, receives the resulting mobility trace, and triggers position updates for all modules it had instantiated. In Table 4.1 is shown a comparison between different VANET integrated simulators.

The main properties of VANET simulators are as follows.

- It should be open source, in order to let other users criticize the validity of the model and the implementation. They should offer documentation.
- The code should be clear, in order to let others performing the task in 1.
- The structure should be modular, in order to analyze single pieces of the simulation process.
- Ability of developing complex vehicular mobility models for the simulations. To achieve realistic vehicular traffic simulations the mobility model has to take into

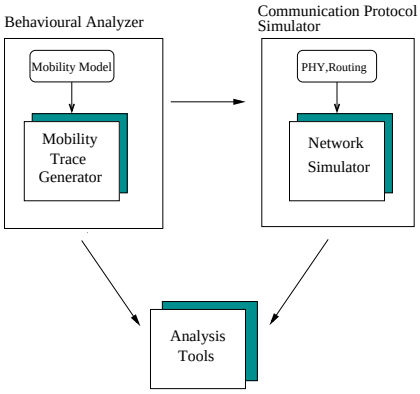


Figure 4.2: Structure of CAVENET.

account both macro-mobility and micro-mobility features of the scenario. The macroscopic description models gross quantities of interest, such as vehicular density or mean velocity, treating vehicular traffic according to fluid dynamics, while the microscopic description considers each vehicle as a distinct entity, modeling its behaviour in a more precise, but computationally more expensive way. Yet, a micro-macro approach may be seen more as a broad classification schema than a formal description of the models' functionalities in each class [31].

- The mobility model may be generated just once and then used for simulating different kind of network configurations.
- NS-2 and NS-3 can be used as network simulator because they have widely accepted network simulation tool, and provides a wide range of protocols. They provides a packet level simulation over a lot of protocols.
- Usage of integrated simulators offers high level of maturity in both areas, traffic simulation and network simulation.

4.2 CAVENET Structure and Description

Our simulator is divided into two blocks, as shown in Fig. 4.2. The first one, which we call Behavioural Analyzer (BA) block, is concerned with the mobility model, and it should take into account the previous parameters in order to produce accurate mobility traces. The second one, which we name Communication Protocol Simulator (CPS), is the protocol simulator, and it is conceived to test the performance of communication protocols given a particular mobility trace. The BA block should be written in a high-level language,

easy to understand and easy to extend. For the particular case of CAVENET, the matrix operations are needed. For this reason, we choose MATLAB. The BA block produces movement patterns which are formatted in a textual format compatible with the CPS's language. Extending the BA block in order to export to other formats is straightforward. The CPS can be one of the many publicly available network simulators, as the well known NS2 or NS3. In principle, the two blocks could also be implemented in two separate machines, in order to speed up the simulation.

4.2.1 Microscopic Model

The core of our simulator is 1-dimensional CA model, which has been first studied by Nagel and Schreckenberg (NaS) [14] in a stochastic settings. The CA is a discrete time model of the vehicle traffic. It is governed by three simple rules. However, as for other CAs, these simple rules can well model and reproduce complex real systems. For this reason, the NaS model has gained a lot of attention during the last ten years.

The time is divided in discrete units Δt, so that $t_n = n\Delta t$. There are N vehicles. A lane k of the road at time $t_n, n \in \mathbb{N}$, is represented by a vector $\mathbf{L}_n^k$ of L sites. The lane is assigned an Nx1 velocity vector $\mathbf{v}_n^k = (v_{i,n}^k)_{i=1}^N$, where $v_{i,n} \in \mathbb{N}_{v_{\max}}$ is the velocity of the vehicle at time t_n and position i. If the ith site is occupied by a car, $L_{i,n} = v_{i,n}$. Otherwise, $L_{i,n} = -1$. We use the lane index only when it is explicitly required. Every cell or site of the lane has a length of s meters. By setting $v_{\max} = 135$km/h and $\Delta t = 1$s, we obtain $s = 7.5$m. At every time step, the velocity v is changed according to the following rules[1].

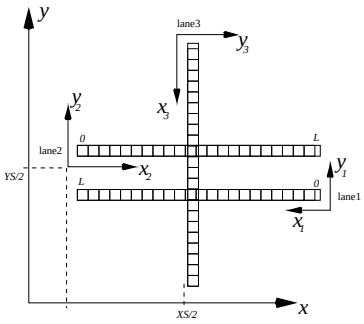
Deterministic, $p = 0$ **or** $p = 1, \forall i$

- 1. $v_{i,n+1} = \min(v_{i,n} + 1, v_{\max})$
- 2. $v_{i,n+1} = \min(v_{i,n}, L_{i+1,n} - L_{i,n} - 1)$
- 3. $\mathbf{L}_{n+1} = \mathbf{L}_n + \mathbf{v}_{n+1}$

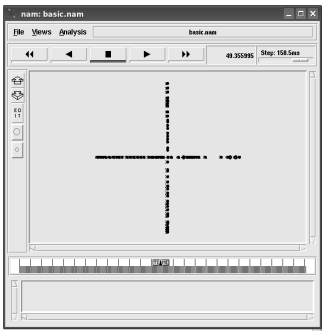
Stochastic

- $2^{'}$. $v_{n+1,i} = \max(0, v_{n,i} - 1)$, with probability p.

[1] We assume parallel update only, i.e. the rules are applied in parallel to every vehicle on the lane.



(a) Line construction.



(b) Excerpt of the generated NS2 trace for 2 lanes network.

Figure 4.3: Lanes construction and NS2 trace.

The vehicle density is $\rho = N/L$. This simple model can recreate the footprints of real traffic scenarios, such as the $1/f$ noise of the average velocity observed in real traffic. The dynamics of the systems are regulated by three important parameters, p, ρ and L. For example, if $p = 0$ the average velocity is SRD, otherwise the system present LRD[2].

4.2.2 Vehicle Model

Every vehicle is a data structure VE_i indexed by its position on the lane. The data structure for the ith vehicle stores: the gap, the velocity, and the current lane position. The relative euclidean position on the lane given by X_i is a unique identifier used for the generation of mobility trace. Moreover, for closed boundaries, i.e. if we suppose circular movement of vehicle on the lane, we check if a shift has taken place. This information will serve to properly generate the trace for NS2 and NS3. It is straightforward to arrange all these information in a vector form, what is the preferred form used in MATLAB.

[2] A stochastic process $\{X_n\}_{n=1}^{n=+\infty}$ is SRD if the autocorrelation is summable:

$$\sum_{k=1}^{+\infty} r(k) < +\infty,$$

where $r(k) = E[(X_n - \overline{X})(X_{n+1} - \overline{X})]/\sigma^2$. Otherwise, if $r(k)$ is not summable, the process is LRD. This means that very distant samples are not statistically independent, contrary to processes without memory, as the Poisson process which is an SRD process.

4.2.3 Lane Construction

Instead of using a particular textual language for describing the position of the lanes in the plane, we use a more general approach. Besides its length, every lane is given a lane transformation, which is used in order to set its real aspect on the plane. This information is used at the mobility trace generation stage. The transformation is a simple affine transformation of the vector $\mathbf{X}_i^k = (X_i, Y_i, 1)$, i.e. the coordinate vector of the *i*th vehicle on the *k*th road with respect to the relative reference system. For example, for the lane $\mathbf{L}^k$, we have the vehicle structure VE_i^k. This structure contains the vector $\mathbf{X}_i^k$. The real position on the plane is computed as $\widetilde{\mathbf{X}}_i^k = \mathbf{A}(k)\mathbf{X}_i^k$ where $\mathbf{A}(k)$ is the lane transformation matrix associated with the *k*-th lane, and $\widetilde{\mathbf{X}}_i^k$ is the vector of coordinates in the absolute reference system (i.e. that used for exporting the NS2 traces). For example, in Fig. 4.3, the third lane has the following absolute coordinates:

$$\widetilde{\mathbf{X}}_i^3 = \begin{pmatrix} 0 & 1 & \frac{\mathrm{XS}}{2} \\ 1 & 0 & \Delta \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_i \\ 0 \\ 1 \end{pmatrix},$$

where XS is the length of the simulation area[3].

4.2.4 NS2

The NS2 is a object oriented simulator, written in C++, with an OTcl interpreter as a front-end. It uses two languages because simulator has two different kind of things it needs to do. On one hand, detailed simulations of protocols require a system programming language, which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. In NS2, the front-end of the program is written in Tool Command Language (TCL). The back-end of NS2 simulator is written in C++ and when the tcl program is compiled, a trace file and nam file are created which define the movement pattern of the nodes and keeps track of the number of packets sent, number of hops between two nodes and connection type at each instance of time.

[3]The parameter Δ is used to avoid an apparent bug in NS2, which fires strange errors when the absolute position is 0.

4.2.5 NS3

The NS3 [37] simulator is developed and distributed completely in the C++ programming language, because it better facilitated the inclusion of C-based implementation code. The NS3 architecture is similar to Linux computers, with internal interface and application interfaces such as network interfaces, device drivers and sockets. The goals of NS3 are set very high: to create a new network simulator aligned with modern research needs and develop it in an open source community. Users of NS3 are free to write their simulation scripts as either C++ main() programs or Python programs. The NS3s low-level API is oriented towards the power-user but more accessible “helper” APIs are overlaid on top of the low-level API.

In order to achieve scalability of a very large number of simulated network elements, the NS3 simulation tools also support distributed simulation. The NS3 support standardized output formats for trace data, such as the pcap format used by network packet analyzing. tools such as tcpdump, and a standardized input format such as importing mobility trace files from NS2 [36].

The NS3 simulator has models for all network elements that comprise a computer network. For example, network devices represent the physical device that connects a node to the communication channel. This might be a simple Ethernet network interface card, or a more complex wireless IEEE 802.11 device.

Chapter 5

JXTA and JXTA-Overlay Platform

5.1 Introduction

JXTA technology is an open and innovative collaboration platform. It consist of six protocols that allow different types of peers to communicate and collaborate among them. The main purpose of JXTA is to built P2P systems that offers the basic functions for P2P communication [40]. JXTA realize this issue working and resolving three main problems of the existing P2P networks:

1. OS independence.
2. Language independence.
3. Providing services and infrastructure for P2P applications.

JXTA-Overlay is a middleware built on top of the JXTA specification, which defines a set of protocols that standardizes how different devices may communicate and collaborate among them. JXTA-Overlay provides a set of basic functionalities, primitives, intended to be as complete as possible to satisfy the needs of most JXTA-based applications. In this chapter, we describe JXTA and JXTA-Overlay P2P distributed platform.

5.2 JXTA Concepts and Components

The main concepts of JXTA networks are as follows.

Peer

A peer is any networked device that implements one or more of the JXTA protocols. Peers can include sensors, phones, and PDAs, PCs, servers, and supercomputers. Each peer operates independently and asynchronously from all other peers, and is uniquely identified by a Peer ID. Peers publish one or more network interfaces for use with the JXTA protocols. Each published interface is advertised as a peer endpoint, which uniquely identifies the network interface. Peer endpoints are used by peers to establish direct point-to-point connections between two peers. Peers are not required to have direct point-to-point network connections between themselves. Intermediary peers may be used to route messages to peers that are separated due to physical network connections or network configuration (e.g., NATS, firewalls, proxies). Peers are typically configured to spontaneously discover each other on the network to form transient or persistent relationships called peer groups.

Peer Groups

A peer group is a collection of peers that have agreed upon a common set of services [41]. Peers self-organize into peer groups, each identified by a unique peer group ID. Each peer group can establish its own membership policy from open (anybody can join) to highly secure and protected (sufficient credentials are required to join). Peers may belong to more than one peer group simultaneously. By default, the first group that is instantiated is the Net Peer Group. All peers belong to the Net Peer Group. The JXTA protocols describe how peers may publish, discover, join, and monitor peer groups. They do not dictate when or why peer groups are created.
There are several motivations for creating peer groups:

- *To create a secure environment.* Groups create a local domain of control in which a specific security policy can be enforced. The security policy may be as simple as a plain text username/password exchange, or as sophisticated as public key cryptography. Peer group boundaries permit member peers to access and publish protected contents.

- *To create a scoping environment.* Groups allow the establishment of a local domain of specialization. For example, peers may group together to implement a document sharing network or a CPU sharing network. Peer groups serve to subdivide the network into abstract regions providing an implicit scoping mechanism. Peer group boundaries define the search scope when searching for a group‘s content.

- *To create a monitoring environment.* Peer groups permit peers to monitor a set of peers for any special purpose.

Groups also form a hierarchical parent-child relationship, in which each group has single parent. Search requests are propagated within the group. The advertisement for the group is published in the parent group in addition to the group itself.
A peer group provides a set of services called peer group services. JXTA defines a core set of peer group services. Additional services can be developed for delivering specific services. In order for two peers to interact via a service, they must both be part of the same peer group.
The core peer group services include the following:

- *Discovery Service:* It is used by peer members to search for peer group resources, such as peers, peer groups, pipes and services.

- *Membership Service:* It is used by current members to reject or accept a new group membership application. Peers wishing to join a peer group must first locate a current member, and then request to join. The application to join is either rejected or accepted by the collective set of current members. The membership service may enforce a vote of peers or elect a designated group representative to accept or reject new membership applications.

- *Access Service:* It is used to validate requests made by one peer to another. The peer receiving the request provides the requesting peers credentials and information about the request being made to determine if the access is permitted.

- *Pipe Service:* It is used to create and manage pipe connections between the peer group members.

- *Resolver Service:* It is used to send generic query requests to other peers. Peers can define and exchange queries to find any information that may be needed (e.g., the status of a service or the state of a pipe endpoint).

- *Monitoring Service:* It is used to allow one peer to monitor other members of the same peer group.

Network Services

Peers cooperate and communicate to publish, discover, and invoke network services. Peers can publish multiple services. Peers discover network services via the Peer Discovery Protocol. The JXTA protocols recognize two levels of network services:

- *Peer Services:* A peer service is accessible only on the peer that is publishing that service. If that peer should fail, the service also fails. Multiple instances of the service can be run on different peers, but each instance publishes its own advertisement.

- *Peer Group Services:* A peer group service is composed of a collection of instances (potentially cooperating with each other) of the service running on multiple members of the peer group. If any one peer fails, the collective peer group service is not affected (assuming the service is still available from another peer member). Peer group services are published as part of the peer group advertisement.

Modules

Modules provides a generic abstraction to allow a peer to instantiate a new behavior. As peers browse or join a new peer group, they may find new behaviors that they may want to instantiate. For example, when joining a peer group, a peer may have to learn a new search service that is only used in this peer group. In order to join this group, the peer must instantiate this new search service. The module enables the representation and advertisement of platform-independent behaviors, and allows peers to describe and instantiate any type of implementation of a behavior. For example, a peer has the ability to instantiate either a Java or a C implementation of the behavior. The ability to describe and publish platform-independent behavior is essential to support peer groups composed of heterogeneous peers. The module advertisements enable JXTA peers to describe a behavior in a platform-independent manner. The JXTA platform uses module advertisements to self-describe itself.
The module abstraction includes a module class, module specification, and module implementation:

- *Module Class.* The module class is primarily used to advertise the existence of a behavior. The class definition represents an expected behavior and an expected

binding to support the module. Each module class is identified by a unique ID, the ModuleClassID.

- *Module Specification.* The module specification is primarily used to access a module. It contains all the information necessary to access or invoke the module. A module specification is one approach to provide the functionality that a module class implies. There can be multiple module specifications for a given module class. Each module specification is identified by a unique ID, the ModuleSpecID. The ModuleSpecID contains the ModuleClass ID, indicating the associated module class. A module specification implies network compatibility. All implementations of a given module specification must use the same protocols and are compatible, although they may be written in a different language.

- *Module Implementation.* The module implementation is the implementation of a given module specification. There may be multiple module implementations for a given module specification. Each module implementation contains the ModuleSpecID of the associated specification it implements.

Pipes

Pipes are an asynchronous and unidirectional message transfer mechanism used for service communication. Pipes are virtual communication channels and may connect peers that do not have a direct physical link. Pipes support the transfer of any object, including binary code, data strings, and Java technology-based objects. The pipe endpoints are referred to as the input pipe (the receiving end) and the output pipe (the sending end). Pipe endpoints are dynamically bound to peer endpoints at runtime. Peer endpoints correspond to available peer network interfaces (e.g., a TCP port and associated IP address) that can be used to send and receive message [42]. JXTA pipes can have endpoints that are connected to different peers at different times, or may not be connected at all.

- *Point-to-point Pipes.* A point-to-point pipe connects exactly two pipe endpoints together. An input pipe on one peer receives messages sent from the output pipe of another peer.

- *Propagate Pipes.* A propagate pipe connects one output pipe to multiple input pipes. Messages flow from the output pipe (the propagation source) into the input

pipes. All propagation is done within the scope of a peer group. That is, the output pipe and all input pipes must belong to the same peer group.

- *Secure Unicast Pipes.* A secure unicast pipe is a type of point-to-point pipe that provides a secure communication channel.

Messages

A message is an object that is sent between JXTA peers. It is the basic unit of data exchange between peers. Messages are sent and received by the Pipe Service and by the Endpoint Service. Typically, applications use the Pipe Service to create, send, and receive messages. A message is an ordered sequence of named and typed contents called message elements. Thus a message is essentially a set of name/value pairs. The content can be an arbitrary type. The JXTA protocols are specified as a set of messages exchanged between peers. Each software platform binding describes how a message is converted to and from a native data structure such as a Java technology object or a C structure. There are two representations for messages: XML and binary. Binary data may be encoded using a Base64 encoding scheme in the body of an XML message. The use of XML messages to define protocols allows many different kinds of peers to participate in a protocol. Because the data is tagged, each peer is free to implement the protocol in a manner best-suited to its abilities and role. If a peer only needs some subset of the message, the XML data tags enable that peer to identify the parts of the message that are of interest.

Advertisements

All JXTA network resources, such as peers, peer groups, pipes, and services, are represented by an advertisement. Advertisements are language-neutral metadata structures represented as XML documents. The JXTA protocols use advertisements to describe and publish the existence of a peer resources. Peers discover resources by searching for their corresponding advertisements, and may cache any discovered advertisements locally. Each advertisement is published with a lifetime that specifies the availability of its associated resource. Lifetimes enable the deletion of obsolete resources without requiring any centralized control. An advertisement can be republished (before the original advertisement expires) to extend the lifetime of a resource.

Security

Dynamic P2P networks such as the JXTA network need to support different levels of resource access. JXTA peers operate in a role-based trust model, in which an individual peer acts under the authority granted to it by another trusted peer to perform a particular task. Five basic security requirements must be provided:

- *Confidentiality*: It guarantees that the contents of a message are not disclosed to unauthorized individuals.
- *Authentication*: It guarantees that the sender is who he or she claims to be.
- *Authorization*: It guarantees that the sender is authorized to send a message.
- *Data integrity*: It guarantees that the message was not modified accidentally or deliberately in transit.
- *Refutability*: It guarantees that the message was transmitted by a properly identified sender and is not a replay of a previously transmitted message.

XML messages provide the ability to add metadata such as credentials, certificates, digests, and public keys to JXTA messages, enabling these basic security requirements to be met. Message digests guarantee the data integrity of messages. Messages may also be encrypted (using public keys) and signed (using certificates) for confidentiality and refutability.
Credentials can be used to provide message authentication and authorization. A credential is a token that is used to identify a sender, and can be used to verify the sender's right to send a message to a specified endpoint. The credential is an opaque token that must be presented each time a message is sent. The sending address placed in a JXTA message envelope is cross-checked with the senders identity in the credential. Each credential's implementation is specified as a plug-in configuration, which allows multiple authentication configurations to co-exists on the same network. It is the intent of the JXTA protocols to be compatible with widely accepted transport-layer security mechanisms for message-based architectures, such as Secure Sockets Layer (SSL) and Internet Protocol Security (IPSec). However, secure transport protocols such as SSL and IPSec only provide the integrity and confidentiality of message transfer between two communicating peers. In order to provide secure transfer in a multi-hop network like JXTA, a trust association must

be established among all intermediary peers. Security is compromised if any one of the communication links is not secured.

IDs

Peers, peer groups, pipes and other JXTA resources need to be uniquely identifiable. A JXTA ID uniquely identifies an entity and serves as a canonical way of referring to that entity. Currently, there are six types of JXTA entities which have JXTA ID types defined: peers, peer group, pipes, contents, module classes, and module specifications.

5.3 JXTA Architecture

JXTA has a structure with three layers, as shown in Fig. 5.1. The system is designed in a modular way to let the developers to choose the set of services that satisfy their needs.

1. **The Core Layer.** This layer implements the essential set of primitives that are common to P2P networking. These primitives includes creation of peers, discovery, transport, peer groups and communication even behind firewalls or NATs. The JXTA core include also basic security services.

2. **The Services Layer.** This layer implements some services that are integrated to JXTA. These services include searching and indexing, file sharing, protocol translation, authentication and Public Key Infrastructure (PKI) services, as well resource search.

3. **Applications Layer.** This layer implements applications that are integrated to JXTA, such as P2P instant messaging, file sharing and content management.

The distinction between services and final applications may not always be clear, since what a client may consider an application may be considered a service by another peer. For that reason, the system is designed in a modular way, letting developers choose the set of services and applications which most satisfy their needs. All JXTA components are within these three layers.

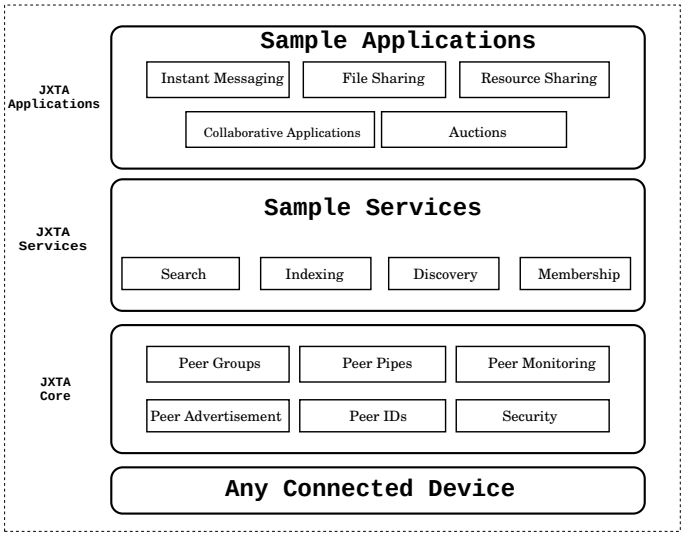


Figure 5.1: JXTA architecture.

5.4 JXTA Protocols

JXTA comprises a set of six open protocols that allow devices with different software to collaborate. A peer can implement one of these protocols and they can ask the other peers for supplement functionalities. These protocols are briefly described below.

Peer Resolver Protocol (PRP) - It allows a peer to send a search query to another peer. This protocol is a basic communications protocol that follows a request/response format. The resolver is used to support communications in the JXTA protocols like the router and the discovery protocols. The resolver also allows for the propagation of queries. For example, if a peer receives a query and does not know the answer, the resolver sends the query to other peers. This is an interesting feature, especially because the originating peer does not need to have any knowledge of a peer that may actually have the result to the query.

Peer Information Protocol (PIP) - It allows a peer to learn about the status of another peer. The information protocol is used partially like ping and partially to obtain basic information about a peer's status.

Pipe Binding Protocol (PBP) - It is used to create a communications path between one or more peers. A pipe is a virtual channel between two peers. The protocol is primarily concerned with connecting peers via the route(s) supplied by the peer endpoint protocol.

Rendezvous Protocol (RVP) - The Rendezvous Protocol is responsible for the propaga-

tion of the messages in JXTA groups. The Rendezvous Protocol defines a base protocol for peers to send and receive messages inside the group of peers. This protocol controls how the messages are propagated.

Endpoint Routing Protocol (ERP) - Is used to find available routes to route messages to destination peers. The protocol uses gateways between peers to create a path that consists of one or more of the pipe protocols suitable for creating a pipe. The pipe binding protocol uses the list of peers to create routes between peers.

Peer Discovery Protocol (PDP) - It allows a peer to discover other peer advertisements (peer, group, service, or pipe). This protocol uses a searching mechanism to locate information. The protocol can find peers, peer groups, and all other published advertisements. The advertisements are mapped to peers, groups, pipes or other resources.

5.5 JXTA-Overlay Platform

JXTA-Overlay project is an effort to use JXTA technology for building an overlay on top of JXTA offering a set of basic primitives (functionalities) that are most commonly needed in JXTA-based applications. The proposed overlay comprises the following primitives: peer discovery, peer's resources discovery, resource allocation, task submission and execution, file/data sharing, discovery and transmission, instant communication, peer group functionalities (groups, rooms etc.), monitoring of peers, groups and tasks.

The overlay is built on top of JXTA layer and provides a set of primitives that can be used by other applications, which on their hand, will be built on top of the overlay, with complete independence. The JXTA-Overlay project has been developed using the ver-2.3 JXTA libraries. In fact, the project offers several improvements of the original JXTA protocols/services in order to increase the reliability of JXTA-based distributed applications [43] and to support group management and file sharing.

The architecture of P2P distributed platform which we have developed using JXTA technology has two main peers: Broker and Client. Altogether these two peers form a new overlay on top of JXTA. The structure of JXTA-Overlay system is shown in Fig. 5.2.

5.5.1 Internal Architecture of JXTA-Overlay

The internal architecture of JXTA-Overlay is shown in Fig. 5.3. Except Broker and Client peers, the JXTA-Overlay has also SimpleClient peers. The control layer interacts with the

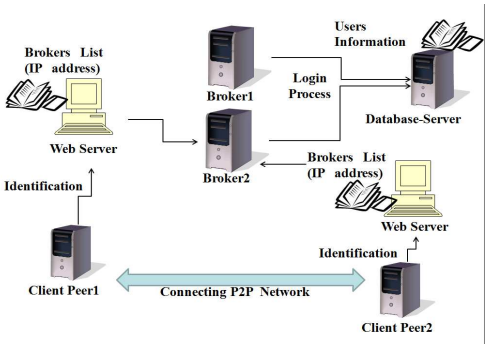


Figure 5.2: Structure of JXTA-Overlay system.

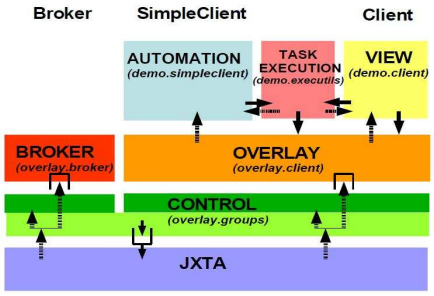


Figure 5.3: Internal architecture of JXTA-Overlay.

JXTA layer and is divided into two parts: a lower part with functionality common to any kind of peer, and a higher part with functionality specific to Brokers and Clients.

- The common part provides functionality for doing JXTA messaging, discovery and advertisement.
- The Broker specific part provides functionality for managing groups of Brokers and keeping broker statistics.
- The Client specific part provides functionality for managing groups of Clients, keeping client statistics, managing its shareable files, managing the user configuration and creating the connection with a Broker.

The lower part enqueues the JXTA messages to be sent. Whenever a message arrives, the JXTA layer fires an event to the lower layer, which in turn fires a notifications to the upper layers.

Chapter 6

JXTA-Overlay P2P Applications

6.1 Application of JXTA-Overlay for Secure Robot Control

Robotics is rapidly expanding into human environments and vigorously engaged in its new emerging challenges. Interacting, exploring, and working with humans, the new generation of robots will increasingly touch people and their lives [44]. The successful introduction of robots in human environments will rely on the development of competent and practical systems that are dependable, safe, and easy to use [45].

Physical interactivity is a key characteristic for providing these robots with the ability to perform and interact with the surrounding world. The ability to interact with people in the human environment has been a recent motivator of the humanoid robotics community and the service robotics community.

In this section we describe the design and implementation of our proposed applications for robot control based on JXTA-Overlay.

6.1.1 Robot Control Using the Original Primitives of JXTA-Overlay

In this section are described the steps followed for the robot control, using unsecure primitives. The first step before the client peer may try to join JXTA-Overlay network is Broker connection. This connection is realized via the connect primitive. This is non-secure method and just locates the broker and wait until the connection become available. No message exchange happens between the broker and the client peer.
After the establishment of the connection with the broker, the client peer tries to authenti-

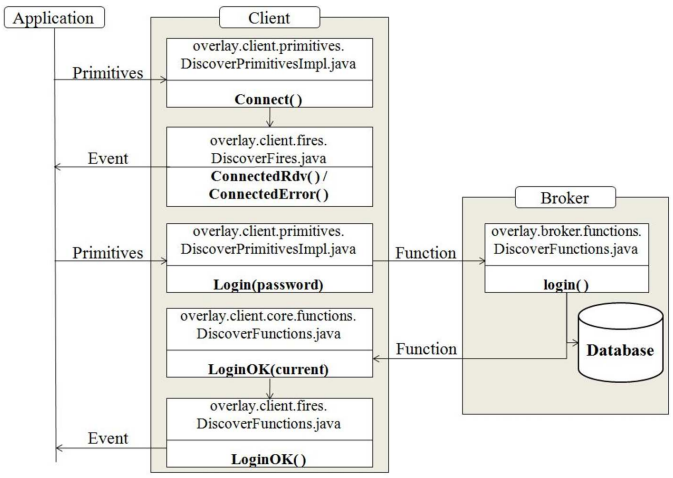


Figure 6.1: User authentication system (no security).

cate to the broker via sending its username and password. This message sent to the broker executes the login function to the broker side. When the username or password results in error, a response loginError function is sent back to the client. When the username and password are ok, a response loginOk is sent back to the client.

Once the client peers are connected to JXTA-Overlay network, they are able to exchange messages. The Robot control command is sent like a simple text message using sendMessage primitive. In Fig. 6.1 is shown the user authentication system when no security exist on JXTA-Overlay.

6.1.2 Robot Control Using Secure Discovery Primitives

The secure version of the *connection* primitive, is the primitive *authBroker*. A two-way message is produced for the authentication of the Broker and setup some initial cryptographic data exchange. The resulting message executes the *authRequest* function at the broker side and final broker response executes the *authResponse* function at the original client. At these stage the client is ready to try to join the network. It locates the broker and waits for a connection to open. After that, it authenticates the broker by initiating a challenge-response protocol. The Challenge-Handshake Authentication Protocol (CHAP) is used to periodically verify the identity of the peer using a 3-way handshake. This is done upon initial link establishment, and may be repeated anytime after the link has been established.

1. After the Link Establishment phase is complete, the authenticator sends a “challenge” message to the peer.

2. The peer responds with a value calculated using a “one-way hash” function.

3. The authenticator checks the response against its own calculation of the expected hash value. If the values match, the authentication is acknowledged, otherwise the connection should be terminated.

4. At random intervals, the authenticator sends a new challenge to the peer, and repeats steps 1 to 3.

Then, the broker responds to the challenge by signing the data and sending its own credential. An SID (Session Identifier)(128-160 bit) is generated and sent to the client peer. The client peer verifies the challenge signature and the credential. If the signature is correct and the credential is valid, then is achieved a legitimate connection with a broker. Both the SID and the broker’s credential are locally stored in the client peer. The primitive secureLogin maintains the message exchange, and signs and encrypts the username and password sent to the broker, together with the SID, obtained from the authBroker primitive call. The message is encrypted using the broker‘s public key, retrieved during broker authentication. The session identifier and the signature are used to avoid reply attacks, because just encryption does not protect against them and an attacker can reuse them to impersonate client peers.

During the login process, a credential may be used to authenticate other client peers and as a means to transport and publish key information. All the information is processed at the broker-side function *secureLogin*. The class Credential-Request ID is used to encapsulate all data related to the credential request, as well as the username and password. This class format is serialized and is transparent to both the control layer and JXTA messaging. The serialized form is then encrypted and Base64 encoded, thus becoming an ordinary String.

Once the client peers are connected to the JXTA-Overlay network in a secure manner, possessing as a credential and the corresponding cryptographic keys. They are able to exchange secure messages.

The primitive *secureSendMessage* is used to send a secure message (robot control command) to the broker. This secure version encrypts data using the destination’s public key via a wrapped key approach. Once the data is encrypted, it is encoded back to a String

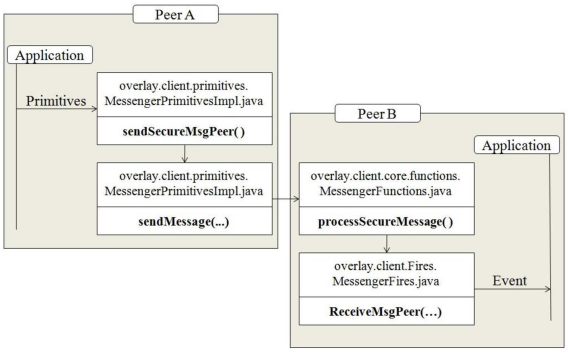


Figure 6.2: Peer encryption communication system.

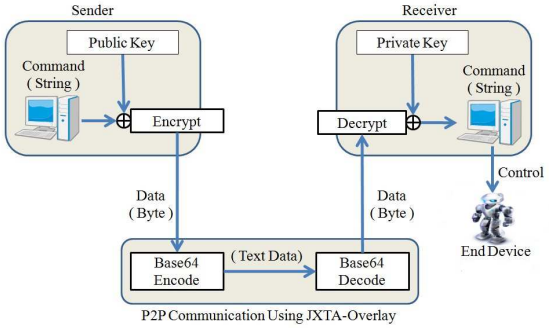


Figure 6.3: The secure transmission of control commands in JXTA-Overlay.

using the Base64 algorithm and then uses the original non-secure version to transmit the data. Fig. 6.2 shows the peer encryption communication system. This process is shown in Fig. 6.3. In Fig. 6.4 is shown the user authentication system using encryption technology.

6.2 Data Replication in JXTA-Overlay P2P System: A Fuzzy-based Approach

6.2.1 Fuzzy Logic

Application of Fuzzy Logic for Control

The ability of fuzzy sets and possibility theory to model gradual properties or soft constraints whose satisfaction is matter of degree, as well as information pervaded with

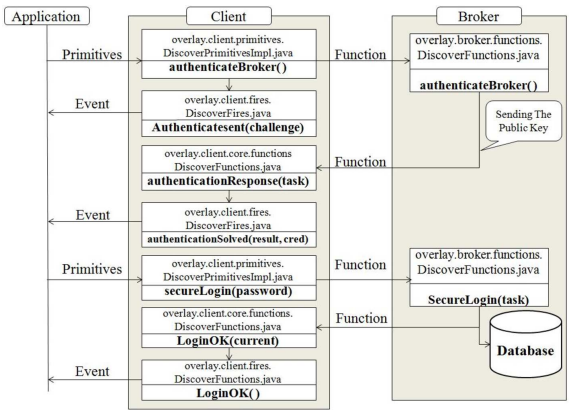


Figure 6.4: User authentication system using encryption technology.

imprecision and uncertainty, makes them useful in a great variety of applications. The most popular area of application is Fuzzy Control (FC), since the appearance, especially in Japan, of industrial applications in domestic appliances, process control, and automotive systems, among many other fields.

FC

In the FC systems, expert knowledge is encoded in the form of fuzzy rules, which describe recommended actions for different classes of situations represented by fuzzy sets.

In fact, any kind of control law can be modeled by the FC methodology, provided that this law is expressible in terms of ”if ... then ...” rules, just like in the case of expert systems. However, FL diverges from the standard expert system approach by providing an interpolation mechanism from several rules. In the contents of complex processes, it may turn out to be more practical to get knowledge from an expert operator than to calculate an optimal control, due to modeling costs or because a model is out of reach.

Linguistic Variables

A concept that plays a central role in the application of FL is that of a linguistic variable. The linguistic variables may be viewed as a form of data compression. One linguistic variable may represent many numerical variables. It is suggestive to refer to this form of data compression as granulation [46].

The same effect can be achieved by conventional quantization, but in the case of quantization, the values are intervals, whereas in the case of granulation the values are overlapping fuzzy sets. The advantages of granulation over quantization are as follows:

- it is more general;
- it mimics the way in which humans interpret linguistic values;
- the transition from one linguistic value to a contiguous linguistic value is gradual rather than abrupt, resulting in continuity and robustness.

FC Rules

FC describes the algorithm for process control as a fuzzy relation between information about the conditions of the process to be controlled, x and y, and the output for the process z. The control algorithm is given in "if-then" expression, such as:

If x is small and y is big, then z is medium;
If x is big and y is medium, then z is big.

These rules are called *FC rules*. The "if" clause of the rules is called the antecedent and the "then" clause is called consequent. In general, variables x and y are called the input and z the output. The "small" and "big" are fuzzy values for x and y, and they are expressed by fuzzy sets.

Fuzzy controllers are constructed of groups of these FC rules, and when an actual input is given, the output is calculated by means of fuzzy inference.

Control Knowledge Base

There are two main tasks in designing the control knowledge base. First, a set of linguistic variables must be selected which describe the values of the main control parameters of the process. Both the input and output parameters must be linguistically defined in this stage using proper term sets. The selection of the level of granularity of a term set for an input variable or an output variable plays an important role in the smoothness of control. Second, a control knowledge base must be developed which uses the above linguistic description of the input and output parameters. Four methods [47, 48, 49, 50] have been suggested for doing this:

- expert's experience and knowledge;
- modelling the operator's control action;
- modelling a process;
- self organization.

Among the above methods, the first one is the most widely used. In the modeling of the human expert operator's knowledge, fuzzy rules of the form "If Error is small and Change-in-error is small then the Force is small" have been used in several studies [51, 52]. This method is effective when expert human operators can express the heuristics or the knowledge that they use in controlling a process in terms of rules of the above form.

Defuzzification Methods

The defuzzification operation produces a non-FC action that best represent the membership function of an inferred FC action. Several defuzzification methods have been suggested in literature. Among them, four methods which have been applied most often are:

- Tsukamoto's Defuzzification Method;
- The Center of Area (COA) Method;
- The Mean of Maximum (MOM) Method;
- Defuzzification when Output of Rules are Function of Their Inputs.

6.2.2 Proposed Fuzzy-based System for Data Replication in P2P Networks

This section presents a fuzzy-based replication system for P2P systems. If a part or the document in a peer changes, other peers that have the replica of this document make the changes. In data replication systems it is important to find a right replication factor. The replication factor is considered the total number of replicated documents over the total number of documents which means the sum of the replicas and original documents in all peers.

For the Replication Factor (*RF*), we take in consideration three parameters: Number of Documents per Peer (*NDP*), Replication Percentage (*RP*), and Scale of Replication per

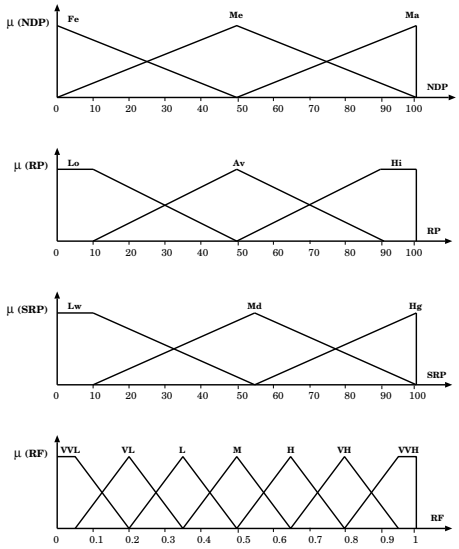


Figure 6.5: Membership functions.

Peer (*SRP*). The membership functions for this system are shown in Fig. 6.5. In Table 6.1, we show the Fuzzy Rule Base (FRB) of this system, which consists of 27 rules.

The input parameters for peer-reliability assessment are: NDP, RP, NPG, while the output linguistic parameter is *RF*. The term sets of *NDP*, *RP*, and *SRP* are defined respectively as:

$$\begin{aligned}\mu(NDP) &= \{Few,\ Medium,\ Many\}\\ &= \{Fe,\ Me,\ Ma\};\\ \mu(RP) &= \{Low,\ Average,\ High\}\\ &= \{Lo,\ Av.\ Hi\};\\ \mu(SRP) &= \{Low,\ Medium,\ High\}\\ &= \{Lw,\ Md,\ Hg\}.\end{aligned}$$

The term set for the output (*RF*) is defined as:

$$\begin{aligned}\mu(RF) = \{&Very\ Very\ Low,\ Very\ Low,\ Low,\ Middle,\ High,\\ &Very\ High,\ Very\ Very\ High\}\\ = \{&VVL,\ VL,\ L,\ M,\ H,\ VH,\ VVH\}.\end{aligned}$$

Table 6.1: FRB.

Rules	NDP	RP	SRP	RF
0	Fe	Lo	Lw	VVL
1	Fe	Lo	Md	VL
2	Fe	Lo	Hg	L
3	Fe	Av	Lw	VL
4	Fe	Av	Md	L
5	Fe	Av	Hg	M
6	Fe	Hi	Lw	L
7	Fe	Hi	Md	M
8	Fe	Hi	Hg	H
9	Me	Lo	Lw	VL
10	Me	Lo	Md	L
11	Me	Lo	Hg	M
12	Me	Av	Lw	L
13	Me	Av	Md	M
14	Me	Av	Hg	H
15	Me	Hi	Lw	M
16	Me	Hi	Md	H
17	Me	Hi	Hg	VH
18	Ma	Lo	Lw	L
19	Ma	Lo	Md	M
20	Ma	Lo	Hg	H
21	Ma	Av	Lw	M
22	Ma	Av	Md	H
23	Ma	Av	Hg	VH
24	Ma	Hi	Lw	H
25	Ma	Hi	Md	VH
26	Ma	Hi	Hg	VVH

Chapter 7

Evaluation Results for V2V Communication

In this chapter, we present evaluation results for V2V communication for different scenarios using CAVENET, NS2 and NS3. We evaluate and compare the performance of different routing protocols considering UDP and TCP traffic and using different metrics.

7.1 Mobility Model Validation

We present here some simulations for the NaS model by means of CAVENET. We use as simulation variable the average velocity $\overline{v}(t) = N^{-1}\sum_{i=1}^{N} v_i(t) = N^{-1}\| \mathbf{v}(\mathbf{t}) \|_1$ of all cars. CAVENET can analyze and design single and multiple lanes traces and also can run Monte Carlo simulations. In Fig. 7.1, we present the results for the so called fundamental diagram (the flow vs. density diagram). The flow at a particular lane section is defined as $J = \rho\overline{v}$. Each point in the figure is the ensemble average over 20 trials of a simulation trace lasting 500 iterations. Moreover, we can also visualize the space-time plot of the traffic, i.e. the evolution of the velocity for every vehicle along the road. We obtain the two traffic regimes, namely the laminar regime and the jammed or congested regime, as shown in Fig. 7.2-a and Fig. 7.2-b, respectively. We are interested in the stationary distribution and transient time, which are very important to assess the next stage simulations, i.e. those related to the communication protocol analysis.

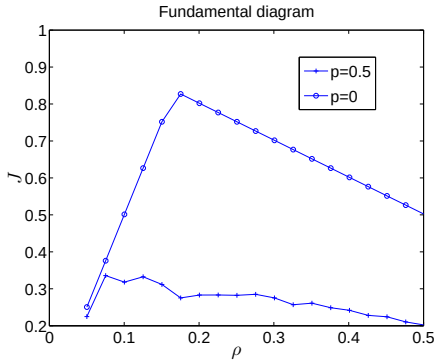
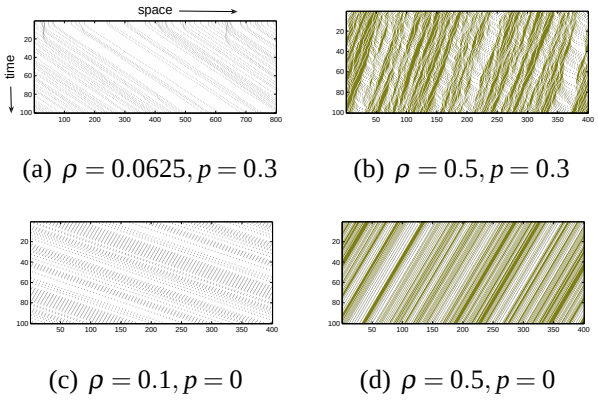


Figure 7.1: Traffic flow as a function of ρ and p for $L = 400$.



(a) $\rho = 0.0625, p = 0.3$ (b) $\rho = 0.5, p = 0.3$

(c) $\rho = 0.1, p = 0$ (d) $\rho = 0.5, p = 0$

Figure 7.2: Space-time plots showing the jam wave in different settings.

7.2 Stationary Distribution

Usually, RW-like mobility models used in simulation exhibit the velocity decay problem. That means that the simulation variable slowly decays towards a steady state value as the simulation time proceeds. This is problematic, because we do not know when this transient ends. Consequently, we do not know precisely how to remove the transient values. The root of this phenomenon has to be attributed to the underlying mobility model, which has been assumed random. Every node randomly picks a velocity from a continuous uniformly distributed random variable between $[v_{\min}, v_{\max}]$. The velocity is changed at particular points called waypoints. In this way, the system has an infinite (but countable) number of states. The general solution to this problem consists in finding the steady state distribution of the simulation variable and let the system starts with that distribution. This reasoning is also equivalent to consider Palm probability distributions instead of the usual ones [11].

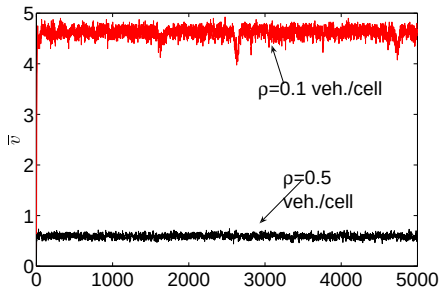
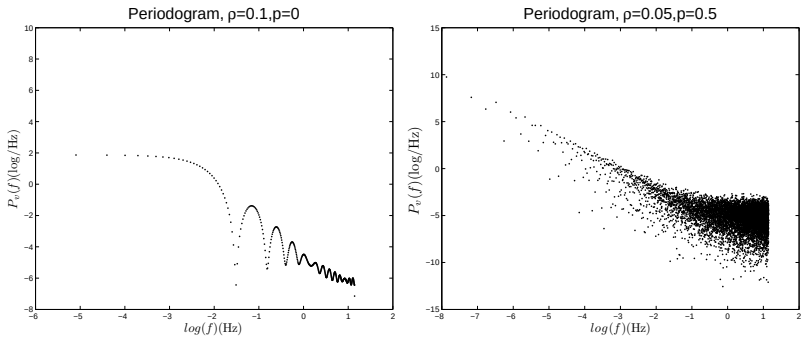


Figure 7.3: Sample realizations of $\overline{v}(t)$.

In our case, the system has inherently a finite state space. The automaton could be represented by a discrete-time finite-state Markov chain. We know that a Markov chain with a single class of recurrent states has always a steady state distribution. Moreover, since a Markov chain with finite state space has always at least one recurrent state, we conclude that the steady state distribution exists and is unique. The convergence rate toward this steady-state distribution depends on the eigenstructure of the transition probability matrix of the Markov chain. The problem here is that a Markov chain model is not suitable, because the process can be, in general, LRD, for $0 < p < 1$. Moreover, even in the SRD case, finding the transition probabilities is not easy.

In general, mobility models for vehicular traffic exhibits a phase transition around a particular value of ρ. As we can see in Fig. 7.2, for $p > 0$, the traffic is composed of jammed regions which travel on the opposite direction of movement. For low densities, these waves die out very quickly, as shown also in Fig. 7.3. But for higher densities there are many interconnected clusters of jammed vehicles. In this case, the steady state is reached very slowly. Therefore, it is important to investigate how many samples should be removed from the staring point in order to sample a process in its stationary regime.

In order to clarify this phenomenon, we measured the transient time τ for $p = 0$, i.e. the deterministic case. In this case, $\overline{v}(t)$ is not LRD. We can show this fact also by plotting the periodogram of $\overline{v}(t)$. In Fig. 7.4-a, we see that for $f \to 0$, the periodogram does not diverge. On the other hand, for $p > 0$, in Fig. 7.4-b, the estimated spectrum diverges at the origin, i.e. the underlying process has the LRD property.



(a) Deterministic model.

(b) 1/f noise like spectrum form the the stochastic version of the NaS model.

Figure 7.4: Deterministic model and stochastic version.

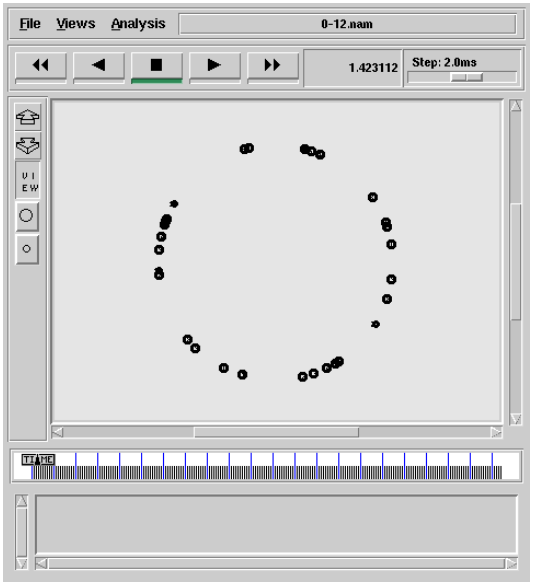


Figure 7.5: Excerpt of the generated NS-2 trace for one lane network.

7.3 Performance Evaluation of Different Routing Protocols Using NS2

7.3.1 Scenario 1: Simulation Results for UDP traffic

We used one circular line and 30 nodes for simulations. The excerpt of the generated NS-2 trace for one lane network is shown in Fig. 7.5. The simulation time is 100 seconds. The receiving node is node 0 and the sending nodes are from node 1 to node 29. We prepared each scenario based on nodes ID. The mobility pattern for all scenarios is the

Table 7.1: Simulation parameters for scenario 1.

Parameter	Value
Network Simulator	NS2
Routing Protocol	AODV, OLSR, DYMO, DSDV, DSR
Simulation Time	100 s
Simulation Area	3000 m Circuit (400 cells)
Number of Nodes	30
Traffic Source/Destination	Deterministic
DATA TYPE	CBR
Packets Generation Rate	5 packets/s
Packet Size	512 bytes
MAC Protocol	IEEE802.11 DCF
MAC Rate	2 Mbps
RTS/CTS	None
Transmission Range	250 m
Radio Propagation Models	Two-ray Ground
Hello_{AODV} Interval	1 s
Hello_{OLSR} Interval	1 s
TC_{OLSR} Interval	2 s
Hello_{DYMO} Interval	1 s
Maximum	135 km/h
Network Simulation Time	10 s - 90s

same. In order to evaluate the performance of each protocol, 5 packets per second as a Constant Bit Rate (CBR) traffic were transmitted between 10 seconds and 90 seconds. As transport protocol we used UDP. When the packets are not received, they are calculated as loss packets and reflected in the PDR value. The simulation parameters are shown in Table 7.1.

As evaluation metrics, we use two parameters: Packet Delivery Ratio (PDR) and goodput.

- *Packet Delivery Ratio (PDR)* - It is the ratio between the number of packets delivered to the receiver and the number of packets sent by the source.

- *Goodput* - It is the number of correct packets received in the destination node. The goodput does not consider the header data and re-sent packets.

The simulations results of goodput are shown in Fig. 7.6, Fig. 7.7, Fig. 7.8, Fig. 7.9 and Fig. 7.10. In Fig. 7.6 is shown the goodput of AODV protocol. The packet bit rate is 20 kbps. The goodput of AODV is about ten times of CBR packet size. This is because after a back-off time all the accumulated data packets are transmitted in the discovered route. If we increase the background traffic, the number of transmitted packets will again

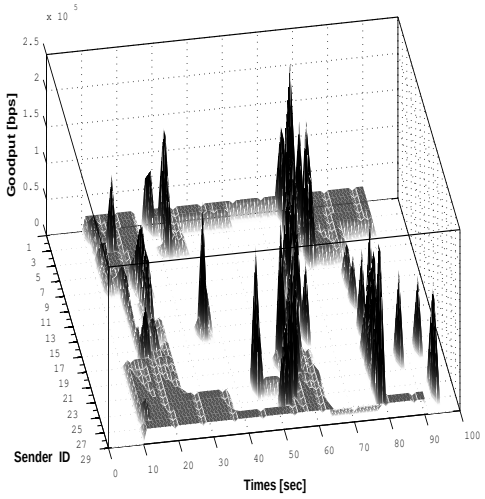


Figure 7.6: Goodput of AODV.

increases and the network may be congested. Also, after 60 seconds, in AODV protocol, there is a delay caused by route finding mechanism.

For OLSR protocol the communication between node 15 and receiver can not established, so the goodput is zero. This is because the MPR nodes are not equipped with buffer. In the case of DYMO, the goodput is stable.

Comparing Fig. 7.6, Fig. 7.7 and Fig. 7.8, we can see that reactive protocols (AODV and DYMO) have better goodput than OLSR. For AODV and DYMO, even the nodes are far from each other they communicate with each other.

Comparing Fig. 7.9 and Fig. 7.8, we can see that DSR have better goodput than DYMO protocol. However, DSR protocol needs more time to search a new route. In the case of DYMO, the goodput is stable. DYMO combines the advantages of reactive protocols AODV and DSR with some link state features of OLSR. These characteristics make it more adaptive and obtain stable goodput.

In Fig. 7.10 is shown the goodput of DSDV protocol. As we can see, nodes from 9 to 23 can not communicate with the destination node. This happens because the DSDV protocol maintains only one route per destination and packets that cannot be delivered by the MAC layer are dropped due to the lack of alternate routes.

Comparing Fig. 7.10 and Fig. 7.8, we can see that DYMO have better goodput than DSDV protocol. The different basic working mechanism of these protocols leads to the differences in the performance. For DYMO, even the nodes are far from each other they communicate together.

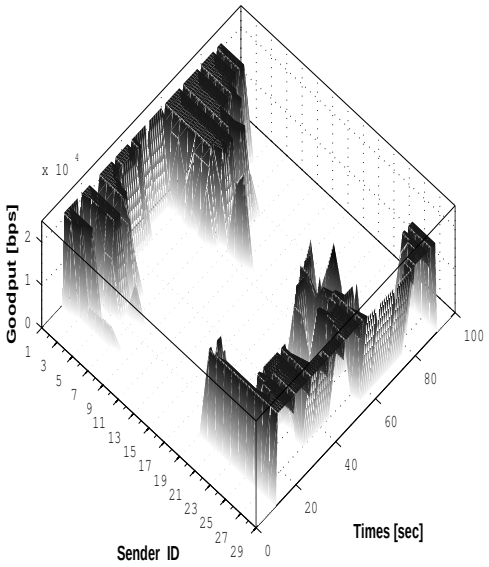


Figure 7.7: Goodput of OLSR.

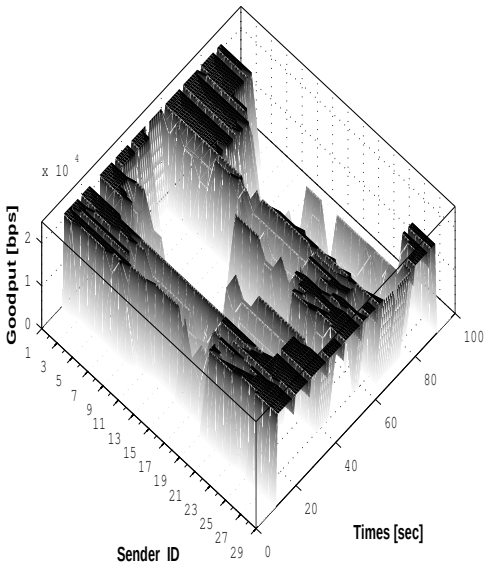


Figure 7.8: Goodput of DYMO.

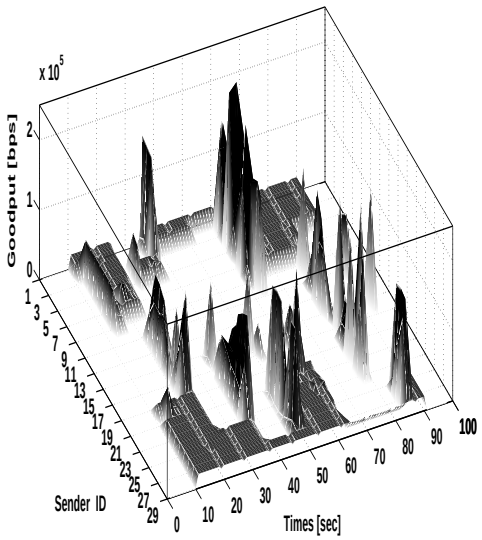


Figure 7.9: Goodput of DSR.

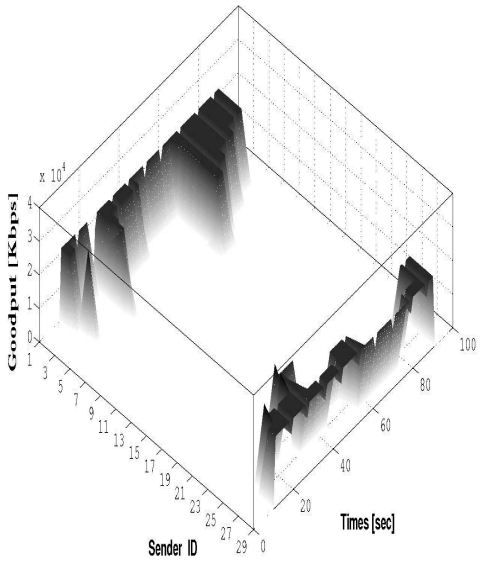


Figure 7.10: Goodput of DSDV.

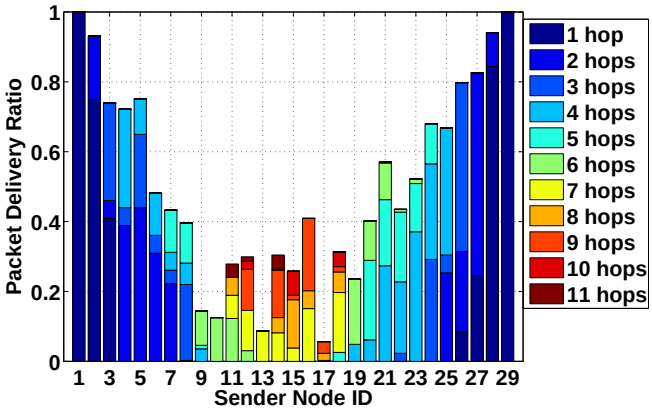


Figure 7.11: PDR for AODV.

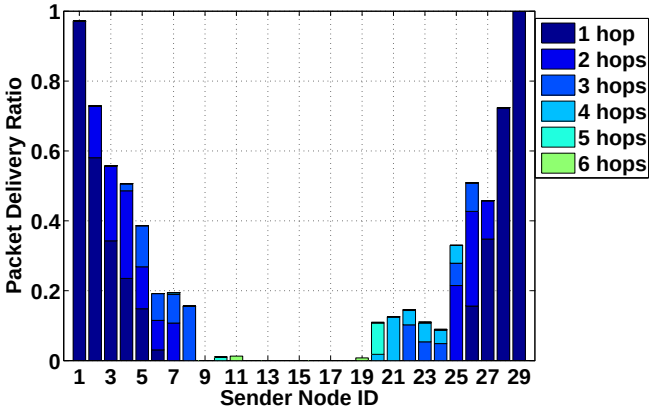


Figure 7.12: PDR for OLSR.

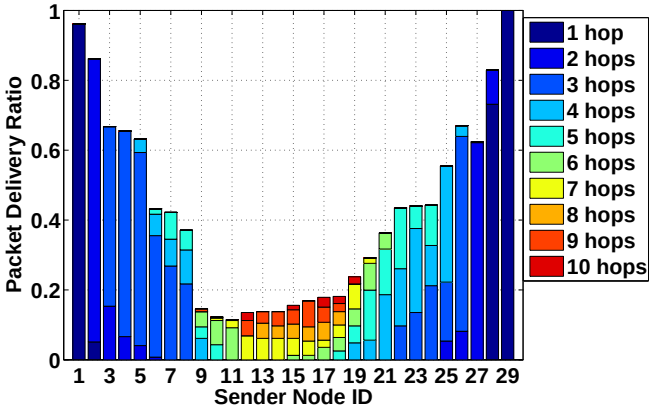


Figure 7.13: PDR for DYMO.

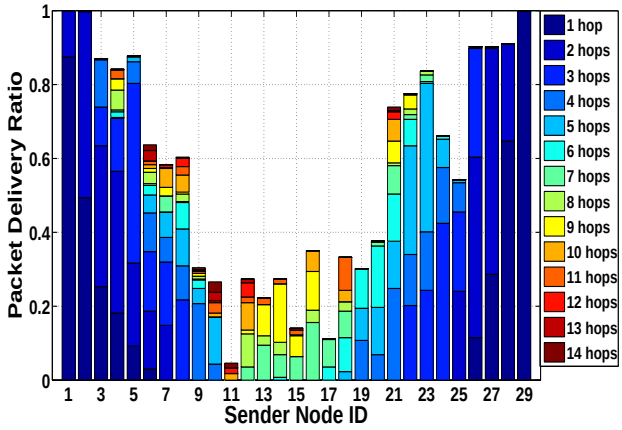


Figure 7.14: PDR of DSR.

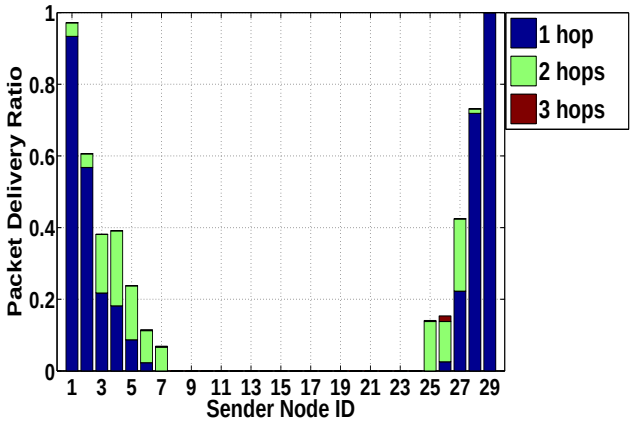


Figure 7.15: PDR of DSDV.

From Fig. 7.11 to Fig. 7.13, we show the PDR for AODV, OLSR and DYMO routing protocols. We can see that among three protocols AODV has a better goodput. However, the AODV needs more time for searching a new route compared with DYMO. So, the delay of AODV is higher than DYMO. The route searching time of DYMO is almost the same with OLSR protocol. However, DYMO have better goodput than OLSR. Thus, DYMO has a better performance than AODV and OLSR protocols.

In Fig. 7.14 are shown the simulation results of PDR for DSR protocol. For this protocol the maximum number of hops is 14. Comparing with simulation results of DYMO in Fig. 7.13, we can see that both on-demand protocols drop a considerable number of packets during the route discovery process between the source and destination.

As can be seen from the figures, DSR protocol has better PDR than DYMO protocol. DSR protocol use more hops to keep the communication between nodes compared with DYMO and this is reflected to a bigger delay of DSR. DSR protocol carries full routing information which cause significant overhead. Also, the packet header size grows with route length due to source routing. DYMO does not find many routes (as DSR protocol does) because DYMO only saves link disjoint routes. DSR uses any route from its cache and so the average route length is longer.

In Fig. 7.15 are shown the simulation results of PDR for DSDV protocol. For this protocol the maximum number of hops is 3. As can be seen from the Fig. 7.15, the DSDV protocol has a low packet delivery rate. This is because the usage of stale routes in case of broken links. In DSDV the existence of stale route does not imply that there is no valid route to the destination. DSDV maintains only one route per destination and consequently, each packet that the MAC layer is unable to deliver is dropped because there are no alternate routes. DYMO different from DSDV tries to use more hops to keep the communication between nodes. Thus, the route maintenance mechanism of DYMO is better than DSDV.

7.3.2 Scenario 2: Effect of Speed and Node Density on the Network Performance

We made extensive simulations to evaluate the performance of DYMO protocol for different node speeds and densities. In these simulations, all vehicles move in a circular line. Number of vehicles is considered 20, 40, 60, 90 and for each scenario three speed cases are used: 27km/h, 81km/h and 108 km/h. The incidence of decelerating is considered 30 %. The simulation time is 100 seconds. In all simulations, the receiving node is node 0 and all other nodes are sending nodes.

We prepared each simulation based on nodes ID and the mobility pattern is the same. In order to evaluate the performance of each protocol, 5 packets per second as a CBR traffic were transmitted between 10 seconds and 90 seconds. As transport protocol we used UDP. As Traffic Model, CBR traffic sources and 512-byte data packets are used. Other simulation parameters are shown in Table 7.2.

As evaluation metrics, we use the PDR and the Average Packet Delivery Ratio (Av. PDR). The Av. PDR is the average PDR of the network and is calculated by the following

Table 7.2: Simulation parameters for scenario 2.

Parameter	Value
Network Simulator	NS2
Routing Protocol	DYMO
Simulation Time	100 s
Simulation Area	3000 m Circuit
Number of Nodes	20, 40, 60, 90
Max speed (km/h)	27, 81, 108
Incidence of Decelerating	30%
Traffic Source/Destination	Deterministic
DATA TYPE	CBR
Packets Generation Rate	5 packets/s
Packet Size	512 bytes
MAC Protocol	IEEE802.11 DCF
MAC Rate	2 Mbps
RTS/CTS	None
Transmission Range	250 m
Radio Propagation Models	Two-ray Ground
Hello_{DYMO} Interval	1 s
Network Simulation Time	10 s - 90s

formula:

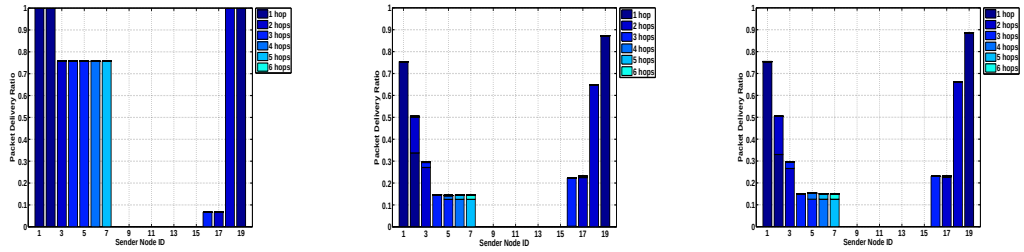
$$Av.PDR = \frac{\sum_{i=1}^{n} PDR_n}{n-1} \tag{7.1}$$

where n is the number of nodes in the network.

In Fig. 7.16 are shown the simulation results of DYMO protocol for PDR of every node when the number of vehicles in the network is 20. The figures show also the number of hops used to send the packets from the sender nodes to the receiver node. In Fig. 7.16(a), the maximum speed of the vehicles is considered 27 km/h and in Fig. 7.16(b) and Fig. 7.16(c), 81 km/h and 108 km/h, respectively. For the same number of vehicles and different maximum speeds, the maximum number of hops is 6 hops.

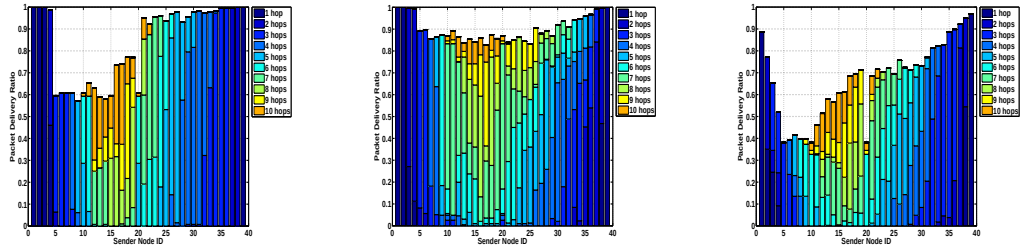
The PDR from node 8 to node 15 is zero. This happens because there is a big distance between these nodes and the receiving node. Only when source node and destination node are very close to each other, a valid route may be established. In this situation, the unique routes than can be established consist of few hops. When the speed is low, the network topology does not change very fast so DYMO protocol can easily establish and maintain the routes.

In Fig. 7.17 the number of vehicles is considered 40. In Fig. 7.17(a), the maximum speed is 27 km/h and all nodes in the network can communicate with the receiver. Some nodes between node 4 and node 20 after transmitting some data go out of the communica-



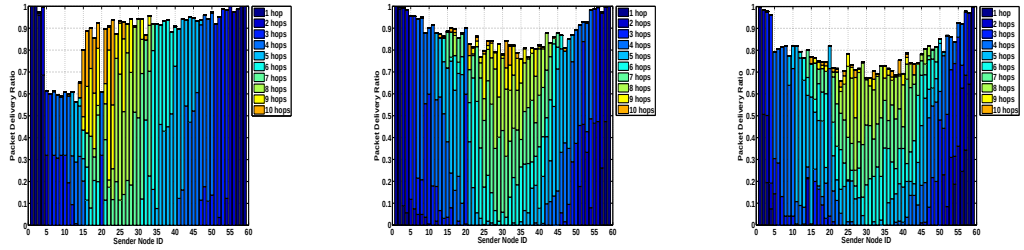
(a) Maximum speed 27 km/h (b) Maximum speed 81 km/h (c) Maximum speed 108 km/h

Figure 7.16: DYMO simulation results for 20 nodes.



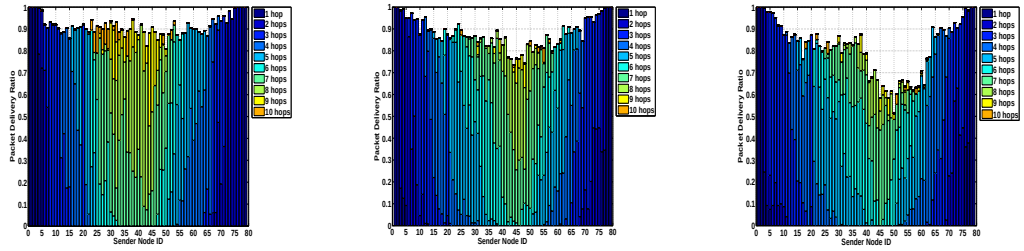
(a) Maximum speed 27 km/h (b) Maximum speed 81 km/h (c) Maximum speed 108 km/h

Figure 7.17: DYMO simulation results for 40 nodes.



(a) Maximum speed 27 km/h (b) Maximum speed 81 km/h (c) Maximum speed 108 km/h

Figure 7.18: DYMO simulation results for 60 nodes.



(a) Maximum speed 27 km/h (b) Maximum speed 81 km/h (c) Maximum speed 108 km/h

Figure 7.19: DYMO simulation results for 80 nodes.

Table 7.3: Average PDR for scenario 2.

No. of nodes	Speed (km/h)	Av. PDR
20	27	0.48
	81	0.24
	108	0.2
40	27	0.8
	81	0.85
	108	0.67
60	27	0.86
	81	0.87
	108	0.78
80	27	0.91
	81	0.88
	108	0.79

tion range, which results in the decrease of PDR value. In Fig. 7.17(b) the maximum speed is considered 81 km/h. In this case, the PDR is very high for all nodes. In Fig. 7.17(c) the PDR is lower compared with Fig. 7.17(a) and Fig. 7.17(b). This happens because when the nodes moves fast, the routes are broken often.

We increased the number of vehicles in the network to 60 and the simulation results for different speeds are shown in Fig. 7.18. For the speeds 27 km/h and 81 km/h we got better results (see Fig. 7.18(a) and Fig. 7.18(b)).

When the number of nodes increases to 80 (see Fig. 7.19), the PDR has better values in case of the speed 27 km/h (Fig. 7.19(c)). With the increasing of the speed, the route change becomes more frequently, which causes the link failure. For this reason, DYMO protocol uses more control packets (like RERR) and the routing table is updated more often, so the PDR value is decreased.

From all figures, we can see that for nodes which are located in the middle of circle, the results of PDR are lower because the distance between these nodes and the receiver is higher than other nodes.

From all simulation results, we can find that the PDR values are good for the speeds 27 km/h and 81 km/h. However, when the maximal speed is 108 km/h, the probability to maintain a valid route is low and DYMO protocol tents to use less communication hops.

In Table 7.3 are shown the values of Av. PDR. From the results we can notice that the Av. PDR increases with the increase of the network density. The best results for Av. PDR are reached for low and medium speed.

Table 7.4: Simulation parameters for scenario 3.

Parameter	Value
Network Simulator	NS2
Routing Protocol	DYMO
Simulation Time	100 s
Simulation Area	3000 m Circuit
Number of Nodes	40
Max speed	108 (km/h)
Incidence of Decelerating	20%
Traffic Source/Destination	Deterministic
Application type	FTP
Packet Size	512 bytes
MAC Protocol	IEEE802.11 DCF
MAC Rate	2 Mbps
RTS/CTS	None
Transmission Range	250 m
Radio Propagation Models	Two-ray Ground
Hello_{DYMO} Interval	1 s
Network Simulation Time	10 s - 90s

7.3.3 Scenario 3: Simulation Results for TCP Traffic

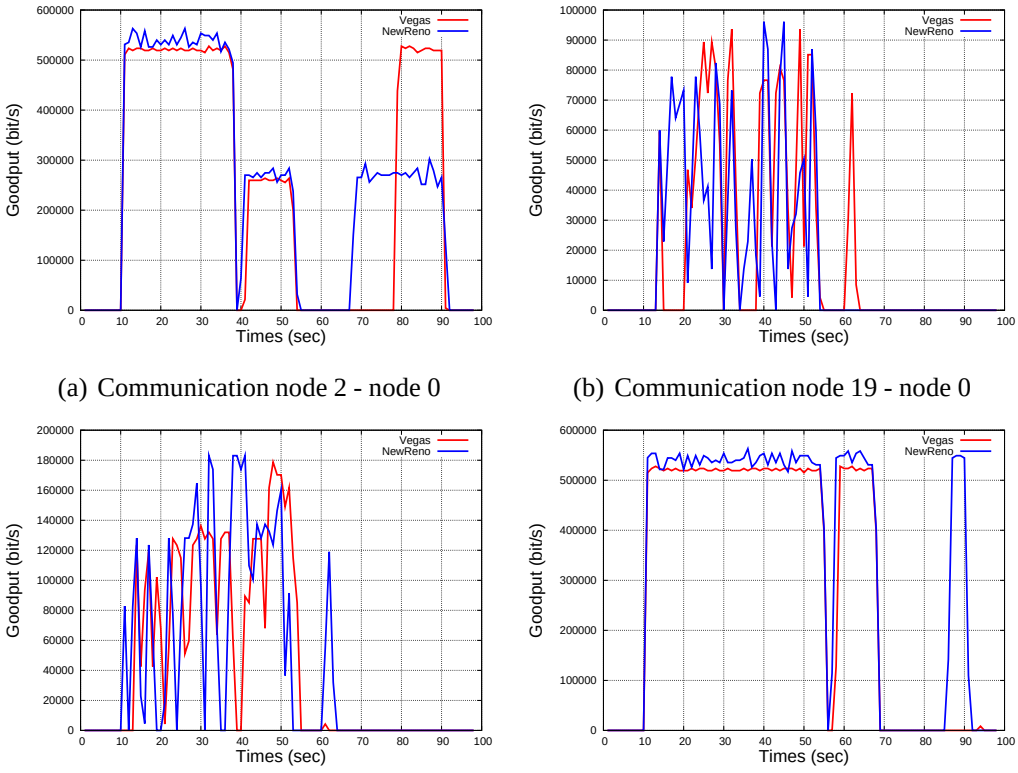
In these simulations, all vehicles move in a circular line. Number of vehicles is considered 40 and the maximal speed 108 km/h. The incidence of decelerating is considered 20%. The simulation time is 100 seconds. The receiving node is node 0 and nodes 2, 19, 31 and 39 are sending nodes.

We prepared each simulation based on nodes ID and the mobility pattern for all scenarios is the same. In order to evaluate the performance of DYMO protocol, TCP traffic were transmitted between 10 seconds and 90 seconds. As transport protocol we used FTP. As evaluation metric, we use Goodput. We also took in consideration cwnd, seqno and sshthresh. As traffic model the TCP traffic sources and 512-byte data packets are used. Other simulation parameters are shown in Table 7.4.

We evaluate the goodput of TCP NewReno and TCP Vegas over DYMO for different nodes communication. The simulation results are shown in Fig. 7.20. We use Goodput as evaluation metric because it is a direct indicator of network performance.

In Fig. 7.20(a) and Fig. 7.20(d), we can see that for both TCP NewReno and TCP Vegas the value of Goodput is high. In Fig. 7.20(b) and Fig. 7.20(c), the communicating nodes are far from each other and the Goodput is unstable.

In Table 7.5 is shown the Av. Goodput of NewReno and Vegas for different nodes communication. As we can see from the table, NewReno has better Av. Goodput. For



(a) Communication node 2 - node 0

(b) Communication node 19 - node 0

(c) Communication node 31 - node 0

(d) Communication node 39 - node 0

Figure 7.20: Simulation results of Goodput for NewReno and Vegas.

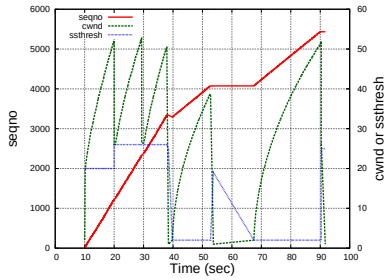
Table 7.5: Average Goodput for scenario 3.

SrcID - DestID Communication	TCP NewReno Av. Goodput [kbps]	TCP Vegas Av. Goodput [kbps]
node 2 - node 0	311	264
node 19 - node 0	42	19
node 31 - node 0	78	53
node 39 - node 0	404	307

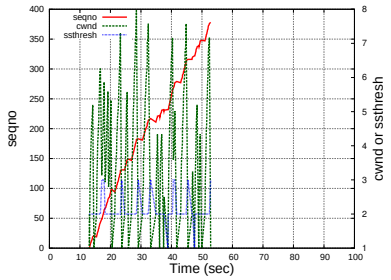
both TCP NewReno and Vegas, the Av. Goodput is decreased significantly when the distance between communicating nodes is increased.

Fig. 7.21 and Fig. 7.22 present the combined outline of seqno, cwnd and ssthresh vs. time for TCP NewReno and TCP Vegas, respectively.

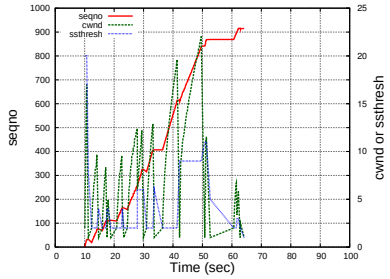
TCP sender uses the slow-start and congestion avoidance phases to control the amount of outstanding data being injected in the network. TCP makes use of seqno to ensure that the packets arrive in the correct order. The cwnd determines the number of bytes that can



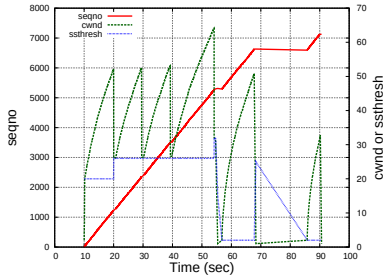
(a) Communication node 2 - node 0



(b) Communication node 19 - node 0

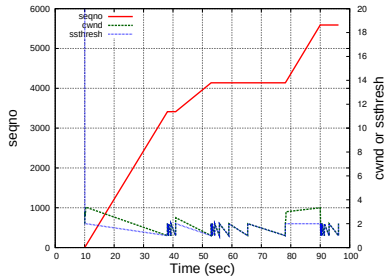


(c) Communication node 31 - node 0

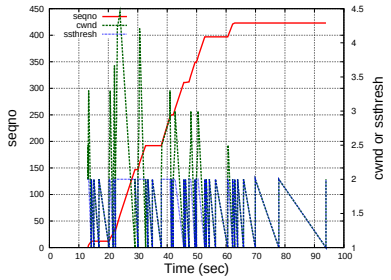


(d) Communication node 39 - node 0

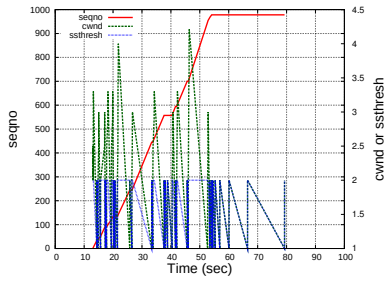
Figure 7.21: Simulation results of seqno, cwnd and ssthresh vs. time for newReno.



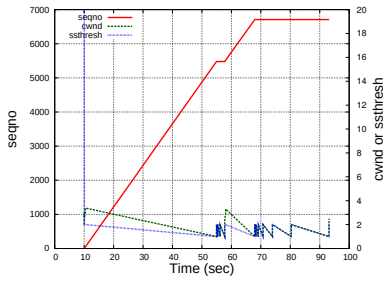
(a) Communication node 2 - node 0



(b) Communication node 19 - node 0



(c) Communication node 31 - node 0



(d) Communication node 39 - node 0

Figure 7.22: Simulation results of seqno, cwnd and ssthresh vs. time for Vegas.

be outstanding at any time. The cwnd and ssthresh define the slow start and congestion

avoidance. If cwnd is less than or equal to ssthresh, TCP is in slow start, otherwise TCP is performing congestion avoidance.

TCP NewReno stays in fast recovery until all packet losses in window are recovered. It can recover 1 packet loss per RTT without causing a timeout. TCP NewReno continue to increase the window size until packet losses occur. Regarding the cwnd size, TCP-NewReno operates at a larger window size compared with TCP Vegas which is approximately 54 packets. This explains why it has a higher goodput. TCP NewReno uses larger cwnd compared with TCP Vegas (see Fig. 7.21 and Fig. 7.22). This indicates the better utilization of available bandwidth.

TCP Vegas congestion avoidance mechanisms detects congestion at early stage. TCP Vegas controls its window size by observing RTTs (Round Trip Time) of packets that the connection has sent before. TCP-Vegas frequently goes into the slow start phase where cwnd starts at 1. TCP Vegas uses an estimate of the propagation delay (baseRTT) to adjust its window size. Rerouting a path may change the propagation delay of the connection and this could result in decreasing of Goodput.

Because of the mobility, frequent link failure happens in the network and also packet loss (see Fig. 7.20(a) during time 55 sec and 78 sec). If we refer to Fig. 7.21(a) and Fig. 7.22(a), we can see that the seqno does not change until the recovery for both TCP variants. The TCP NewReno has faster recovery compared with TCP Vegas and it performs better in VANETs.

The TCP Vegas can not differentiate congestion from packet losses due to link failures, so it reduces the size of cwnd. This is reflected in the decreasing of the Goodput.

7.4 Performance Evaluation of Different Routing Protocols Using NS3

7.4.1 Scenario 4: Performance Evaluation of OLSR and DSDV Using NS3

In these simulations, 30 vehicles move in a circular line. The maximum speed of the vehicles is 135 km/h. The incidence of decelerating is considered 20 %. The simulation time is 100 seconds. In every scenario, the receiving node is node 0 and nodes 5, 10, 15 and 20 are sending nodes.

Table 7.6: Simulation parameters for scenario 4.

Parameter	Value
Network Simulator	NS3
Routing Protocol	OLSR, DSDV
Simulation Time	100 s
Simulation Area	3000 m Circuit
Number of Nodes	30
Max speed (km/h)	135
Incidence of Decelerating	20%
Traffic Source/Destination	Deterministic
DATA TYPE	CBR
Packets Generation Rate	5 packets/s
Packet Size	512 bytes
MAC Protocol	IEEE802.11 p
MAC Rate	2 Mbps
RTS/CTS	None
Transmission Range	250 m
Radio Propagation Models	Two-ray Ground
Hello_{OLSR} Interval	1.5 s
Hello_{DSDV} Interval	1.5 s
Network Simulation Time	10 s - 90s

In order to evaluate the performance of each protocol, 5 packets per second as a CBR traffic were transmitted between 10 seconds and 90 seconds. As transport protocol we used UDP and as networking protocols OLSR and DSDV. As evaluation metrics, we use Throughput and PDR.

In our simulations we used IEEE 802.11p standard and TwoRayGroundPropagation-LossModel. IEEE 802.11p is an approved amendment to the IEEE 802.11 standard to add wireless access in vehicular environments (WAVE). It defines enhancements to 802.11 required to support Intelligent Transportation Systems (ITS) applications. The 802.11p standard is based on the 802.11 architecture, but version ”p” is aimed at communications between vehicles and between them and fixed infrastructure. This new technology uses the 5.9 GHz band in various propagation environments: vehicle, open, urban, etc. This standard defines the WAVE as the signaling technique and interface functions that are controlled by the physical layer (MAC) devices where the physical layer properties change rapidly and where the exchanges of information have a short duration. The purpose of this standard is to provide a set of specifications to ensure interoperability between wireless devices trying to communicate in rapidly changing environments and in particular time periods.

Table 7.7: Average Throughput for scenario 4.

Communicating nodes SrcID - DstID	OLSR Av. Throughput	DSDV Av. Throughput
node5 - node0	22 kbps	18 kbps
node10 - node0	11 kbps	8 kbps
node15 - node0	6 kbps	6 kbps
node20 - node0	13 kbps	12 kbps

TwoRayGroundPropagationLossModel considers the direct path and a ground reflection path. The received power at distance t is calculated with the following equation.

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}$$

where h_t and h_r are the heights of the transmit and receive antennas.

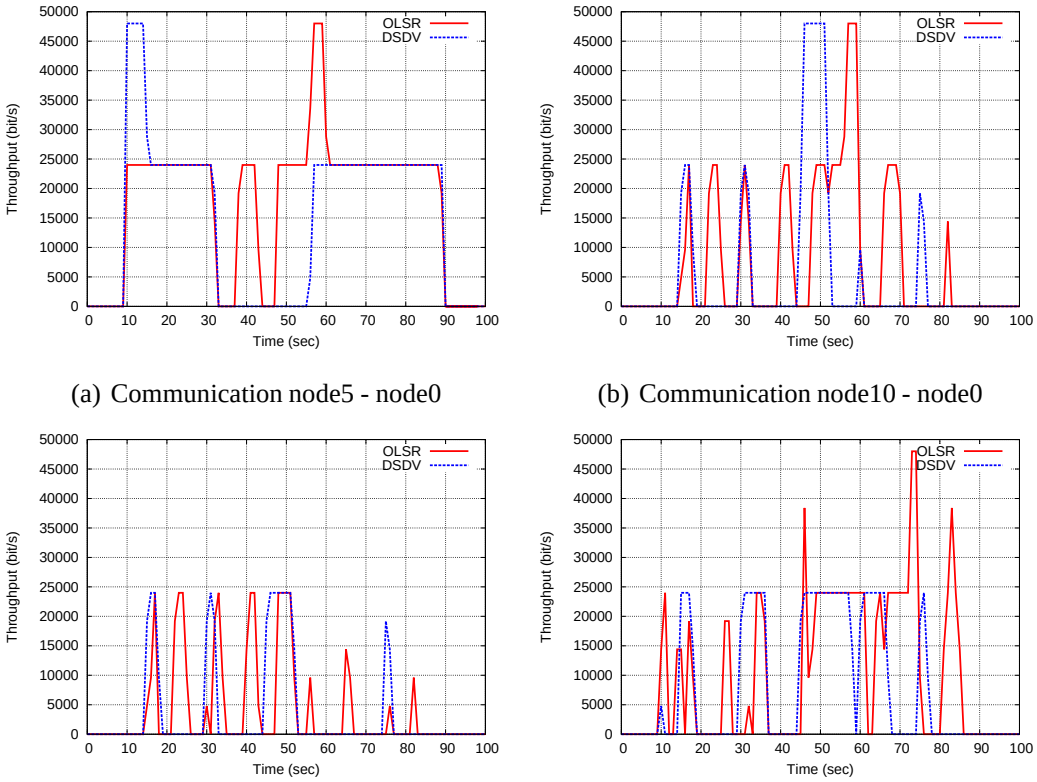
The simulations are carried out in Ubuntu 11.10 and as traffic model we used CBR traffic sources and 512-byte data packets. Other simulation parameters are shown in Table 7.6.

We present here the simulation results for OLSR and DSDV protocols by means of CAVENET and NS3.

In Fig. 7.23(a) are shown the simulation results of throughput for the communication between node 5 and node 0. Both protocols have disconnections, but the disconnection time of DSDV is longer than OLSR. OLSR protocol, after the disconnection in second 33 can find faster than DSDV a new route. DSDV converges slowly before detecting a valid route. During this time, a lot of packets are dropped and the throughput is low. In OLSR, the link duration and path stability is higher than DSDV due to MPR mechanism that reduces routing overhead.

The Av. Throughput of OLSR is higher than DSDV (see Table 7.7). DSDV protocol uses the table-driven approach for maintaining route informations. But, it is not adaptive with the route changes that occur during the high mobility.

In Fig. 7.23(b) are shown the simulation results of throughput for the communication between node 10 and node 0. In this case, for both protocols there are many oscillations of the throughput and the number of disconnections is high. The Av. Throughput of OLSR in this scenario is 11 kbps and it is higher compared with DSDV (8 kbps).



(a) Communication node5 - node0

(b) Communication node10 - node0

(c) Communication node15 - node0

(d) Communication node20 - node0

Figure 7.23: Simulation results for different node communications.

The simulation results of throughput for the communication between node 15 and node 0 are shown in Fig. 7.23(c) Because the number of the nodes in the network is not high and the distance between node 15 and node 0 is long, the disconnections and network partitioning occurs. In this case, both protocols have the same performance and the Av. Throughput is low (6 kbps).

In Fig. 7.23(d) are shown the simulation results of throughput for the communication between node 20 and node 0. In this case, the number of intermediate nodes between source and destination is nine nodes and the initial distance for these nodes is the same with the scenario in Fig. 7.23(b). From Table 7.7, we can see that the Av. Throughput for communication between node 20 and node 0 is higher than between node 10 and node 0. In this case, the performance of routing is also effected by the direction of the movement.

The simulation results for PDR of the above scenarios are shown in Fig. 7.24. When node 5 and node 0 communicate between each other, both protocols have a high PDR. In this case, the distance between the communicating nodes is small and the sender easily finds valid routes to send the packets. For the communication between node 15 and node

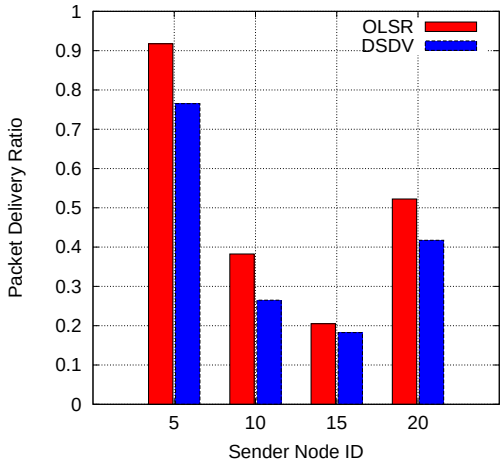


Figure 7.24: PDR simulation results for OLSR and DSDV.

Table 7.8: Simulation parameters for scenario 5.

Parameter	**Value**
Network Simulator	NS3
Routing Protocol	OLSR, AODV
Simulation Time	50 s
Simulation Area	200 m x 200 m
Simulation Scenario	Crossroad
Number of Nodes	32
Max speed (km/h)	135
Incidence of Decelerating	30%
Number of Connections	10
DATA TYPE	CBR
Transport Protocol	UDP
Packet Size	512 bytes
MAC Protocol	IEEE802.11 p
MAC Rate	2 Mbps
RTS/CTS	None
Transmission Range	250 m
Radio Propagation Models	Two-ray Ground
Network Simulation Time	5 s - 45 s

0 because the distance between nodes is long and the speed of vehicles is high, the link breakage happens and this is reflected in the low value of PDR for both protocols.

7.4.2 Scenario 5: Effect of Transmission Rate on the Network Performance

For simulations, we used OLSR and AODV protocols and sent multiple CBR flows over UDP. Ten connections are created and the same pairs source-destination are used for both protocols. The simulation time is 50 sec.

We present the simulation results for OLSR and AODV protocols done by means of CAVENET and NS3. For performance evaluation, we use three metrics: the average PDR, throughput and delay. The simulation parameters are shown in Table 7.8.

In Fig. 7.25 are shown the simulation results of PDR vs. transmission rate for OLSR and AODV protocols. For transmission rates less than 200 kbps, all packets sent from the source are received at the destination and the PDR is maximal for both protocols. When the transmission rate is higher than 200 kbps, the PDR is decreased for both protocols. The PDR of AODV is smaller than OLSR. This is related with the fact that AODV different from OLSR is not equipped with MPR nodes that facilitates the route discovery. When the transmission rate is 850 kbps, the PDR is low and almost the same for both protocols.

In Fig. 7.26 are shown the simulation results of throughput vs. transmission rate. When the transmission rate is smaller than 200 kbps, the PDR is maximal and the throughput is theoretical. After 300 kbps, the throughput of AODV is smaller than OLSR and the difference between is big. When the transmission rate is 750 kbps to 850 kbps, the PDR of both protocols are almost the same (see Fig. 7.25) but the difference of the throughput is high. This happens because the effect of mobility and delay. The throughput of OLSR protocol for transmission rates from 300 kbps to 850 kbps is almost 2.1 Mbps and has not many oscillations.

The simulation results for delay vs. transmission rate are shown in Fig. 7.27. For the transmission rates up to 200 kbps, the delay is less than 0.1 sec and both protocols can be used for real time applications such as safety messages. For other values of transmission rate, the delay is higher than 1 sec. In this case these protocols can be used for applications that tolerate this delay such as streaming and entertainment.

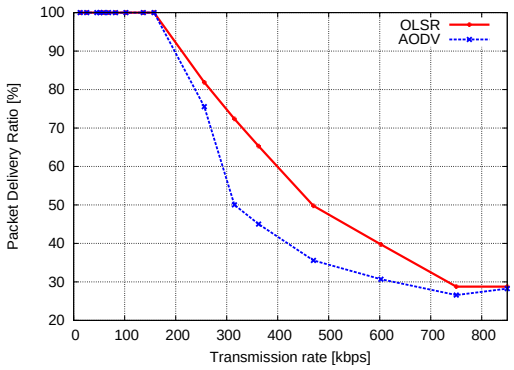


Figure 7.25: Simulation results of PDR for different transmission rates.

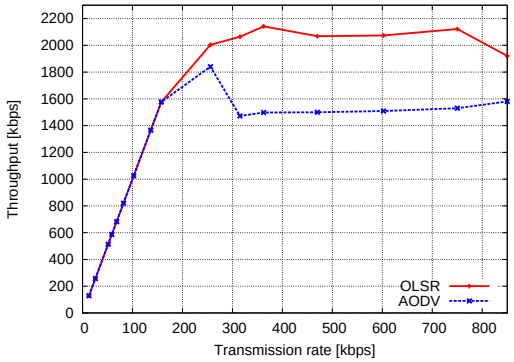


Figure 7.26: Simulation results of throughput for different transmission rates.

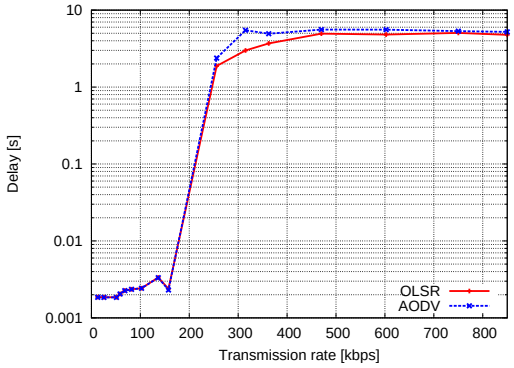


Figure 7.27: Simulation results of delay for different transmission rates.

7.4.3 Scenario 6: Effect of Number of Connections on the Network Performance

For this scenario, we used IEEE 802.11p standard and TwoRayGroundPropagationLoss-Model. The simulations are carried out in Ubuntu 11.10 and as traffic model is used CBR

Table 7.9: Simulation parameters for scenario 6.

Parameter	Value
Network Simulator	NS3
Routing Protocol	OLSR, AODV
Simulation Time	50 s
Simulation Area	200 m x 200 m
Simulation Scenario	Crossroad
Number of Nodes	32
Max speed (km/h)	135
Incidence of Decelerating	30%
Traffic Source/Destination	Deterministic
DATA TYPE	CBR
Packet Size	512 bytes
MAC Protocol	IEEE802.11 p
MAC Rate	2 Mbps
RTS/CTS	None
Transmission Range	250 m
Radio Propagation Models	TwoRayGround
Network Simulation Time	5 s - 45 s

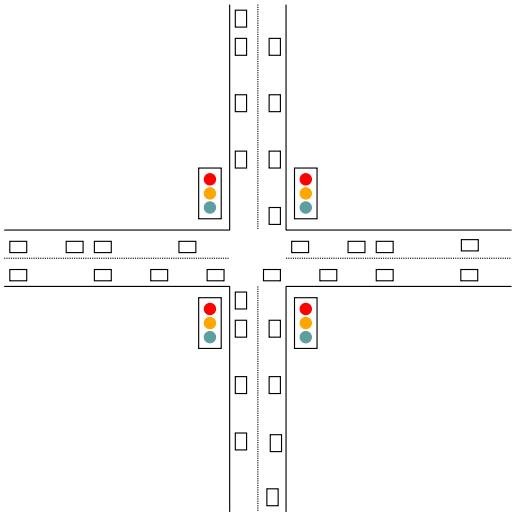


Figure 7.28: Simulation scenario.

traffic sources and 512-byte data packets. Other simulation parameters are shown in Table 7.9.

For simulations, we consider a two lane crossroad scenario as shown in Fig. 7.28. During the simulations, 32 vehicles move in roads according to CA model and respecting the traffic light. Vehicles are randomly positioned at the begin of the intersection. The maximum speed of the vehicles is 135 km/h. The vehicles accelerate and decelerate according to CA model. The incidence of decelerating is considered 30%. The vehicles

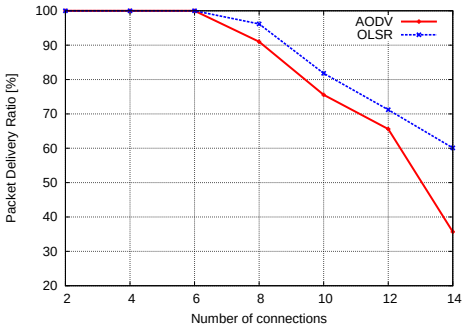


Figure 7.29: Simulation results of PDR for different number of connections.

follow each other and when a vehicle reach at the end of the road it is shifted at the begin of the same road. When a vehicle goes near the intersection, it checks for the state of the traffic light. The traffic lights change the state in a constant time period. If the traffic light is red, it decelerates and stop. If the traffic light is green it reduce the speed and enter to the intersection. The cars in front of the crossroad and in the junction will have a speed of 27 km/h (minimal).

For simulations, we used OLSR and AODV protocols and sent multiple CBR flows over UDP. Different number of connections are created and the same pairs source-destination are used for both protocols. The simulation time is 50 sec.

As simulators we use CAVENET and NS3. For performance evaluation, we use three metrics: the average PDR, throughput and delay. We carried out simulations for 2, 4, 6, 8, 10, 12 and 14 connections in the network. The CBR data sent in one connection during all simulation time is 256 kbps. Based on the number of connections, the total rate transmitted in the network changes.

In Fig. 7.29 are shown the simulation results of PDR vs. number of connections for OLSR and AODV protocols. For less than 6 connections in the network, both protocols have a maximal PDR and a very good performance. For number of connection from 6 to 12 connections, because of the amount of data sent in the network is increased, the PDR for both protocols is decreased (however it is higher than 65% for both protocols). In this case, OLSR protocol has better PDR. For 14 connections in the network, the information sent between the pairs source-destination is very high and a lot of packets are dropped. The PDR of AODV is highly decreased compared with OLSR.

In Fig. 7.30 are shown the simulation results of throughput vs. number of connection. When the number of connections in the network is less than 6, the PDR is maximal and the throughput is theoretical. For number of connections between 6 and 12, the throughput

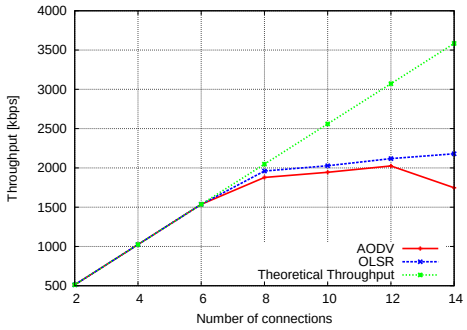


Figure 7.30: Simulation results of throughput for different number of connections.

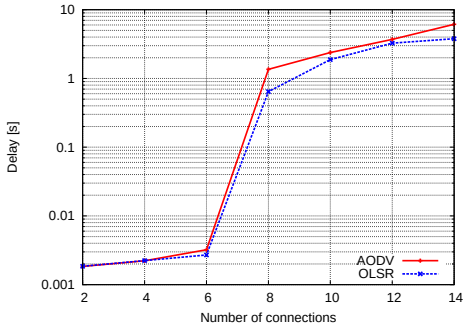


Figure 7.31: Simulation results of delay for different number of connections.

is slowly increased for both protocols. For number of connections 14, the throughput of OLSR continues to increase slowly but the throughput of AODV is decreased.

The simulation results for delay vs. number of connection are shown in Fig. 7.31. For number of connections less than 8, the delay for both protocols is less than 1 s and they can be both used for real time applications, but after 8 connections, the delay is high.

Chapter 8

Evaluation Results and Solutions for P2P Systems

In this chapter, we give experimental results for secure robot control and the simulation results for fuzzy-based data replication system.

8.1 Results of JXTA-Overlay for Secure Robot Control

In this section are presented some experimental results for robot control in JXTA-Overlay. During the experiments are observed only the messenger primitives because the login and connect primitives just happens once for the full session. The command of the robot control is sent as simple message using chat service. All the experiments have been run in the nodes of a small network as shown in Fig. 8.1 and peer node specifications are shown in Table 8.1.

Specifications of the robot

For the experiments is used KHR-1 robot with the following specifications: Heigh 340 mm, Width 180 mm, depth 80 mm and Weight 1.2 kg. The robot has 17 servo-motors as shown in Figure 8.2. To create the motions we used HeartToHeart (motion editor for RCB-1). A snapshot of this program is shown in Fig. 8.3

Table 8.1: Peer node specifications.

Peer node	**Characteristics**
Broker	AMD Athlon (tm) 64 Processor 3200+2.01 GHz, 100 GB RAM
Peer 1	Pentium M Centrino 1.2 GHz, 512 MB RAM
Peer 2	Intel Centrino Duo 1.3 GHz, 2GB RAM
Connection	100 Mbps

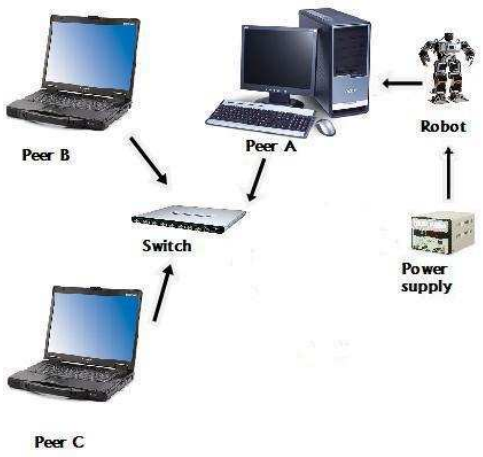


Figure 8.1: P2P Robot control system.

Time synchronization

Before starting the experiments, the time synchronization is done by FGTime Sync version 1.0.0.4, which is a software developed by FreeStone Group. The usage of FG Time Sync utility keep all computers on LAN with the exact current time. It can work as client or server. Using this software are synchronized the times on Desktop PC and two laptops. In the Desktop PC is used “FG Time Server” and on two laptops the “FG Time Sync”. A snapshot of FGTime Sync is shown in Fig. 8.4.

Robot control using sendMesgPeer and secureMsgPeer primitives

We experimentally studied the time for Robot Control for two different cases:

- Using *sendMsgPeer* primitive,
- Using *secureMsgPeer* primitive.

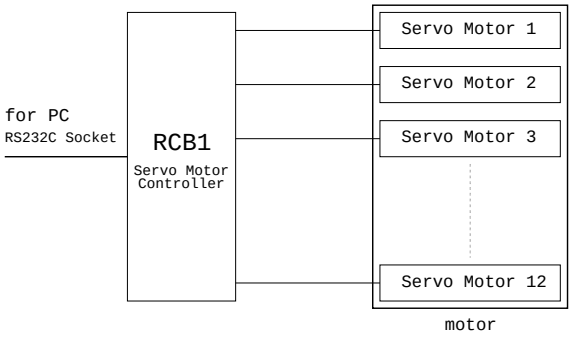


Figure 8.2: KHR-1 Robot interface.

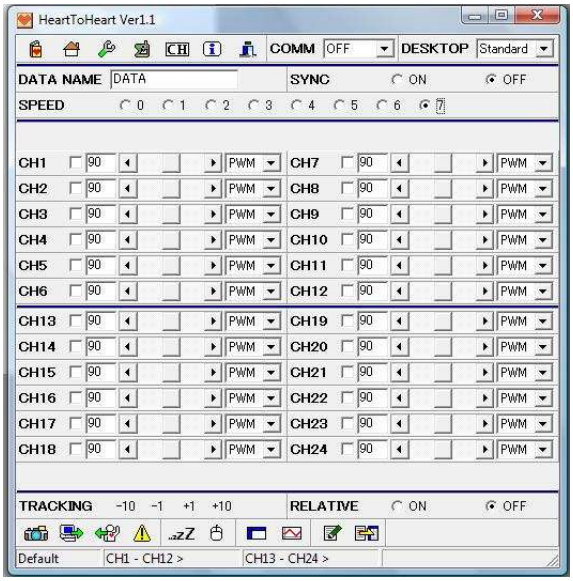


Figure 8.3: Snapshot of HeartToHeart program for robot control.

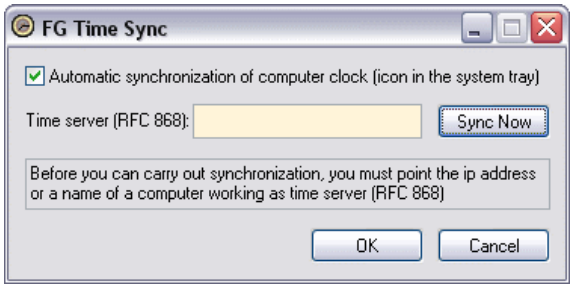
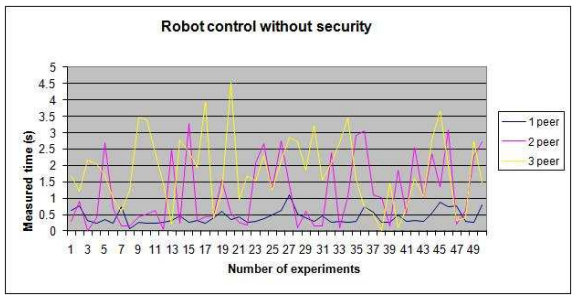
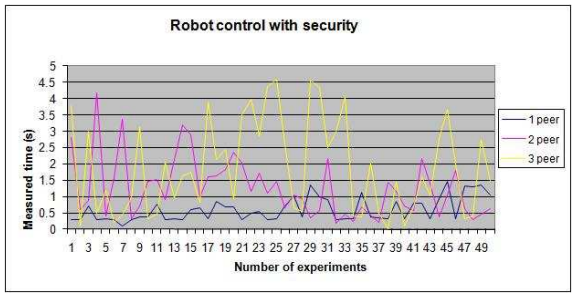


Figure 8.4: Snapshot of FGTime Sync.



(a) Experimental results for robot control using sendMsgPeer primitive.



(b) Experimental results for robot control using secureMsgPeer primitive.

Figure 8.5: Experimental results for robot control.

These messenger primitives define how to directly exchange simple text messages between end-users, such as a chat service, without requiring broker intervention. The command of robot control is sent as simple text message.

First, we deployed a network with only one peer (this peer plays the role of broker). The robot is connected with this peer via RS232C connector, which is a legal interface and many devices have implemented it. Then, the time of robot control is measured using both primitives.

Two other scenarios were realized adding two other peers to the network and the times of robot control were measured again. In Fig. 8.5 are shown the experimental results for robot control. In Fig. 8.5(a) is shown the time for robot control measured for 50 experiments, using sendMsgPeer primitive (non secure primitive). While, in Fig. 8.5(b) the time for Robot Control when is used secureMsgPeer primitive (secure primitive).

In Fig. 8.6 is shown the average time for robot control using and not using security. As can be observed, the difference between the average time of unsecure robot control and secure robot control is small and with increase of number of peers the difference time

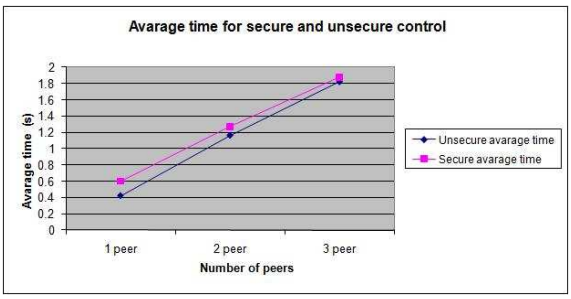


Figure 8.6: Average time for secure and unsecure robot control.

Table 8.2: Statistical results.

Peer node	Primitives	Time under 1s	Time 1-3s	Time over 3s
1 Peer	sendMsgPeer	98%	2%	0%
	secureMsgPeer	82%	18%	0%
2 Peer	sendMsgPeer	56%	38%	6%
	secureMsgPeer	48%	46 %	6 %
3 Peer	sendMsgPeer	62%	22%	16%
	secureMsgPeer	40%	36%	24%

decreases. In Table 8.2 are shown the statistical data for above experiments. In the case that is used only one peer, the major part of measured times for both cases (using unsecure and secure primitives) is less than 1s.

The average time for the unsecure control is 0.42 s and when it is used security 0.6 s. When another peer joins the network, the time for robot control for both primitives increases. The join of other peers in the network increase the number of requests that the client primitives send to the broker. In this situation, the broker should manage all requests of the client primitives. This cause an overhead in the time of robot control. From the experimental data is observed that some of the values of the time measured are over 3 seconds. The reasons are explained in the following.

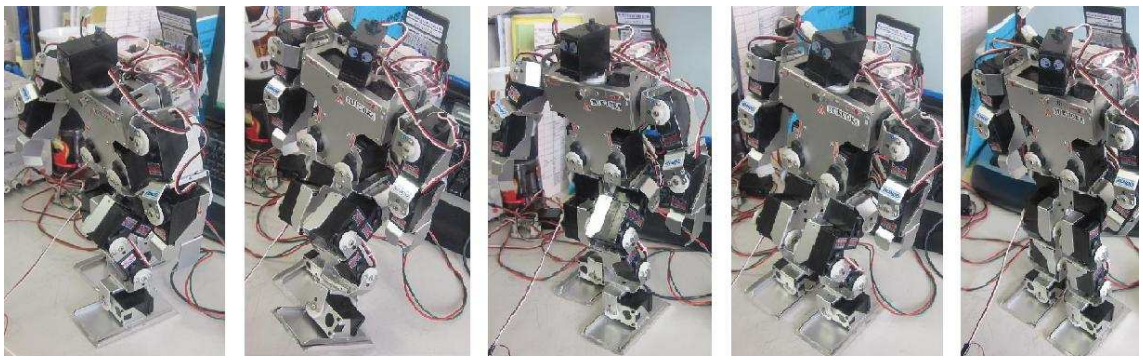
- This situation can happen because the broker is very busy. The mission of the broker node is to manage and control all the requests received from the client peers. Broker manages the events produced by such requests and propagates them to superior levels of overlay. Broker performs a big number of tasks because Broker functions are always called as a result of client primitives. Some of the broker

functionalities are: event management, controlling the resources connected to the broker, maintaining the organization of the resources in groups, finding the best resource for file sharing, finding the best resource for executing submitted tasks and maintaining updated statistic information.

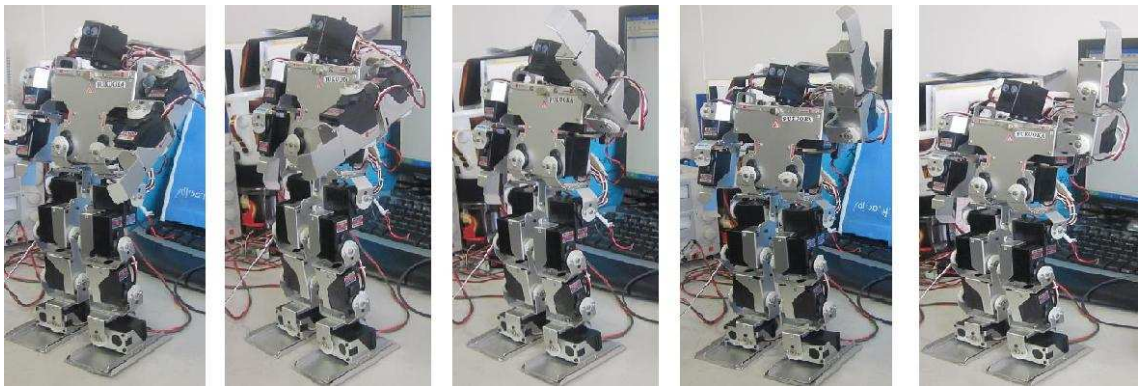
- Another reason is that JXTA-Overlay platform used in experiments has no parameter tuning. P2P applications involve many parameters, whose values cannot be arbitrarily fixed and they require a careful tuning to achieve expected performance. These parameters are divided in: parameters related to the Broker peers and parameters related to the Client peers. Broker peer parameters include discovery time, publishing time, advertisement time for broker's statistic. High values of these parameters could lead to an inconsistent state of the network. Also, small values of the parameters could cause congestion of Brokers peers due to very frequent updates. In the case of the Client peer the parameters include time parameters related to the peer's advertisement, publishing and expiration. Small values of these parameters could cause saturation of client peers due to very frequent generation of advertisement and high traffic in the network due to the publishing and propagation of the peer's information. These parameters directly relate to the time interval for checking the state of the network, performing action and sending state information to the network.

- Network latency is the delay that is introduced by the network. Network situation such as congestion can be a major contributing factor. Packets are forwarded from source to destination and traverse network link including LAN switches. The combination of various latency sources such as buffering, propagating, switching, queuing, and processing delays produces a complex and variable network latency profile. The sources of network latency include:

    1. Network interface delay for serialization, including signal modulation and data framing (packetizing).
    2. Network processing delays from gateways, firewalls, or other network elements.
    3. Signal propagation delay, which is the time it takes for signals to travel in the physical cable.
    4. Switch delay is the time to shift packets from an ingress port to an egress port.



(a) Robot walking.

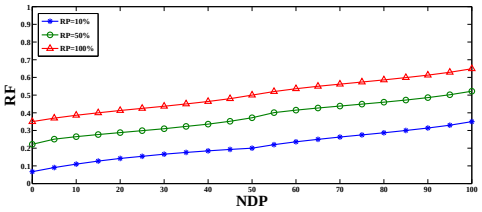


(b) Robot moving the left hand.

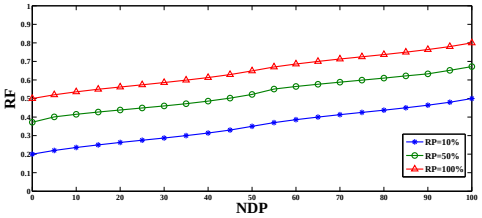
Figure 8.7: Robot movements.

5. Queuing delay occurs in the switch when packets from different ingress ports are heading to the same egress port concurrently.

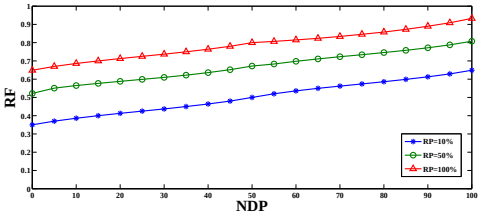
The average time for robot control when in the network is only one peer is 0.42 s for unsecure and 0.6 for secure primitive and for two peers 1.17 s and 1.27 s respectively. When in the network are three peers the time of robot control using unsecure primitive is 1.82 s and using secure primitive is 1.88 s. From this results we can conclude that the primitives of JXTA-Overlay *sendMsgPeer* and *secureMsgPeer* can be successfully used to control the robot in a smoothly way. Also, it can be used in applications that tolerate this range of time. In Fig. 8.7 are shown robot movements. In Fig. 8.7(a) from the left to the right, are shown the robot steps. In Fig. 8.7(b) from the left to the right the robot moves the left hand.



(a) Replication factor for SRP=10%.



(b) Replication factor for SRP=60%.



(c) Replication factor for SRP=100%.

Figure 8.8: Simulation results for data replication in P2P.

8.2 Simulation Results for Data Replication in P2P Networks

In this section, we evaluate the proposed fuzzy-based system for data replication in P2P networks by computer simulations. The simulations are carried out using MATLAB. In Fig. 8.8(a), we show the relation between RF and NDP, RP, SRP.

In this case the SRP is considered 10%. From the figure we can see that for RP=10% with the increase of the NDP the RF increases. Also, when the RP is increased the replication factor is increased.

In Fig. 8.8(b) are shown the simulation results for SRP=60%. As can be seen, the replication factor increases with the increase of SRP parameter.

In Fig. 8.8(c), the value of the SRP is considered 1. We can see that, with the increase of NDP and RP, the PR is increased. From the simulation results we conclude that the RF increases proportionally with increase of NDP, RP and SRP parameters.

Chapter 9

Concluding Remarks

9.1 Conclusions

In this thesis, we consider the P2P paradigm to build applications for Inter-Vehicular networks and robot control in wireless and wired environment. We implement and investigate the performance of different routing protocols in V2V scenarios using CAVENET simulation system, NS2 and NS3. We propose and experimentally evaluate the performance of the application of secure robot control using JXTA-overlay P2P platform.

In Chapter 1, we presented an introduction to the thesis and its content. Here we described the background, purpose and contribution of this study and the outline of this thesis.

In Chapter 2 of this thesis, we presented an introduction in P2P systems. We described the characteristics, requirements, the current state and problems of P2P systems. We also introduced data replication in P2P systems and different replication techniques to assure availability in case of peer failure.

In Chapter 3, we introduced vehicular networks and different mobility models of these networks. We classified different routing protocols for vehicular networks in three categories: unicast approach, multicast and geocast approach and broadcast approach. This chapter we also described in details the characteristics and functionality of AODV, OLSR, DYMO, DSR and DSDV routing protocols we used in simulations.

In Chapter 4, we gave the related work on VANET simulators. We also described the structure of our simulation system for Vehicular networks called CAVENET.

In Chapter 5, we introduced JXTA components and JXTA-Overlay P2P distributed platform which we have used to develop our application for secure robot control.

In Chapter 6, we described the design and implementation of our proposed applications based on JXTA-Overlay.

In Chapter 7, we gave evaluation results for V2V communication for different scenarios using CAVENET, NS2 and NS3. We evaluated and compared the performance of different routing protocols using different metrics.

In Chapter 8, we discussed the experimental results for secure robot control and the simulation results for fuzzy-based data replication system.

9.1.1 Conclusions for V2V Communication

We evaluated and compared the performance of different routing protocols in different scenarios using NS2. From the simulations we found the following results:

- We compared AODV, OLSR, DSDV, DSR and DYMO protocols considering PDR and goodput metrics. The simulation results showed that DYMO performed better than other protocols.

- We evaluated the effect of speed and node density on the network performance when sending UDP traffic considering DYMO routing protocol. When the speed of vehicles is low, the network topology does not change very fast so DYMO protocol easily establish and maintain routes. When the vehicles moves fast, routes are broken often, DYMO protocol uses more control packets (like RERR), the routing table is updated often and this is reflected on the decreasing of PDR. The Av. PDR is increased with the increasing of the network density and best results are achieved for low and medium speeds.

- We also evaluate and compare the performance of Vegas and NewReno TCP congestion avoidance algorithms considering DYMO protocol. TCP NewReno uses larger cwnd compared with TCP Vegas, this indicates the better utilization of available bandwidth. Regarding the cwnd size, TCP-NewReno operates at a larger window size compared with TCP Vegas which is approximately 54 packets. This explains why it has a higher goodput. TCP Vegas can not differentiate congestion from packet losses due to link failures, so it reduces the size of cwnd. This is reflected in the decreasing of the Goodput. TCP New Reno over DYMO offers better Goodput compared with TCP Vegas. TCP NewReno has faster recovery compared with TCP Vegas and it performs better in VANETs.

We also evaluated the performance of AODV, OLSR and DSDV routing protocols for different scenarios using NS3. From the simulation results, we conclude as follows:

- When the communicating nodes are near each other, the sender easily finds valid routes to send the packets and the Av. Throughput and PDR are high for both protocols. In OLSR, the link duration and path stability is higher than DSDV due to MPR mechanism that reduces routing overhead. DSDV converge slowly before detecting a valid route. During this time, a lot of packets are dropped and the throughput is low. When the distance between the communicating nodes is long and the speed of nodes is high, the topology of the network changes often, thus the routes are broken and the probability of maintain a valid route is low. In this case, both protocols have almost the same performance, low throughput and low PDR. The routing performance for both protocols is also effected by the direction of the movement. In general, OLSR has better throughput and PDR than DSDV protocol.

- We made extensive simulation using NS3 to evaluate the effect of transmission rate on the performance of the network and compare the performance of OLSR and AODV. For simulations, we considered IEEE 802.11p standard and TwoRay-GroundPropagationLossModel and sent multiple CBR flows over UDP. From the simulation results, we found out that for small transmission rates, all packets sent from the source are received at the destination and the PDR is maximal for both protocols. For high transmission rate the PDR is decreased for both protocols but, the PDR of AODV is smaller than OLSR. For small transmission rates, the PDR of both protocols is maximal and the throughput is theoretical. For big transmission rates, the OLSR has better throughput compared with AODV. The delay for small transmission rates is small for both protocols and they can be used for real time applications such as safety applications. For big transmission rates, the delay for both protocols is higher than 1 sec and in this case the protocols can be used for applications that tolerate this delay such as streaming and entertainment applications.

- We also made other simulations to evaluate the effect of the number of connections on the network performance. Also in these simulations we considered OLSR and AODV protocols. For small number of connections in the network, both protocols have a very good performance with maximal PDR, theoretical throughput and a very small delay. For average number of connection in the network, the performance of both protocols is decreased, but PDR is higher than 65%. In this case,

OLSR performs better than AODV. For big number of connections in the network, the amount of data sent in the network is very high, the network is congested, many packets are dropped and the delay is increased. In this case, the PDR and throughput of OLSR are higher than AODV.

9.1.2 Conclusions for JXTA-Overlay P2P Systems

We have experimentally measured the time for robot control, using two different primitives sendMsgPeer and secureMsgPeer. The objective was to see the performance as well as limitations of the P2P robot control when it is used as end-device and the cost of using security in robot control. This approach has been experimentally studied by deploying the JXTA-Overlay based application in a small network.

- The join of other peers in the network increase the number of requests that the client primitives send to the broker. In this situation, the broker should manage all requests of the client primitives, and this cause a overhead in the time of robot control. But, with the increasing of the number of peers in the network, the difference between the secure and unsecure average time for robot control is nearly at the same degree. From the experimental study some values of times over 3 s are measured during the robot control. However, tuning the parameters of this application can decrease the time of robot control. The average time for robot control when in the network is only one peer is 0.42 s for unsecure and 0.6 for secure primitive and for two peers 1.17 s and 1.27 s respectively. When in the network are three peers the time of robot control using unsecure primitive is 1.82 s and using secure primitive is 1.88 s.

- Experimental results have shown that on the proposed system the usage of security has a cost at the application efficiency by adding some overhead. Analyzing this results, we can conclude that the primitives of JXTA-Overlay sendMsgPeer and secureMsgPeer can be used to control the robot in a smoothly way and can be used in application that tolerate this range of time for robot control. JXTA-Overlay platform can be used not only for efficient and reliable distributed computing but also for collaborative activities and ubiquitous computing by integrating in the platform also end-devices and overcoming thus intrinsic difficulties of current Internet architecture and protocols. JXTA-Overlay is a very good approach for secure robot control.

- We also presented a fuzzy-based system for data replication in P2P system. To decide the RF, we took into consideration three parameters: Number of Documents per Peer (NDP), Replication Percentage (RP) and Scale of Replication per Peer (SRP) and evaluated the proposed system by computer simulations. Data replication improves the system availability by making possible to access the same data from multiply sites. Users can access the nearest replicas reducing the latency and improving the system performance. Our proposed system have a good behaviour can find a right replication factor.

9.2 Future Work

In the future, we would like to consider the following issues.

- Evaluate our implemented system for bigger number of vehicles and more complex scenarios. Until now, we have implemented only vehicle to vehicle communication but as a future work could be the implementation of a complex network that realize car to infrastructure communication. The systems could be a hybrid sensor-vehicular network consisting on the communications between vehicles driving on a highway environment and a network of sensors or roadside infrastructure.

- Modify existing routing protocols to include additional features suitable for vehicular networks.

- Analyze different type of traffics (warnings messages, road messages, video streaming, Internet browsing), different QoS requirements according to the type of traffic, and several protocol interfaces that can be included into the vehicles.

- Implementation of new secure functions and primitives for JXTA-Overlay, which can offer several improvements of the existing JXTA-Overlay protocols and services and increase the reliability of JXTA-based distribution applications and support group management of file sharing.

- Implementation of the platform in the robots where a robot will be considered as a peer. We want to evaluate the implemented system for different scenarios and compare its performance with the existing systems.

- Realization of robot to robot communication combining Web, Sensor and P2P technologies. In autonomous distributed robot systems, many robots cooperate together to carry out many difficult tasks that single robot can not realize. But, in order to cooperate together the robots should communicate with each other. Therefore, the communication among robots is very important problem to be solved. For this reason, we want to implement a P2P system based on JXTA-Overlay platform for robot to robot communication.

- Design and implementation of JXTA-Overlay platform for small wireless terminals. Currently, JXTA-Overlay is implemented only in wired environment and we plan to implement and evaluate its performance in heterogeneous environment.

References

[1] R. Steinmetz and K. Wehrle, “Peer-to-Peer Networking and Computing”, Springer, Informatik - Spectrum, pp. 51-54, 2004.

[2] J. Buford, H. Yu, and E. Lua, “P2P Networking and Applications”, Elsevier, 2008.

[3] S. Oaks, B. Traversat and L. Gong, “JXTA in a Nutshell”, O’Railly, 2001.

[4] P. Charas, “Peer-to-Peer Mobile Network Architecture”, In Proc. of the 1st International Conference on Peer-to-Peer Computing, pp. 55-61, 2001.

[5] V. Martins, E. Pacitti, P. Valduriez, “Survey of Data Replication in P2P Systems”, Technical Report, 2006.

[6] C. Coulon , E. Pacitti , P. Valduriez, “Consistency Management for Partial Replication in a High Performance Database Cluster”, Proc. of the 11th International Conference on Parallel and Distributed Systems (ICPADS’05), pp. 809-815, 2005.

[7] P. A. Bernstein, V. Hadzilacos, N. Goodman, “Concurrency Control and Recovery in Database Systems”, 1987.

[8] B. Kemme, G. Alonso, “A New Approach to Developing and Implementing Eager Database Replication Protocols”, ACM Transactions on Database Systems, Vol. 25, No. 3, pp. 333-379, 2000.

[9] E. Pacitti, P. Minet, and E. Simon, “Fast Algorithms for Maintaining Replica Consistency in Lazy Master Replicated Databases”, Proc. of the 25th International Conference on Very Large Data Bases (VLDB ’99), pp. 126-137, 1999.

[10] Y. Saito and M. Shapiro, “Optimistic Replication”, ACM Comput. Surv. Vol. 37, No. 1, pp. 42-81, 2005.

[11] J. Y. Le Boudec and M. Vojnovic, “The Random Trip Model: Stability, Stationary Regime, and Perfect Simulation”, IEEE/ACM Transactions on Networking, Vol. 14, No. 6, pp. 1153-1166, 2006.

[12] J. Yoon, M. Liu and B. Noble, “A General Framework to Construct Stationary Mobility Models for the Simulation of Mobile Networks”, IEEE Transactions on Mobile Computing, Vol. 5, No. 7, pp. 860-871, July 2006.

[13] J. Kraaier, U. Killat, “The Random Waypoint City Model, User Distribution in a Street-Based Mobility Model for Wireless Network Simulations”, Proc. of WMASH-2005, Cologne, pp. 100-103, Germany, 2005.

[14] K. Nagel and M. Schreckenberg, “A Cellular Automaton Model for Freeway Traffic”, Journal of Physics I France, Vol. 2, pp. 2221-2229, December 1992.

[15] D. Choffnes, F. Bustamante, “STRAW An Integrated Mobility and Traffic Model for VANETs”, Proc. of the 10th International Command and Control Research and Technology Symposium (CCRTS-2005), pp. 69-78, 2005.

[16] F. Bai, N. Sadagopan, and A. Helmy, “IMPORTANT: A Framework to Systematically Analyze the Impact of Mobility on Performance of Routing Protocols for Ad-hoc Networks”, Proc. of IEEE INFOCOM-2003, pp. 825-835, March-April 2003.

[17] N. Potnis and A. Mahajan, “Mobility Models for Vehicular Ad hoc Network Simulations”, Proc. of the 44th annual Southeast Regional Conference, pp. 746-747, New York, 2006.

[18] H. Wu, “MDDV: A Mobility-centric Data Dissemination Algorithm for Vehicular Networks”, Proc. of First ACM International Workshop on Vehicular Ad Hoc Networks (VANET-2004), pp. 47-56, USA, 2004.

[19] J. Zhao and G. Cao, “VADD: Vehicle-assisted Data Delivery in Vehicular Ad hoc Networks”, IEEE Computer Communications, pp. 1-12, 2006.

[20] Y. Bi, H. Zhao, and X. Shen, “A Directional Broadcast Protocol for Emergency Message Exchange in Inter-Vehicle Communications”, Proc. of IEEE ICC-2009, Dresden, Germany, pp. 1-5, 2009.

[21] M. Kihl, M. L. Sichitiu, and H. P. Joshi, “Design and Evaluation of two Geocast Protocols for Vehicular Ad-hoc Networks”, Journal of Internet Engineering, Vol. 2, No. 1, pp. 127-135, 2008.

[22] T. Atchian, L. Brunie, “DG-CASTOR: Direction-based GeoCast Routing Protocol for Query Dissemination in VANET”, IADIS International Telecommunications, Networks and Systems, pp. 105-109, 2008.

[23] G. Korkmaz, E. Ekici, F. Ozguner, U. Ozguner, “Urban Multi-hop Broadcast Protocol for Inter-vehicle Communication Systems”, Proc. of ACM International Workshop on Vehicular Ad hoc Networks, pp. 76-85, 2004.

[24] K. Tokuda, M. Akiyama, and H. Fuji, “Dolphin for Inter-vehicle Communications System”, Proc. of IEEE Intelligent Vehicles Symposium, pp. 504-509, 2000.

[25] M. Sun, W. Feng, T. Lai, K. Yamada, H. Okada, and K. Fujimura, “Gps-based Message Broadcasting for Inter-vehicle Communication”, Proc. of ICPP-2000, pp. 279-286, 2000.

[26] M. Durresi, A. Durresi, L. Barolli, “Emergency Broadcast Protocol for Inte-vehicle Communications”, Proc. of 11th International Conference on Parallel and Distributed Systems (ICDAPS-2005), Vol. 2, pp. 402-406, 2005.

[27] O. Tonguz, N. Wisitpongphan, F. Bai, P. Mudalige, and V. Sadekar, “Broadcasting in VANET”, Proc. of IEEE Mobile Networking for Vehicular Environments, pp. 1-6, 2007.

[28] C. Perkins, E. Belding-Royer and S. Das, “Ad hoc On-Demand Distance Vector (AODV) Routing”, IETF RFC 3561 (Experimental), July 2003.

[29] T. Clausen and P. Jacquet, “Optimized Link State Routing Protocol (OLSR)”, IETF RFC 3626, October 2003.

[30] I. Chakeres and C. Perkins, “Dynamic MANET On-demand (DYMO) Routing”, Internet Draft (draft-ietf-manet-dymo-14), June 2008, Work in progress.

[31] J. Harri, F. Filali and C. Bonnet, “Mobility Models for Vehicular Ad Hoc Networks: A Survey and Taxonomy”, IEEE Communications Surveys & Tutorials, Vol. 11, No. 4, pp. 19-41, Fourth Quarter 2009.

[32] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy, “PATHS: Analysis of Path Duration Statistics and Their Impact on Reactive MANET Routing Protocols”, Proc. of the 4-th ACM International Symposium on Mobile Ad Hoc Networking & Computing, pp. 245-256, 2003.

[33] M. Fiore, J. Harri, F. Filali, and C. Bonnet, “Vehicular Mobility Simulation for VANETs”, Proc. of the 40-th Annual Simulation Symposium (ANSS-2007), pp. 301-309, 2007.

[34] L. Smith, R. Beckan, R. Anson, K. Nagel, and M. Williams, “TRANSIMS: Transportation Analysis and Simulation System”, Proc. of the 5-th National Transportation Planning Methods Applications Conference, LA-UR 95-1664, 1995.

[35] F. Karnadi, Zh. Mo, K. Lan, “Rapid Generation of Realistic Mobility Models for VANET”, Wireless Communications and Networking Conference (WCNC), pp. 2506-2511, 2007.

[36] Information Science Institute (ISI), “Network Simulator Version 2 (NS-2)”, http://www.isi.edu/nsnam.

[37] “The ns-3 Network Simulator”, http://www.nsnam.org.

[38] R. Mangharam, D. Weller, R. Rajkumar, P. Mudalige, F. Bai, “GrooveNet: A Hybrid Simulator for Vehicle-to-Vehicle Networks”, Proc. of Second International Workshop on Vehicle-to-Vehicle Communications (V2VCOM) Invited paper, 2006.

[39] M. Piorkowski, M. Raya, A. L. Lugo, M. Grossglauser, and J. P. Hubaux, “Joint Traffic and Network Simulator for VANETs”, Proc. of Mobile and Information Communication Systems Conference (MICS-2006), October 2006, Available on line at: http://www.mics.ch/.

[40] D. Brookshier, D. Govoni, N. Krishnan, and J.C Soto, “JXTA: Java P2P Programming”, Sams Publishing, 2002.

[41] SUN Microsystem, “Project JXTA v 2.0: Java Programmer’s Guide”, May 2003.

[42] E. Halepovic and R. Deters, “The Costs of Using JXTA”, Proc. of Third International Conference on Peer-to-Peer Computing, pp. 160-166, 2003.

[43] F. Xhafa, R. Fernandez, T. Daradoumis, L. Barolli, S. Caballe, “Improvement of JXTA Protocols for Supporting Reliable Distributed Applications in P2P Systems”, Proc. of NBiS-2007 (Regensburg, Germany), pp. 345-354, September 2007.

[44] P. Caloud, W. Choi, J. C. Latombe, C. L. Pape, and M. Yim, “Indoor Automation with Many Mobile Robots”, Proc. of the IEEE International Workshop on Intelligent Robots and Systems, pp. 67-72, July 1990.

[45] H. Asama, K. Ozaki, A. Matsumoto, Y. Ishida, and I. Endo, “Development of Task Assignment System Using Communication for Multiple Autonomous Robots”, Journal of Robotics and Mechatronics, pp. 122-127, 1992.

[46] A. Kandel, “Fuzzy Expert Systems”, CRC Press, 1992.

[47] H. J. Zimmermann, “Fuzzy Set Theory and Its Applications”, Kluwer Academic Publishers, Second Revised Edition, 1991.

[48] F. M. McNeill, and E. Thro, “Fuzzy Logic. A Practical Approach”, Academic Press, Inc., 1994.

[49] L. A. Zadeh, J. Kacprzyk, “Fuzzy Logic For The Management of Uncertainty”, John Wiley & Sons, Inc., 1992.

[50] T. J. Procyk and E. H. Mamdani, “A Linguistic Self-organizing Process Controller”, Automatica, Vol. 15, No. 1, pp. 15-30, 1979.

[51] G. J. Klir, and T. A. Folger, “Fuzzy Sets, Uncertainty, And Information”, Prentice Hall, Englewood Cliffs, 1988.

[52] T. Munakata, and Y. Jani, “Fuzzy Systems: An Overview”, Commun. of ACM, Vol. 37, No. 3, pp. 69-76, March 1994.

List of Abbreviations

P2P - Peer-to-Peer
VANET - Vehicle Ad-hoc Network
MANET - Mobile Ad-hoc Network
V2V - Vehicle-to-Vehicle
IVC - Inter Vehicle Communication
CAVENET - Cellular Automaton Vehicle Network
RW - Random Waypoint
SRD - Short Range Dependent
LRD - Long Range Dependent
AODV - Ad hoc On-Demand Distance Vector
OLSR - Optimized Link State Routing
DYMO - Dynamic MANET On-demand
DSR - Dynamic Source Routing
DSDV - Destination-Sequenced Distance Vector
NS2 - Network Simulator 2
NS3 - Network Simulator 3
FL - Fuzzy Logic
FC - Fuzzy Control
CBR - Constant Bit Rate
UDP - User Datagram Protocol
TCP - Transport Control Protocol
PDR - Packet Delivery Ratio
SSL - Secure Socket Layer
PKI - Public Key Infrastructure

List of Papers

Journals Papers

1. E. Spaho, L. Barolli, Gj. Mino, F. Xhafa, V. Kolici, R. Miho, "Implementation of CAVENET and Its Usage for Performance Evaluation of AODV, OLSR and DYMO Protocols in Vehicular Networks", Journal of Mobile Information Systems (MIS), IOS Press, Vol. 6, No. 3, pp. 213-227, 2010.

2. E. Spaho, K. Matsuo, L. Barolli, F. Xhafa, J. Arnedo-Moreno, and V. Kolici, "Application of JXTA-Overlay Platform for Secure Robot Control", Journal of Mobile Multimedia (JMM), Rinton Press, Vol. 6, No. 3, pp. 227-242, 2010.

3. E. Spaho, L. Barolli, Gj. Mino, F. Xhafa, "Goodput and PDR Analysis of AODV, OLSR and DYMO Protocols for Vehicular Networks using CAVENET", Journal of Grid, Utility and Computing, Vol. 2, No. 2, pp. 130-138, 2011.

4. E. Spaho, K. Umezaki, L. Barolli, F. Xhafa, M. Younas, "A Fuzzy-based Reliability System for Knowledge Sharing between Robots in P2P JXTA-Overlay Platform", Journal of Cluster Computing, Springer, DOI 10.1007/s10586-012-0230-y, pp. 1-13, 2012.

5. E. Spaho, L. Barolli, F. Xhafa, A. Biberaj, O. Shurdi, "P2P Data Replication and Trustworthiness for a JXTA-Overlay P2P System Using Fuzzy Logic", Journal of Soft Computing, Elsevier, Vol. 13, Issue 1, pp. 321-328, 2013.

International Conference Papers

1. E. Spaho, K. Matsuo, L. Barolli, J. Arnedo-Moreno, F. Xhafa, and V. Kolici, "A Secure JXTA-Overlay Platform for Robot Control", Proc. of 3PGCIC-2010 International Conference, pp.71-76, Fukuoka, Japan, November 2010.

2. E. Spaho, L. Barolli, Gj. Mino, F. Xhafa, "Simulation Evaluation of AODV, OLSR and DYMO Protocols for Vehicular networks using CAVENET", Proc. of CISIS-2011, Conference on Complex, Intelligent and Software Intensive Systems, pp. 152-159, June-July 2011.

3. E. Spaho, L. Barolli, Gj. Mino, F. Xhafa, V. Kolici, "Goodput Evaluation of AODV, OLSR and DYMO Protocols for Vehicular Networks Using CAVENET", Proc. of NBiS-2011 International Conference, Albania, pp. 118-125, September 2011.

4. E. Spaho, L. Barolli, Gj. Mino, F. Xhafa, V. Kolici, "VANET Simulators: A Survey on Mobility and Routing Protocols", Proc. of BWCCA-2011 International Conference, pp. 1-10, October 2011.

5. E. Spaho, L. Barolli, F. Xhafa, J. Iwashige, A. Koyama, "A Fuzzy-Based System for Data Replication in P2P Networks", Proc. of 13th International Symposium on Multimedia Network Systems and Applications (MNSA 2011), IEEE INCoS-2011 International Conference, Fukuoka, Japan, pp. 373-377, November-December 2011.

6. E. Spaho, T. Oda, A. Barolli, F. Xhafa, L. Barolli, M. Takizawa, "A Comparison Study for Different Settings of Crossover and Mutation Rates Using WMN-GA Simulation System", Proc. of First International Workshop on Heterogeneous Networks, Computing and Applications (HNCA-2011), CSA-2011 International Conference, Jeju, South Korea, pp. 643-650, December 2011.

7. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, A. Biberaj, J. Iwashige, "Performance Comparison of DSDV and DYMO Protocols for Vehicular Ad hoc Networks", Proc. of IEEE AINA-2012, Fukuoka, Japan, pp. 629-634, March 2012.

8. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, V. Kolici, M. Takizawa, "Performance Analysis of DSR and DYMO Routing Protocols for VANETs", Proc. of CISIS 2012 Palermo, Italy, pp. 365-369, July 2012.

9. E. Spaho, M. Ikeda, L. Barolli, V. Kolici, F. Xhafa and M. Younas, "Investigation of TCP Traffic in a Vehicular Ad-hoc Network Considering DYMO Routing Protocol", Proc. of Third International Conference on Emerging Intelligent Data and Web Technologies (EIDWT 2012), pp. 111-116, Bucharest, Romania, September 19-21, 2012.

10. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, A. Biberaj and M. Takizawa, "Performance Evaluation of DYMO Protocol in Different VANET Scenarios", Proc. of 15th International Conference on Network-Based Information Systems (NBiS 2012), pp. 97-103, Melbourne, Australia, September 26-28, 2012.

11. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, M. Younas and M. Takizawa, "Performance of OLSR and DSDV Protocols in a VANET Scenario: Evaluation Using CAVENET and NS3", Proc. of Seventh International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA-2012), pp. 108-113, Victoria, Canada, November 12-14, 2012.

12. E. Spaho, E. Kulla, F. Xhafa, L. Barolli, "P2P Solutions to Efficient Mobile Peer Collaboration in MANETs", Proc. of Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 379-383, Victoria, Canada, 2012.

13. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, M. Younas, M. Takizawa, "Performance Evaluation of OLSR and AODV Protocols in a VANET Crossroad Scenario", Proc. of IEEE AINA-2013, pp. 577-582, Barcelona, Spain, March 2013.

14. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, "Performance Evaluation of OLSR Protocol in a Grid Manhattan VANET Scenario for Different Applications", Accepted, To appear on Proc. of CISIS-2013, Taichung, Taiwan, 2013.

15. E. Spaho, M. Ikeda, L. Barolli, F. Xhafa, "Performance Comparison of OLSR and AODV Protocols in a VANET Crossroad Scenario", Accepted, To appear on Proc. of ITCS-2013, Fukuoka, Japan, 2013.

16. E. Spaho, K. Matsuo, L. Barolli, F. Xhafa, "Robot Control Architectures: A Survey", Accepted, To appear on Proc. of IRoA-2013, Fukuoka, Japan, 2013.