SANDIA REPORT

SAND2005-1661
Unlimited Release
Printed March 2005

Advanced Mobile Networking, Sensing, and Controls

J. T. Feddema, R. H. Byrne, J. J. Harrington, D. M. Kilman, C. L. Lewis, R. D. Robinett, B. P. Van Leeuwen, and J. G. Young

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-1661
Unlimited Release
Printed March 2005

Advanced Mobile Networking, Sensing, and Controls

J. T. Feddema, R. H. Byrne, C. L. Lewis, J. G. Young
Intelligent Systems, Sensors, and Controls Department

J. J. Harrington
Mobile Robotics Department

D. M. Kilman, B. P. Van Leeuwen
Networked Systems, Surveillance and Assurance Department

R. D. Robinett
Energy, Infrastructure and Knowledge Systems Center

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1003

Abstract

This report describes an integrated approach for designing communication, sensing, and control systems for mobile distributed systems. Graph theoretic methods are used to analyze the input/output reachability and structural controllability and observability of a decentralized system. Embedded in each network node, this analysis will automatically reconfigure an ad hoc communication network for the sensing and control task at hand. The graph analysis can also be used to create the optimal communication flow control based upon the spatial distribution of the network nodes. Edge coloring algorithms tell us that the minimum number of time slots in a planar network is equal to either the maximum number of adjacent nodes (or degree) of the undirected graph plus some small number. Therefore, the more spread out that the nodes are, the fewer number of time slots are needed for communication, and the smaller the latency between nodes. In a coupled system, this results in a more responsive sensor network and control system. Network protocols are developed to propagate this information, and distributed algorithms are developed to automatically adjust the number of time slots available for communication. These protocols and algorithms must be extremely efficient and only updated as network nodes move. In addition, queuing theory is used to analyze the delay characteristics of Carrier Sense Multiple Access (CSMA) networks. This report documents the analysis, simulation, and implementation of these algorithms performed under this Laboratory Directed Research and Development (LDRD) effort.

This page intentionally blank.

Contents

List of Figures

This page intentionally blank.

1 Introduction

In the wake of current world activities and terrorist events, the United States Army is being transformed into a more agile, deployable, mobile force called the Objective Force. One of the key tenets of this Objective Force is that soldiers will be provided with a situational awareness that will greatly enhance their warfighting capabilities. They will have access to information regarding their mission, aerial and terrain maps, weather, their current location, the location of barriers such as minefields, and the location and strength of friend and foe military forces both on the ground and in the air. They will have access to information from man-delivered and airdropped unattended ground sensors. They will be able to communicate with other soldiers in the field and the higher echelon, and they will be able to command and control unmanned vehicles such as UGVs (Unmanned Ground Vehicles) and UAVs (Unmanned Air Vehicles). Key to all these capabilites is a secure, reliable, highly responsive communication network that scales well with large numbers of nodes.

Most current large-scale distributed networks, such as the Internet, use a Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol for communication. Using a CSMA protocol, when node A wants to talk to node B, it first listens for an open channel. If the channel is open, it transmits its message. If not, it waits a random back-off time before trying again. With the Collision Avoidance option, a short request-to-send/clear-to-send message is exchanged between two nodes to first clear the channel for a longer message. This is used to eliminate the hidden-node problem where two nodes, which can not hear each other, both send long messages to an intermediate node at the same time. The primary advantage of the CSMA/CA protocol is that it does not require a prior knowledge of what other nodes are present in the network. Unfortunately, there are two disadvantages to this protocol. First, the bandwidth of a properly designed system is approximately half of an optimally slotted Time Division Multiple Access (TDMA) protocol and can be substantially less if the number of nodes grows too large. Second, message collisions do occur making the system undeterministic. Being undeterministic makes the network difficult to analyze in terms of bandwidth and latency, and also makes it impossible to provide reliable closed loop controls over a CSMA/CA network. For these reasons, in this project we have investigated decentralized algorithms that would provide optimal flow control in a large TDMA network. Within this report, comparisons are made between the TDMA and CSMA networks. We used graph theory and large scale decentralized control theory as the theoretical underpinnings of this work.

A particular focus of this effort is the distributed control of multiple unmanned robotic vehicles. Over the past decade, considerable research has concentrated on the development of cooperative controls of multiple robotic systems [1]. Much of this work is originally inspired by the observation that social insects such as termites, ants, and bees perform amazing group behaviors when individually executing simple rules [2]. The hypothesis being that it must be possible to create useful "emergent" group behaviors from simple individual behaviors. While this hypothesis has yet to be proven or disproved, many people have demonstrated some interesting group

behaviors. For example, researchers have shown that groups of robots can forage for objects [3], maintain formations [4], and push boxes [5].

More recently, researchers have begun to take a system controls perspective and analyze the stability of multiple vehicles when driving in formations. Chen and Luh [6] examined decentralized control laws that drove a set of holonomic mobile robots into a circular formation. A conservative stability requirement for the sample period is given in terms of the damping ratio and the undamped natural frequency of the system. Similarly, Yamaguchi studied line-formations [7] and general formations [8] of nonholonomic vehicles, as did Yoshida et al. [9]. Decentralized control laws using a potential field approach to guide vehicles away from obstacles can be found in [10]-[11]. In these studies, only continuous time analyses have been performed, assuming that the relative position between vehicles and obstacles can be measured at all time.

Another way of analyzing stability is to investigate the convergence of a distributed algorithm. Beni and Liang [12] prove the convergence of a linear swarm of asynchronous distributed autonomous agents into a synchronously achievable configuration. The linear swarm is modeled as a set of linear equations that are solved iteratively. Their formulation is best applied to resource allocation problems that can be described by linear equations. Liu et al. [13] provide conditions for convergence of an asynchronous swarm in which swarm "cohesiveness" is the stability property under study. Their paper assumes position information is passed between nearest neighbors only and proximity sensors prevent collisions.

Also of importance is the recent research combining graph theory with decentralized controls. Most cooperative mobile robot vehicles have wireless communication, and simulations have shown that a wireless network of mobile robots can be modeled as an undirected graph [14]. These same graphs can be used to control a formation. Desai et al. [15]-[16] used directed graph theory to control a team of robots navigating terrain with obstacles while maintaining a desired formation and changing formations when needed. When changing formations, the transition matrix between the current adjacency matrix and all possible control graphs are evaluated.

Over the past 8 years, Sandia's Intelligent Systems and Robotics Center has been developing cooperative control systems for military missions such as perimeter surveillance [17], facility reconnaissance [18], and chemical plume tracking [19]. Multiple mobile robots perform these missions in a coordinated fashion using low bandwidth communication between nodes. We have demonstrated that by using Provably Convergent Cooperative Controls (PC^3) we can guarantee a specified level of performance even under uncertainty. PC^3 involves selecting an overall performance index and using distributed optimization techniques to minimize this performance index. The stability of the control can be proven using a vector Liapunov function. The scheme is inherently fault tolerant to nodal failure, allowing other nodes to complete a task started by a failed node.

Key to the success of PC^3 is that the communication between nodes is used to synchronize the controls. To date, we have used a TDMA protocol with a small number of nodes (up to 10). Eventually, we would like to extend the analysis and demonstrate this PC^3 capability in larger scaled systems of 100 to 1000 nodes, but

first, we must develop the underlying communication layer and ensure that it is a deterministic system. Therefore, an important aspect of this work is an integrated approach to both communication and controls. In this project, we have used graph theoretic methods to analyze the input/output reachability and structural controllability and observability of a decentralized system. This analysis can be embedded in each node and be used to automatically reconfigure an ad hoc communication network for the control task at hand. The graph analysis can also be used to create the most efficient communication flow control based upon spatial distribution of the network nodes. Edge coloring algorithms in graph theory tell us that the minimum number of time slots in a planar network is equal to the maximum number of adjacent nodes plus some small number. The more spread out the nodes are, the fewer number of time slots are needed for communication, and the smaller the latency between nodes. In a coupled system, smaller latency results in a more responsive control system. In this project, network protocols that propagate this information and distributed algorithms that automatically adjust the number of time slots available for communication were evaluated. These protocols and algorithms must be extremely efficient and only updated as network nodes move.

The next section provides a common mathematical framework that can be used to develop cooperative behaviors among mobile robot vehicles. Section 3 describes the effects of communication on the sampling period of a cooperative system. Four different communication protocols: a Time Division Multiple Access (TDMA) linear broadcast, a TDMA polylogarithmic broadcast, a TDMA coloring algorithm, and a Collision Sense Multiple Access (CSMA) algorithm are compared. These communication effects are verified with OPNET simulations in Section 4. Section 5 concentrates on the delay characteristics of a CSMA network, comparing Monte Carlo simulations to closed form predicted performance using queuing theory. Finally, section 6 investigates solving the cooperative control stability problem using a Linear Matrix Inequality formulation.

2 Cooperative Control

In this section, a common mathematical framework that can be used to describe a number of cooperative behaviors is developed. This mathematical framework is applied to three generic behaviors: containment, coverage, and converging. The authors believe that this same framework could be applied to the other behaviors, although this has not been proven at the time of this publication.

This mathematic framework is motivated by the fact that most of the laws in physics and mechanics can be derived by finding the maximum or minimum of some performance index, in this case, called the Lagrangian integral. In Table 1, notice that the Lagrangian of each phenomenon is a function of some gradient term squared. In the analysis that follows, you will notice that each behavior's performance index is also a function of some gradient term squared.

Following this same optimization approach, a three-step process for developing

Phenomenon	**Lagrangian**
Classical Mechanics	$\frac{1}{2}m\left(\frac{\partial q}{\partial t}\right)^2 - V$
Flexible String or Compressible Fluid	$\frac{1}{2}\rho\left[\left(\frac{\partial q}{\partial t}\right)^2 - c^2\nabla q \bullet \nabla q\right]$
Diffusion Equation	$-\nabla\psi \bullet \psi^* - \ldots$
Schrodinger Equation	$-\frac{h}{2m}\nabla\psi \bullet \nabla\psi^* - \ldots$
Electromagnetic Equations	$4\sum_{n=1}^{4}\nabla q_n \bullet \nabla q_n - \ldots$
Boltzmann Law	$4\left(\frac{\partial q}{\partial E}\right)^2 - \ldots$

Table 1: Lagrangian for various physical phenomena [20]

cooperative control algorithms has been developed [21]. These three steps are as follows:

Step 1. Define a global performance index as a function of parameters from all entities.

Step 2. Partition and eliminate terms in the performance index so that only terms of local neighbors are included.

Step 3. The local control law is the gradient (or the product of the inverse Hessian and gradient) of the partitioned performance index.

The first step requires that one understands the problem well enough that it can be posed as a global optimization problem. This step can be relatively difficult, but as the examples in the remainder of this section will show, that with the right simplifying assumptions, rather simple equations can be used to solve difficult problems.

The second step, partitioning the performance index, is often used in parallel optimization to reduce the computation time for large-scale problems [22]. In this case, the second step is used to reduce communications between robots and to increase robustness of the distributed system. The control law that would result from step 1 would require that every robot be able to communicate with all the other robots. As the number of robots increase to 100s and 1000s, the time delay necessary for communication would make the resulting control infeasible. Instead, partitioning the

performance index and eliminating terms to include only terms of local neighbors results in a control law that only requires communication with nearest neighbors, thus greatly reducing communication delay. Also, using nearest neighbors that change throughout the motion adds an element of robustness. The mathematical formulation of the partition does not specify that robot number 10 must communicate with robot number 6. Instead, the mathematical formulation specifies a group of nearest neighbors that can change based on external forces and environmental conditions. This creates an element of self-organization that allows the system to change and evolve. If a robot fails, a new set of nearest neighbors is formed.

The third step is to solve for the extremum of the partitioned performance index using either a first-order steepest descent algorithm or second order method such as the Newton's Method [23].

The remainder of this section will be spent showing how these three steps have been used in practice. Six examples are given with details on the problem formulation and the task that was performed.

2.1 Example 1: Spreading Apart along a Line - A Containment Behavior

This first example is a simple one-dimensional problem. The goal is for multiple robots to evenly spread apart along a straight line using only information from the neighboring robots on the right and left. In Figure 1, the first and last robots are assumed to be stationary while the ones in between are to spread apart a distance d away from each other.

The optimization steps are as follows.

Step 1. Specified as an optimization problem, the objective is to

$$\min_{\bar{x}} \; v(\bar{x}) \tag{1}$$

where the global performance index is

$$v(\bar{x}) = \frac{1}{2} \sum_{i=1}^{n-1} \left(d - |x_{i+1} - x_i|\right)^2 \tag{2}$$

x_i is the position of robot i, $\bar{x} = [x_1 \dots x_n]^T$ are the positions of all the robots, and d is the desired distance between each robot. The goal is to minimize the sum of squared errors in distances between every robot.

Step 2. This problem is easily partitioned amongst the interior $n-2$ robots. The distributed objective is to

$$\min_{x_i} \; v_i(\bar{x}) \quad \forall \quad i = 2, \dots, n-1 \tag{3}$$

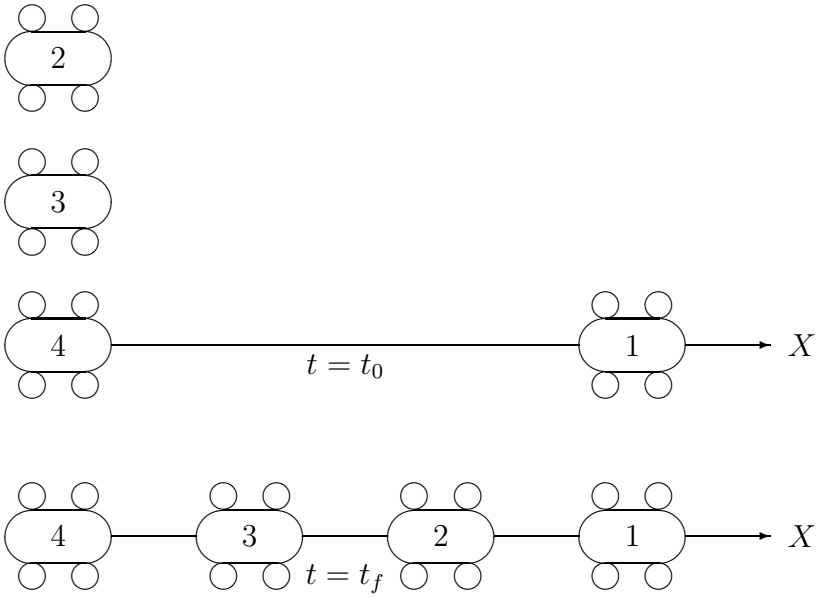

Figure 1: One-dimensional control problem. The top line is the initial state. The second line is the desired final state. Vehicles 1 and 4 are boundary conditions. Vehicles 2 and 3 spread out along the line using only the position of their left and right neighbors.

where the partitioned performance index is

$$v_i(\bar{x}) = \frac{1}{2}\left(d - |x_i - x_{i-1}|\right)^2 + \frac{1}{2}\left(d - |x_i - x_{i+1}|\right)^2 \tag{4}$$

Because of the additive form of Equation (2), simultaneously solving Equation (3) for each robot is the same as minimizing the global performance index in Equation (2). Therefore, in this case, no terms were eliminated. This is not necessarily true for the other example problems below.

Step 3. A steepest descent control law for the partitioned performance index is given by

$$x_i(k+1) = x_i(k) - \alpha \nabla v_i(\bar{x}(k)), \quad 0 < \alpha \leq 1, \tag{5}$$

where

$$\nabla v_i(\bar{x}) = 2x_i - (x_{i+1} + x_{i-1}) \quad if \quad x_{i-1} < x_i < x_{i+1} \tag{6}$$

Note that $\nabla v_i(\bar{x}) = 0$ when $x_i = \frac{1}{2}(x_{i+1} + x_{i-1})$. Therefore, the vehicles will disperse along the line until they have reached a position that is exactly in the middle of its nearest neighbors. In [18], it is shown that α is actually more constrained than indicated in Equation (5) depending on the speed of the vehicle and the communication sample period. The control law in Equations (5)-(6) have been used to spread robot vehicles apart along a perimeter [17] as shown in Figures 2 - 3, as well as to spread out hopping minefield robots [24] as shown in Figures 4 - 5.

Figure 2: Four robot vehicles are shown guarding a perimeter denoted by the blue line segments. When an intrusion detection sensor denoted by the numbered circles alarms, one robot vehicle attends to the alarm (vehicle near sensor 33) while the others spread apart along the perimeter so that each vehicle is midway between its neighbors.

Figure 3: Robot vehicles used to perform perimeter surveillance task

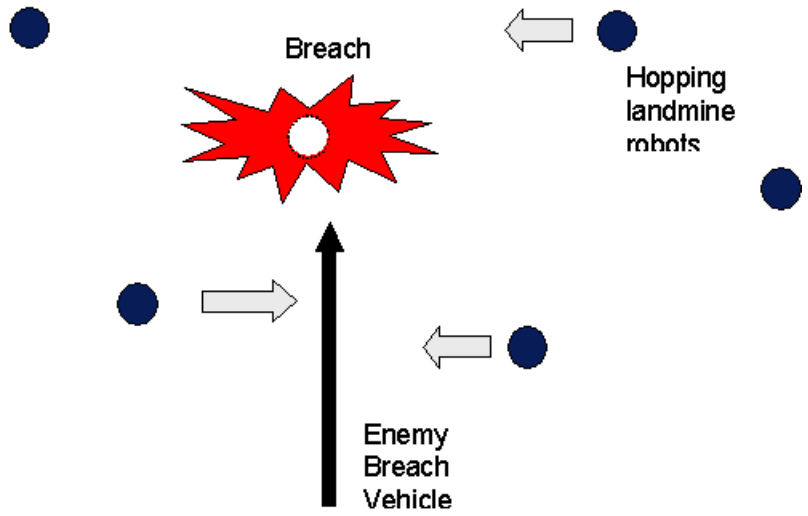

Figure 4: Hopping landmine robots are filling breach left by enemy vehicle. When a robot is breached, the robots will hop towards the missing robot and settle when each robot is midway between its neighbors.

Figure 5: Hopping landmine robots used in self-healing minefield tests.

2.2 Example 2: Coverage of a Two-Dimensional Space

Next, we consider the example of dispersing robots in a plane in a specified pattern. In Figure 6, the robots are to move from the configuration on the left to the configuration on the right. The configuration on the right is specified by the distances d_{ij} between robot i and robot j.

Step 1. The objective is to

$$\min_{\bar{x}} \; v(\bar{x}) \tag{7}$$

where the global performance index is

$$v(\bar{x}) = \frac{1}{2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(d_{ij}^2 - (x_i - x_j)^2 - (y_i - y_j)^2\right)^2 \tag{8}$$

$\bar{x}_i = [x_i\, y_i]^T \in \mathbb{R}^2$ is the position of robot i in the xy plane, and $\bar{x} = [\bar{x}_1^T \ldots \bar{x}_n^T]^T$ is the position of all the robots in the xy plane. By minimizing the error between the squared desired distance and the squared measured distance between every pair-wise combination of robots, we can drive the robots from an initial pattern to the desired specified pattern. Notice that the global performance index does not specify the orientation or final absolute position of the group of robots.

Step 2. The global performance index is over constrained since it is possible to achieve the same minimum solution without having to minimize the error between every pair-wise combination. The same minimum solution can be achieved by only minimizing the error between neighboring robots. The distributed objective is to

$$\min_{\bar{x}_i} \; v_i(\bar{x}) \quad \forall \quad i = 1, \ldots, n \tag{9}$$

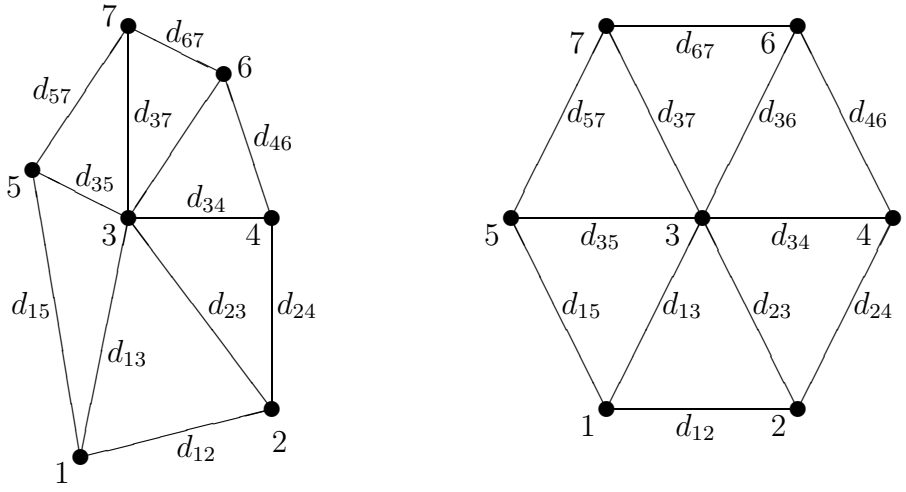

Figure 6: (Left) Initial configuration of robots. (Right) Desired final configuration.

where the partitioned performance index is

$$v_i(\bar{x}) = \frac{1}{2} \sum_{j \in NN} \left(d_{ij}^2 - (x_i - x_j)^2 - (y_i - y_j)^2 \right)^2 \tag{10}$$

and NN stands for nearest neighbor.

Step 3. The steepest descent control law for the partitioned performance index is given by

$$\bar{x}_i(k+1) = \bar{x}_i(k) - \alpha \nabla v_i(\bar{x}(k)), \quad 0 < \alpha \leq 1 \tag{11}$$

where

$$\nabla v_i = \begin{bmatrix} \dfrac{\partial v_i}{\partial x_i} \\ \\ \dfrac{\partial v_i}{\partial y_i} \end{bmatrix} \in \mathbb{R}^2, \tag{12}$$

$$\frac{\partial v_i(\bar{x})}{\partial x_i} = -2 \sum_{j \in NN} \left[d_{ij}^2 - (x_i - x_j)^2 - (y_i - y_j)^2 \right] (x_i - x_j), \tag{13}$$

$$\frac{\partial v_i(\bar{x})}{\partial y_i} = -2 \sum_{j \in NN} \left[d_{ij}^2 - (x_i - x_j)^2 - (y_i - y_j)^2 \right] (y_i - y_j). \tag{14}$$

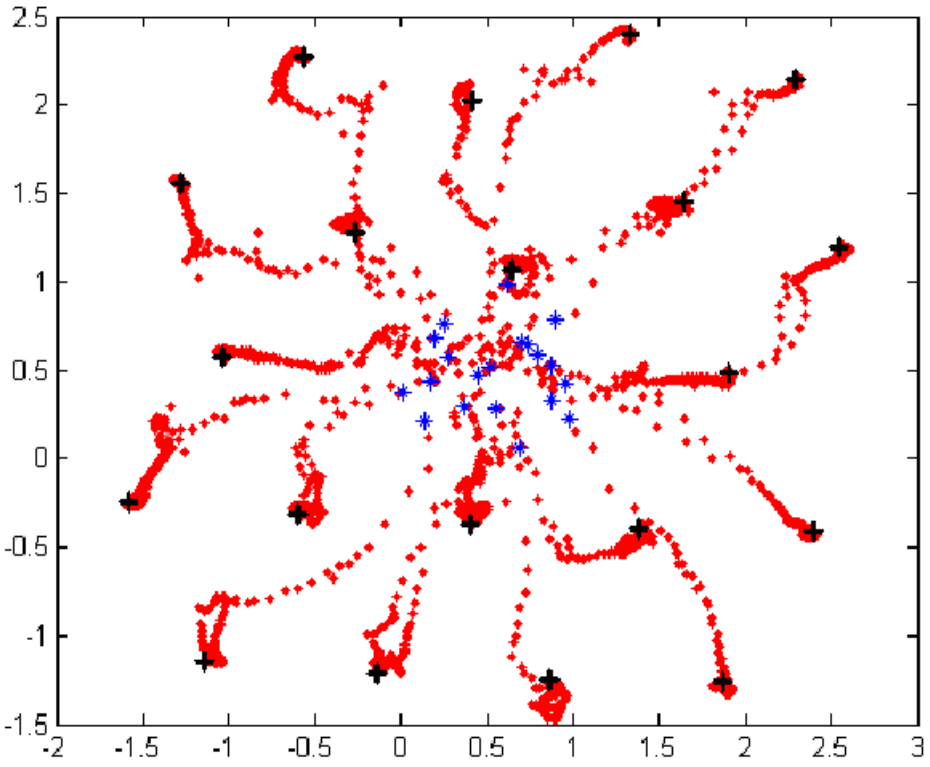

Figure 7: Plot of 20 vehicles' trajectories started from a clustered position with the goal of spreading out uniformly through the space (blue * indicates initial position, red marks indicate trajectory, and black + indicate final position).

Note that $\nabla v_i = 0$ when $d_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 \ for \ j \in NN$. In [25], the connective stability of this control law is proven using a vector Liapunov technique. The control law in Equations (11)- (14) has been used to spread apart the hopping minefield robots as shown in Figure 7. In this case, the specified distances are all equal and the number of nearest neighbors used for control is three.

2.3 Example 3: Coverage of a Two-Dimensional Space with Constraints

Next, consider the same problem as in the previous example, except that the robots are constrained to stay within a region that is bounded by line segments as shown in Figure 8.

Step 1. The objective is to

$$\min_{\bar{x}} \ v(\bar{x}) \tag{15}$$

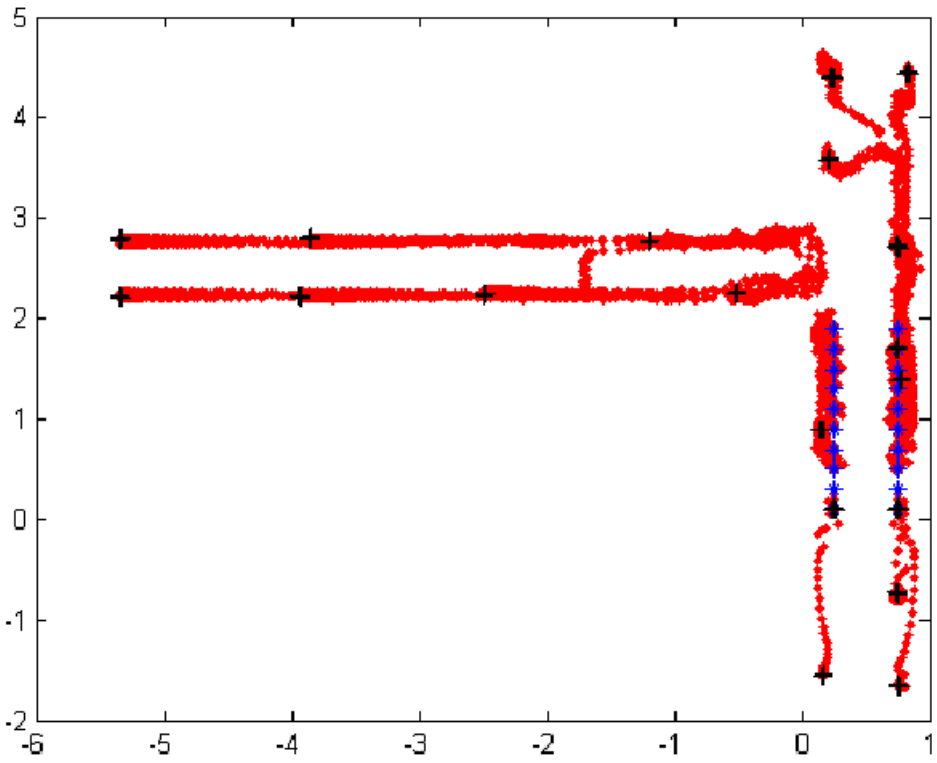

Figure 8: Plot of 20 vehicles' trajectories started from a clustered position with the goal of spreading apart uniformly through a hallway with a side corridor (blue * indicates initial position, red marks indicate trajectory, and black + indicate final position).

where the global performance index is

$$v(\bar{x}) = \frac{1}{2}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(d_{ij}^2 - |\bar{x}_i - \bar{x}_j|^2\right)^2 \tag{16}$$

subject to

$$A\bar{x}_i \leq b \quad \forall \quad i = 1, \ldots, n \tag{17}$$

where $A \in \mathbb{R}^{\mathrm{mx2}}$ and $b \in \mathbb{R}^{\mathrm{m}}$. Equation (17) specifies the boundary conditions of m straight-line segments.

Step 2. The distributed objective is to

$$\min_{\bar{x}_i} \; v_i(\bar{x}) \quad \forall \quad i \tag{18}$$

where the partitioned performance index is

$$v_i(\bar{x}) = \frac{1}{2}\sum_{j\in NN}\left(d^2 - |\bar{x}_i - \bar{x}_j|^2\right)^2 + \frac{1}{2}\Lambda\sum_{l\in NO}\left(A_l\bar{x}_i - b_l\right)^{-2}. \tag{19}$$

Here, the inequality constraints in Equation (17) have been added as a weighted penalty function that is the sum of the inverse squared perpendicular distances between robot i and the nearest obstacle (NO) line segments l. The Δ is a scalar used to vary the importance of obstacle avoidance. As before, NN stands for the set of nearest neighbors. Similarly, the set NO is the set of nearest obstacles.

Step 3. The steepest descent control law is

$$\bar{x}_i(k+1) = \bar{x}_i(k) - \alpha\nabla v_i(\bar{x}(k)), \quad 0 < \alpha \leq 1 \tag{20}$$

where

$$\nabla v_i(\bar{x}) = -2\sum_{i\in NN}\left(d^2 - |\bar{x}_i - \bar{x}_j|^2\right)(\bar{x}_i - \bar{x}_j) - \Lambda\sum_{l\in NO}A_l^T\left(A_l\bar{x}_i - b_l\right)^{-3} \tag{21}$$

The control law in Equations (20) and (21) has been used to spread out the robot vehicles in a hallway as shown in Figure 8. The nearest obstacles are determined from IR proximity sensors. The specified distance d between vehicles was chosen to be within the 10 meter acoustic range of the sensors on top of the vehicle (see Figure 9) [26]. Again, the number of nearest neighbors used for control is three.

2.4 Example 4. Forming an Ellipse with Constraints - A Containment Behavior

Next, consider a path following/formation problem where multiple vehicles are to 1) travel towards and spread apart on an ellipse, 2) not drive into each other, and 3) stay away from obstacle line segments. This is shown in Figure 10.

Figure 9: Robot vehicles used in an indoor communication/navigation network.

Figure 10: Robot path planner drives vehicles towards ellipse while staying away from obstacles, denoted by red line, and the other vehicles.

Step 1. The objective is to

$$\min_{\bar{x}} \; v(\bar{x}) \tag{22}$$

where the global performance index is

$$v(\bar{x}) = \frac{1}{2}\sum_{i=1}^{n}\left((\bar{x}_i - \bar{x}_0)^T \begin{bmatrix} \frac{1}{\rho^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix} (\bar{x}_i - \bar{x}_0) - 1\right)^2 \tag{23}$$

subject to

$$|\bar{x}_{i+1} - \bar{x}_i| > d \quad \forall \quad i = 1, \ldots, n-1 \tag{24}$$

$$A\bar{x}_i \leq b \quad \forall \quad i = 1, \ldots, n. \tag{25}$$

The global performance index is squared error of the robot's position from the ellipse. The position of the center of the ellipse is $\bar{x}_0$, and ρ and σ are the elliptical parameters along the $x-$ and $y-$axes. The first constraint ensures that the vehicles stay a distance d apart from each other. The second constraint ensures that the vehicles stay away from the line constraints as in the previous example.

Step 2. The distributed objective is

$$\min_{\bar{x}_i} \; v_i(\bar{x}) \quad \forall \quad i \tag{26}$$

where the partitioned performance index is

$$v_i(\bar{x}) = \frac{1}{2}\left((\bar{x}_i - \bar{x}_0)^T \begin{bmatrix} \frac{1}{\rho^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix} (\bar{x}_i - \bar{x}_0) - 1\right)^2 +$$

$$\frac{1}{2}\sum_{j \in NN}\left(d^2 - |\bar{x}_i - \bar{x}_j|^2\right)^2 + \frac{1}{2}\Lambda \sum_{l \in NO}(A_l\bar{x}_i - b_l)^{-2} \tag{27}$$

The two constraints are implemented as penalty functions. The equations are the same as in the previous example.

Step 3. The steepest descent control law is

$$\bar{x}_i(k+1) = \bar{x}_i(k) - \alpha \nabla v_i(\bar{x}(k)) \tag{28}$$

where

$$\nabla v_i(\bar{x}) = 2\left((\bar{x}_i - \bar{x}_0)^T \begin{bmatrix} \frac{1}{\rho^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix} (\bar{x}_i - \bar{x}_0) - 1\right) \begin{bmatrix} \frac{1}{\rho^2} & 0 \\ 0 & \frac{1}{\sigma^2} \end{bmatrix} (\bar{x}_i - \bar{x}_0) -$$

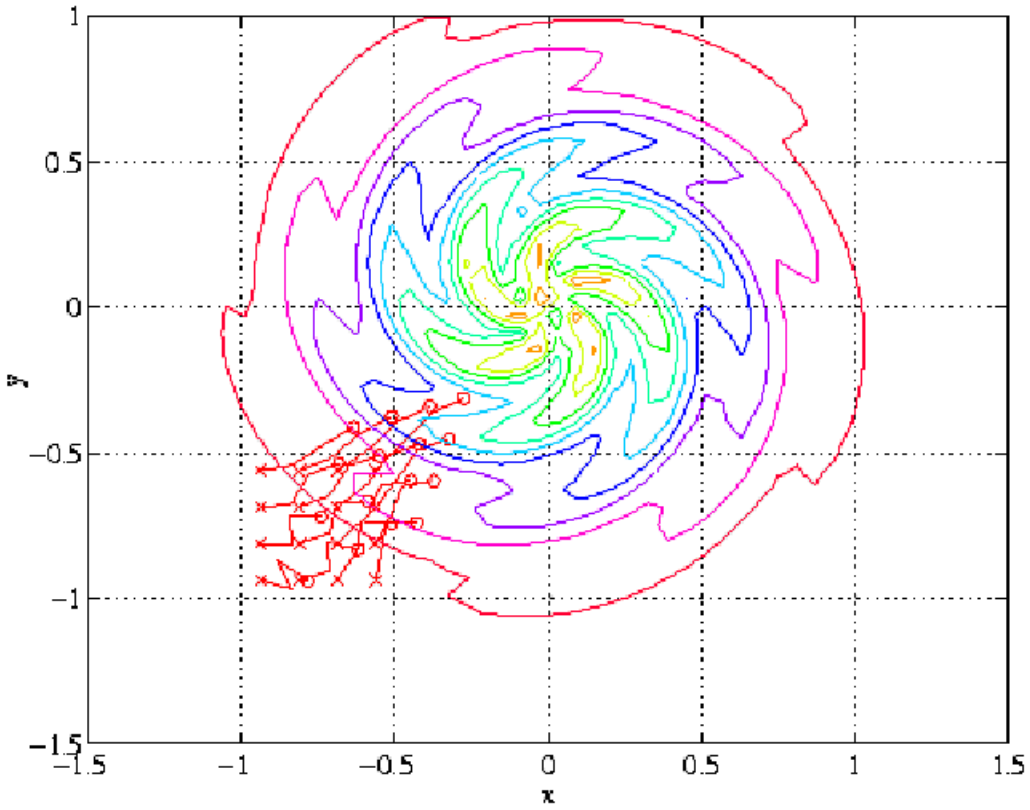

Figure 11: Multiple vehicles converging on a rotating plume.

$$2 \sum_{j \in NN} \left(d^2 - |\bar{x}_i - \bar{x}_j|^2 \right) (\bar{x}_i - \bar{x}_j) - \Lambda \sum_{l \in NO} A_l^T \left(A_l \bar{x}_i - b_l \right)^{-3} \tag{29}$$

The control law in Equations (28)-(29) has been implemented on a path planner as shown in Figure 10. The number of nearest neighbors and number of nearest obstacles can be one if the time step is small. The nearest neighbor and obstacle will continually change throughout the motion.

2.5 Example 5. Converging on the Source of a Plume - 2D Case

The next example is a plume localization problem. The objective is for multiple vehicles to locate and converge on a source, which could either be acoustic, radio frequency, temperature, or chemical (See Figure 11). It is assumed that the spatial signature of the source can be approximated by a quadric surface. The form of this second order equation allows us to easily formulate convergent control to the extremum of the surface. If the data were fit to a higher order surface with many local extremum, then it would not be possible to guarantee convergence to a single solution.

Step 1. The objective is

$$\max_{\bar{x}} \quad v(\bar{x}) \tag{30}$$

where the global performance index is

$$v(\bar{x}) \cong \sum_{i=1}^{N} a_0 + A_1^T(\bar{x}_i - \bar{x}_0) + \frac{1}{2}(\bar{x}_i - \bar{x}_0)^T A_2(\bar{x}_i - \bar{x}_0) \tag{31}$$

The parameters of the quadratic surface are $a_0 \in \mathbb{R}$, $A_1 \in \mathbb{R}^2$, and $A_2 \in \mathbb{R}^{2x2}$. The center of the source is located at $\bar{x}_0 \in \mathbb{R}^2$.

Step 2. The distributed objective is

$$\max_{\bar{x}_i} v_i(\bar{x}) \quad \forall \quad i \tag{32}$$

where the partitioned performance index is

$$v_i(\bar{x}) \cong \sum_{i=1}^{N} a_{0i} + A_1^T(\bar{x}_j - \bar{x}_i) + \frac{1}{2}(\bar{x}_j - \bar{x}_i)^T A_{2i}(\bar{x}_j - \bar{x}_i) \tag{33}$$

Each vehicle determines it's own estimate of the quadratic surface using information from its nearest neighbors. An alternative approach is to use data from as many neighbors as possible and calculate a least-squares estimate of the quadratic coefficients. References [19]-[27] describe the least squares fitting algorithm in more detail.

Step 3. The second order Newton's method control law is

$$\bar{x}_i(k+1) = \bar{x}_i(k) - \alpha A_{2i}^{-1}|_{\bar{x}(k)} \quad A_{1i}|_{\bar{x}(k)} \tag{34}$$

where the quadratic coefficients are determined from the solution to the nearest neighbor equations

$$v(\bar{x}_j) \cong a_{0i} + A_{1i}^T(\bar{x}_j - \bar{x}_i) + \frac{1}{2}(\bar{x}_j - \bar{x}_i)^T A_{2i}^T(\bar{x}_j - \bar{x}_i) \quad \forall \quad j \in NN \tag{35}$$

The control law in Equations (34)-(35) has been implemented on RATLER vehicles (See Figure 3) that locate an acoustic source and on a set of miniature robotic vehicles (See Figure 12) that locate a block of dry ice [19]-[27]. In both cases, the number of nearest neighbors is six because seven measurements (including itself) are needed to uniquely determine the quadric coefficients A_{1i} and A_{2i} .

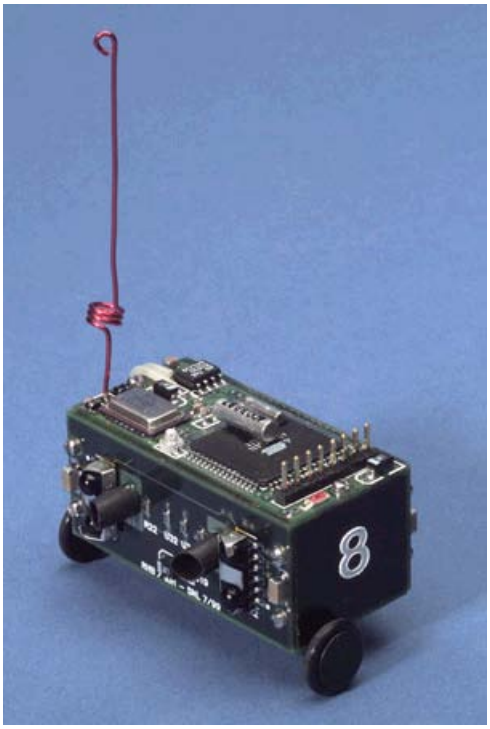

Figure 12: Miniature robot used in the plume localization experiment that located a block of dry ice.

2.6 Example 6. Converging on the Source of a Plume - 3D Case

The last example is a three-dimensional plume localization problem. The objective is for multiple vehicles to locate and converge in on a source, which could either be acoustic, temperature, or chemical. It is assumed that the spatial signature of the source can be approximated by a quadratic surface.

Step 1. The objective is

$$\max_{\bar{x}} \quad v(\bar{x}) \tag{36}$$

where the global performance index is

$$v(\bar{x}) \cong \sum_{i=1}^{N} a_0 + A_1^T(\bar{x}_i - \bar{x}_0) + \frac{1}{2}(\bar{x}_i - \bar{x}_0)^T A_2(\bar{x}_i - \bar{x}_0). \tag{37}$$

The parameters of the quadratic surface are $a_0 \in \mathbb{R}$, $A_1 \in \mathbb{R}^3$, and $A_2 \in \mathbb{R}^{3x3}$. The center of the source is located at $\bar{x}_0 \in \mathbb{R}^3$.

Step 2. The distributed objective is

$$\max_{\bar{x}_i} v_i(\bar{x}) \quad \forall \quad i \tag{38}$$

where the partitioned performance index is

$$v_i(\bar{x}) \cong \sum_{j \in NN} a_{0i} + A_{1i}^T(\bar{x}_j - \bar{x}_i) + \frac{1}{2}(\bar{x}_j - \bar{x}_i)^T A_{2i}(\bar{x}_j - \bar{x}_i) \tag{39}$$

Each vehicle determines its own estimate of the quadratic surface using information from its nearest neighbors.

Step 3. The second order Newton's method control law is

$$\bar{x}_i(k+1) = \bar{x}_i(k) - \alpha A_{2i}^{-1}|_{\bar{x}(k)} \quad A_{1i}|_{\bar{x}(k)} \tag{40}$$

where the quadratic coefficients are determined from the solution to the nearest neighbor equations

$$v(\bar{x}_j) = a_{0i} + A_{1i}^T(\bar{x}_j - \bar{x}_i) + \frac{1}{2}(\bar{x}_j - \bar{x}_i)^T A_{2i}^T(\bar{x}_j - \bar{x}_i) \quad \forall \quad j \in NN \tag{41}$$

For the 3D case, the number of nearest neighbors is nine because ten measurements (including itself) are needed to uniquely determine the quadratic coefficients A_{1i} and A_{2i}. Most recently, this algorithm has been implemented on underwater vehicles that locate and converge in on a 3D plume [28]. Preliminary tests were conducted with a synthetic plume. Synthesized sensor data was calculated as a function of position to debug the algorithm. The underwater robots are shown in Figure 13. The results of a typical test run are shown in Figure 14.

These six examples demonstrate the utility of this three-step process for creating locally optimal distributed controls for multiple robotic vehicles. The resulting control laws are robust and only require sharing of information between nearest neighbors. The robustness is the result of the self-organizing nature of the control where nearest neighbors are continually changing throughout the motions. If a vehicle is lost or dies, another set of nearest neighbors can be used to complete the task. By using penalty functions to approximate constraints, the control laws are in a form that is identical to the potential field control laws often used for controlling single and multiple robots. The main difference is the switching of potential fields based on the nearest neighbors and the nearest obstacles.

3 Communication Effects

In this section, previous analysis regarding stable control of multiple vehicles using large-scale decentralized control techniques [18] is extended to include the communications aspects of the problem. A stability analysis shows that the local feedback control gains of the robotic vehicles must be decreased if the communication sample period is increases. Therefore, there is a tight coupling between communications and controls that cannot be ignored. In general, a system will be more responsive and have shorter settling times if the feedback control gains are as large as possible and

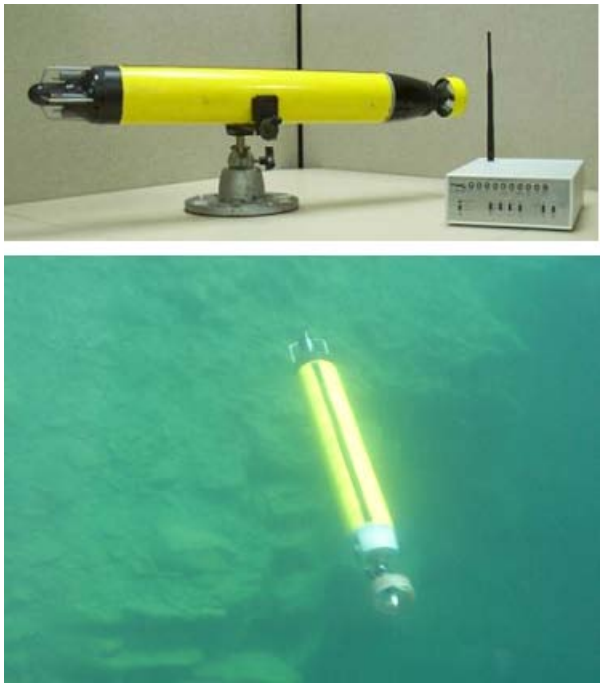

Figure 13: Nekton Research underwater vehicle used to locate synthetic plume source.

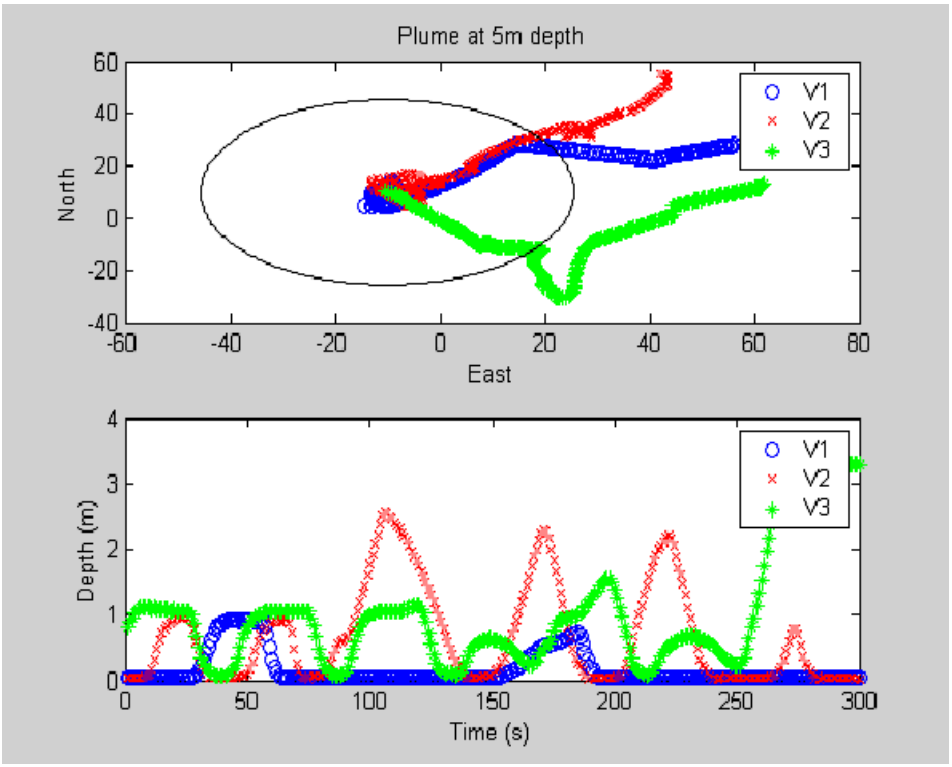

Figure 14: Underwater synthetic plume test results.

the communication sample period is as short as possible. This section evaluates the resulting communication sample period of four different communication protocols: a Time Division Multiple Access (TDMA) linear broadcast, a TDMA polylogarithmic broadcast, a TDMA coloring algorithm, and a Collision Sense Multiple Access (CSMA) coloring algorithm. The selection of the best protocol depends on the density of the robot vehicles and the communication radius of each vehicle.

Throughout this section, the one-dimensional dispersion example from the previous section is used to illustrate the design methodology. In [18], it is shown that Equation (5) is actually more constrained than $0 < \alpha \leq 1$ depending on the speed of the vehicle and the communication sample period. Therefore, the next question to ask is that of connective stability. Under what conditions will the overall system be globally asymptotically stable even under structural perturbations? Analysis of connective stability is based upon the concept of vector Liapunov functions, which associates several scalar functions with a dynamic system in such a way that each function guarantees stability in different portions of the state space. The objective is to prove that there exist Liapunov functions for each of the individual subsystems and then prove that the vector sum of these Liapunov functions is a Liapunov function for the entire system.

3.1 Stability Analysis

To simplify matters, we will assume that the control function has already been chosen and the closed loop dynamics of the discrete time system can be written as

$$S: \quad x_i(k+1) = g_i(k, x_i) + \tilde{g}_i(k, \bar{x}), \quad i \in \{1, \ldots, N\} \tag{42}$$

where $\bar{x}(k) \in \mathbb{R}^{\mathrm{n}}$ is the state of S (e.g., x, y position, orientation, and linear and angular velocities of all vehicles) at time $k \in T$, $x_i(k) \in \mathbb{R}^{\mathrm{n_i}}$ is the state of the i_{th} subsystem S_i at time $k \in T$. The function $g_i : T \times \mathbb{R}^{\mathrm{n_i}} \rightarrow \mathbb{R}^{\mathrm{n_i}}$ describes the local dynamics of S_i , and the function $\tilde{g}_i : T \times \mathbb{R}^{\mathrm{n}} \rightarrow \mathbb{R}^{\mathrm{n_i}}$ represents the dynamic interaction of S_i with the rest of the system S. The interconnection function can be written as

$$\tilde{g}_i(k, \bar{x}) = \tilde{g}_i(k, \bar{e}_{i1}x_1, \bar{e}_{i2}x_2, \ldots, \bar{e}_{iN}x_N) \quad i \in \{1, \ldots, N\}. \tag{43}$$

where $\bar{e}_{ij} \in B^{n_i \times n_j}$, and the elements of the fundamental interconnection matrix $\bar{E} = (\bar{e}_{ij})$ are

$$(\bar{e}_{ij})_{pq} = \begin{cases} 1, & (x_j)_q \text{ occurs in } (\tilde{g}_i(t, x, u))_p \\ 0, & (x_j)_q \text{ does not occur in } (\tilde{g}_i(t, x, u))_p \end{cases} \tag{44}$$

where $q \in \{n_j\}$ and $p \in \{n_j\}$.

The structural perturbations of S are introduced by assuming that the elements of the fundamental interconnection matrix that are one can be replaced by any number

between zero and one, i.e.

$$e_{ij} = \begin{cases} [0,1], & \bar{e}_{ij} = 1 \\ 0, & \bar{e}_{ij} = 0 \end{cases} \tag{45}$$

Therefore, the elements e_{ij} represent the strength of coupling between the individual subsystems. A system is connectively stable if it is stable in the sense of Liapunov for all possible $E = (e_{ij})$ [29]. In other words, if a system is connectively stable, it is stable even if an interconnection becomes decoupled, i.e. $e_{ij} = 0$, or if interconnection parameters are perturbed, i.e. $0 < e_{ij} < 1$. This is potentially very powerful, as it proves that the system will be stable even if an interconnection is lost through communication failure.

For linear systems, the discrete time dynamics may be written as

$$S: \quad x_i(k+1) = A_{ii}x_i(k) + \sum_{j=1}^{N} e_{ij}A_{ij}x_j(k), \quad i \in \{1, \ldots, N\} \tag{46}$$

and the Liapunov function for each individual subsystems is $v_i(x_i) = (x_i^T P_i x_i)^{\frac{1}{2}}$ where P_i is a positive definite matrix. For the system S to be connectively stable, the following test matrix $W = (w_{ij})$ must be an M-matrix (i.e., all leading principal minors must be positive) [30]:

$$w_{ij} = \begin{cases} \xi_i, & i = j \\ -e_{ij}\xi_{ij}, & i \neq j \end{cases} \tag{47}$$

where $\xi_i = 1 - \sqrt{1 - \frac{1}{\lambda_M(H_i^*)}}$, $\xi_{ij} = \lambda_M^{\frac{1}{2}}(A_{ij}^T A_{ij})$, and $A_{ii}^T P_i^* A_{ii} - H_i^* = -I$, $\lambda_m(\bullet)$ and $\lambda_M(\bullet)$ are the minimum and maximum eigenvalues of the corresponding matrices, and the superscript $*$ denotes the Hermitian operator.

For the linear dispersion example, we will model the vehicle dynamics as a discrete time integrator with a position feedback loop (see Figure 15). The proportional control gain is K_p, and the sampling period is T.

The sampling period is both the communication and position update sample time. The state equations of the system are

$$\begin{aligned} S: \quad x_1(k+1) &= (1 - K_pT)x_1(k) + \gamma K_pT x_2(k) \\ x_i(k+1) &= (1 - K_pT)x_i(k) + \gamma K_pT x_{i-1}(k) + \gamma K_pT x_{i+1}(k), \quad i \in \{2, \ldots, N-1\} \\ x_N(k+1) &= (1 - K_pT)x_N(k) + \gamma K_pT x_{N-1}(k) \end{aligned} \tag{48}$$

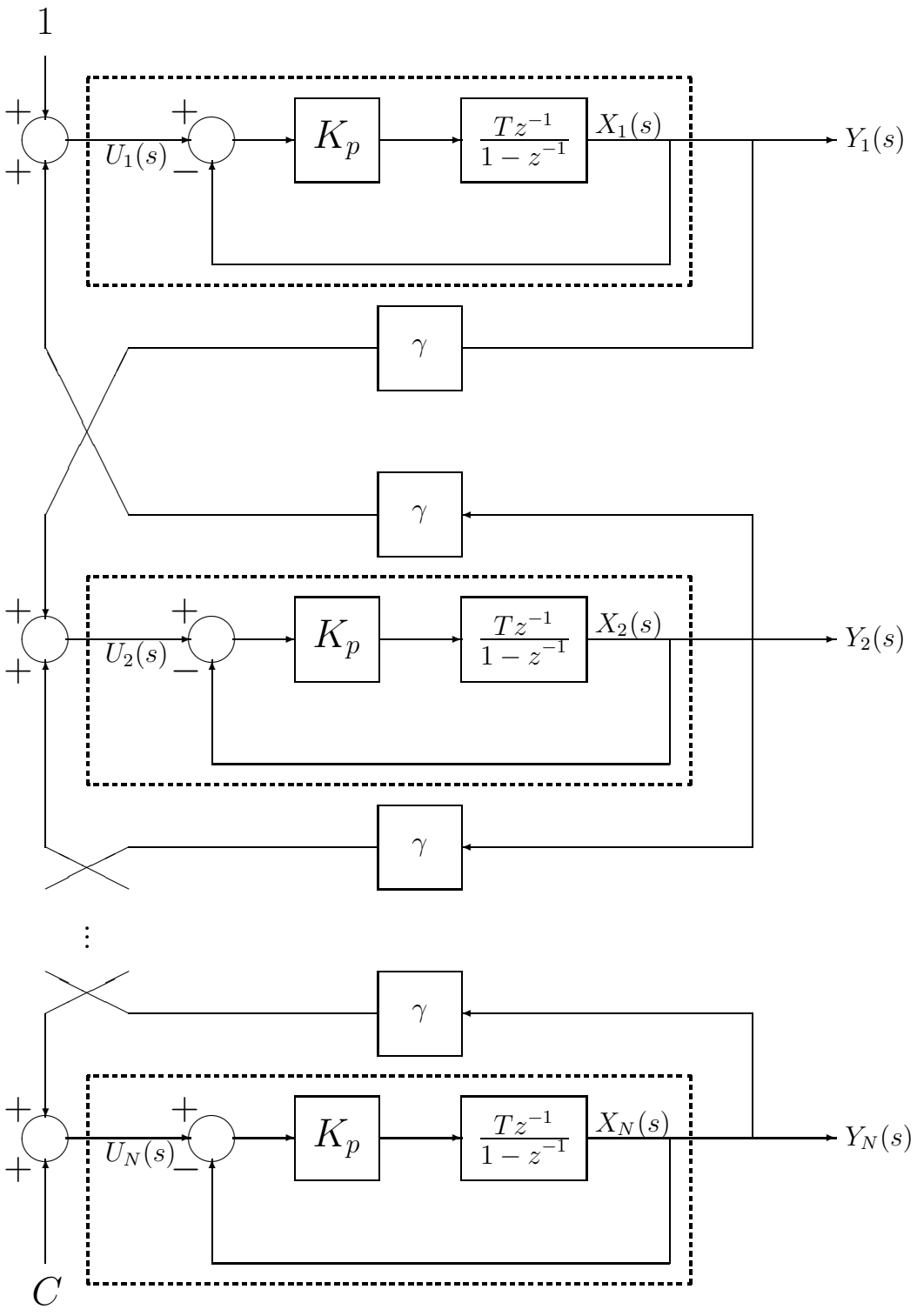

Figure 15: Discrete time control block diagram of N-vehicle interaction problem.

Note that when comparing Equation (48) to Equations (5) and (6), it is evident that $2\alpha = K_pT$ and $\alpha = \gamma K_pT$. If Equation (48) is forced to be exactly equivalent to Equations (5) and (6), then $\gamma = \frac{1}{2}$ and $\alpha = \frac{K_pT}{2}$. The following stability test is less restrictive, and the interaction gain γ is less constrained. If $0 < K_pT \leq 1$, the resulting test matrix is

$$W = \begin{bmatrix} K_pT & -K_pT\gamma & 0 & \dots & 0 \\ -K_pT\gamma & K_pT & -K_pT\gamma & & \vdots \\ 0 & -K_pT\gamma & K_pT & & 0 \\ \vdots & & & \ddots & -K_pT\gamma \\ 0 & \dots & 0 & -K_pT\gamma & K_pT \end{bmatrix}, \tag{49}$$

and if $1 < K_pT \leq 2$, the test matrix is

$$W = \begin{bmatrix} (2-K_pT) & -K_pT\gamma & 0 & \dots & 0 \\ -K_pT\gamma & (2-K_pT) & -K_pT\gamma & & \vdots \\ 0 & -K_pT\gamma & (2-K_pT) & & 0 \\ \vdots & & & \ddots & -K_pT\gamma \\ 0 & \dots & 0 & -K_pT\gamma & (2-K_pT) \end{bmatrix}, \tag{50}$$

For $N = 2$, the test matrix is an M-matrix, and the system is connectively stable if

$$|\gamma| < \begin{cases} 1, & 0 < K_pT \leq 1 \\ \frac{2}{K_pT} - 1, & 1 < K_pT \leq 2 \end{cases} \tag{51}$$

Figure 16 illustrates the stability region for the case of N=2. The dark region represents stable combinations of the interaction gain γ and K_pT(proportional control gain multiplied by the sampling period). The white region represents unstable combinations of γ and K_pT. We refer to the dark region as a stability "house" due to the shape of the stable zone. The size of this stability house varies only with N. As N is increased, the house gets smaller in width but maintains the same height and shape. The size of the stability house is a measure of the robustness of the closed-loop system to parameter variations in interaction gain γ, sampling period T, and proportional control gain K_p. Figure 17 shows the stability region for $N = 10000$.

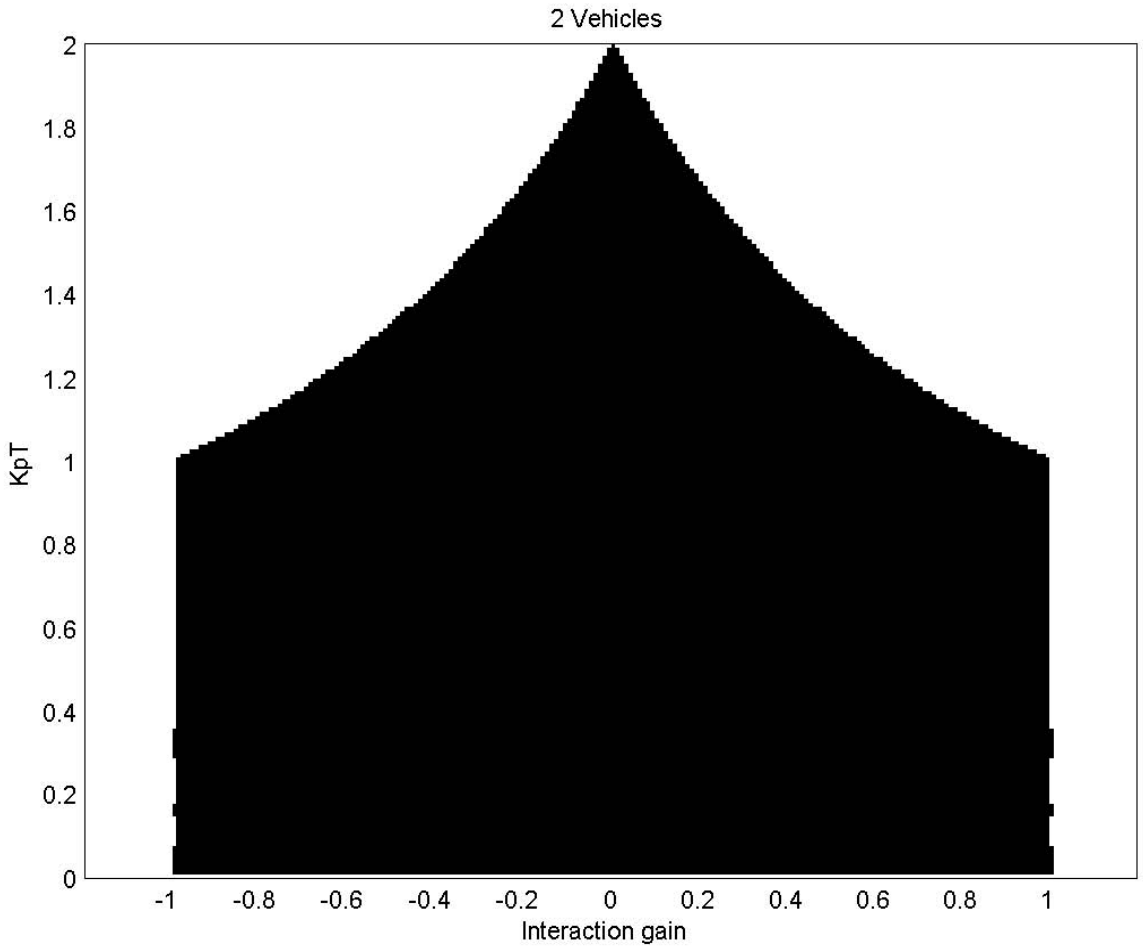

Figure 16: Stability region for the N=2 vehicle case.

For this particular example, another way to check the stability of this linear system is to check that the eigenvalues of the system matrix A are within the unit circle. There is a special formula (p. 59 of [31]) for the eigenvalues of A given by

$$\lambda_i(A) = 1 - K_pT + 2K_pT\gamma\cos(\frac{i\pi}{N+1}), \quad i = 1, \ldots, N \tag{52}$$

From this formula, we can see that as $N \to \infty$, the cosine term becomes unity. This implies that γ must stay between -0.5 and 0.5 for K_pT less than one in order to maintain stability. For K_pT greater than one, the admissible γ values taper off parabolically (the sloped "roof") until $K_pT = 2$.

3.2 Communications Sample Period

Several conclusions can be drawn from this stability analysis. First, asymptotic stability of vehicle positions depends on vehicle responsiveness K_p, communication sampling period T, and vehicle interaction gain γ. If the vehicle is too fast (large K_p) or

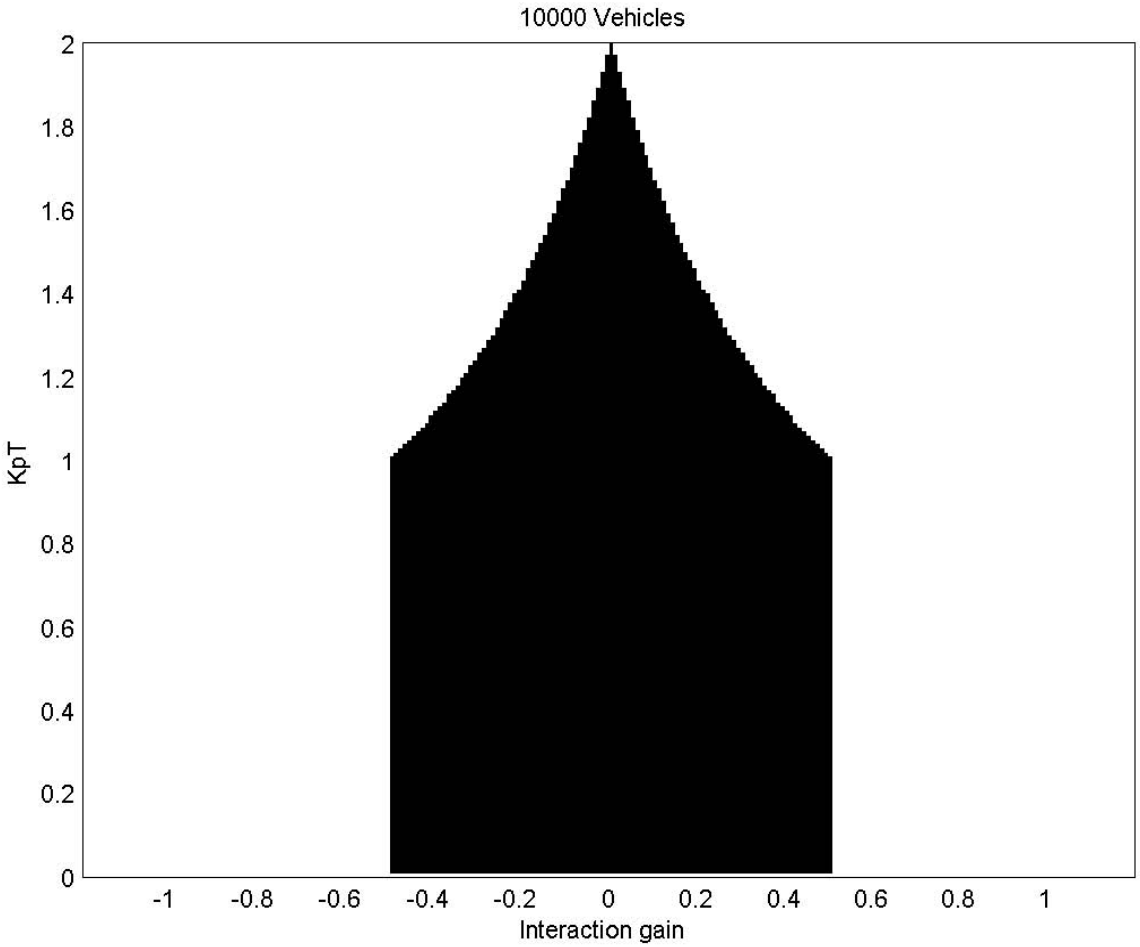

Figure 17: Stability region for the N=10000 vehicle case.

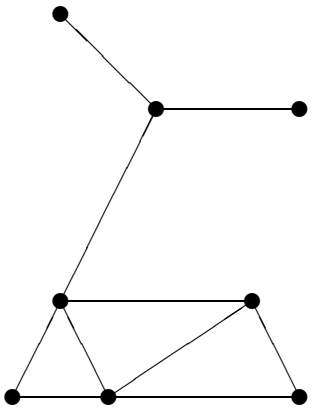

Figure 18: Graph of ad hoc communication network. Nodes with connecting lines can communicate with each other.

the sample period is too long (large T) then the vehicles will go unstable. There is a dependence on interaction gain for stability as well. Second, the interaction gains can be used to bunch the vehicles closer together or spread them out. Third, the stability region shrinks as the number of vehicles, N, increases but only to a defined limit.

As noted, the communication sample period greatly affects the stability of the system. As defined in the equations above, this sample period is the time it takes for every node to communicate once. In this section, we will evaluate the communication sample period of four different communication schemes: a Time Division Multiple Access (TDMA) linear broadcast, a TDMA polylogarithmic broadcast, a TDMA coloring algorithm, and a Collision Sense Multiple Access (CSMA) coloring algorithm. All of these schemes assume that each node has a unique identification number. The TDMA schemes also assume that each node has a synchronized clock that is used to notify each node when it may transmit a message. The CSMA scheme first checks the communication channel for a collision before transmitting a packet.

In order to determine this sample period, one important parameter associated with an ad hoc communication network is the degree of the network. The degree of the network is defined as the maximum number of nodes that any node can communicate with given a limited communication range.

For the network shown in Figure 18, the degree of the network is

$$\Delta = \max_{i \in \{1,\ldots,N\}} \left\{ \sum_{j=1, j \neq i}^{N} range(x_i - x_j, R_c) \right\} \tag{53}$$

where

$$range(x_i - x_j, R_c) = \begin{cases} 1 & if \quad |x_i - x_j| < R_c \\ 0 & else \end{cases} \tag{54}$$

and R_c is the communication radius of each node. Assuming all the robots are evenly spaced along a line of length L and have a density $\delta = \frac{N}{L}$, then the degree of the resulting network is

$$\Delta = 2 \lfloor \delta R_c \rfloor = 2 \left\lfloor \frac{N R_c}{L} \right\rfloor \tag{55}$$

For the TDMA linear broadcast where every node is assigned a unique identification number, the communication sample period time required for every node to send a message is

$$T_{linear} = \tau N \tag{56}$$

where τ is the time period associated with each communication time slot. Notice that the above expression is proportional to N. This delay time can be shortened by using a polylogarithmic broadcast scheme [32] where each node communicates during multiple time slots. Even though multiple messages are being broadcast at the same time, the message is guaranteed a successful broadcast during one of the time slots as long as the degree of the network is below a certain value. The communication period for a polylogarithmic broadcast is

$$T_{polylog} = \tau \left(2 \log_2 N\right)^h \tag{57}$$

when

$$\Delta \leq 2^{h+1} - 1 \tag{58}$$

and where

$$h = \left\lfloor \frac{log_2 N}{(\log_2 \log_2 N + 1)} \right\rfloor \tag{59}$$

Notice that this expression is proportional to the $\log_2 N$ instead of N. Figure 19 compares the linear broadcast to the polylogarithmic broadcast for a network spread out over a line and with each node having a communication radius that is one-tenth the length of the line. At 20 robots, the average degree of the network becomes too large and the polylogarithmic broadcast will no longer work. Even better than the linear broadcast and the polylogarithmic broadcast, a coloring scheme allows multiple nodes to communicate at the same time by using spatial reuse of time slots. Time slots (called colors) are assigned so that each node has a different color than its first and second nearest neighbors. By using different colors, the hidden node problem, where two nodes speak to an intermediate node at the same time, is eliminated. In graph theory, the minimum number of colors can range from the maximum degree of the network plus one to the square of the maximum degree of the network plus one.

$$\Delta + 1 \leq k \leq \Delta^2 + 1 \tag{60}$$

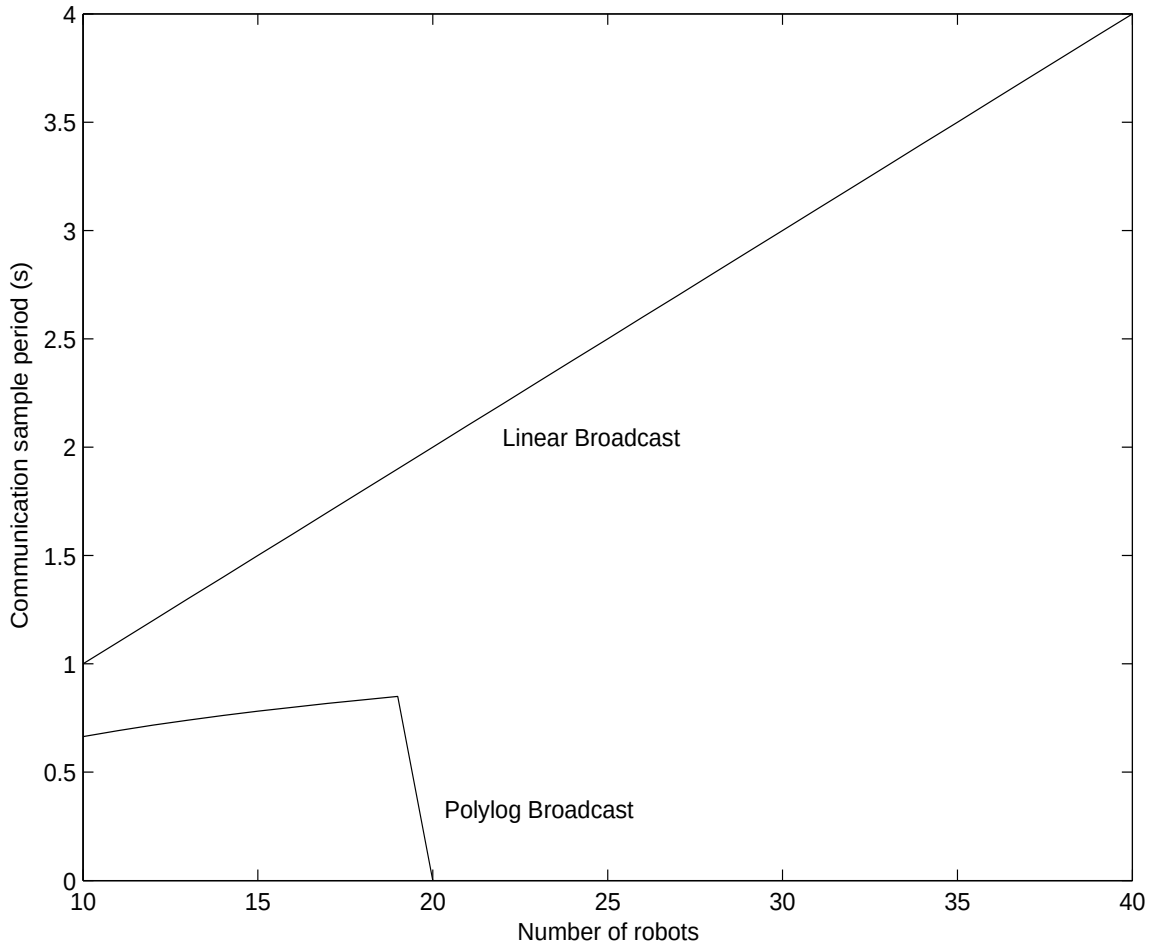

Figure 19: Communication sample period for TDMA linear and polylogarithmic broadcasts when $\frac{R_c}{L} = 0.1$ and $\tau = 0.1s$.

However, in a typical planar wireless network the number of colors is typically bounded by

$$k \leq \Delta + \epsilon \tag{61}$$

where ϵ is a small number, typically 1 to 5.

For a colored TDMA network, the communication sample period is given by

$$T_{TDMA} = \tau(\Delta + \epsilon) \tag{62}$$

where ϵ is small.

For a CSMA network, the actual communication time is non-deterministic because the packets often collide and a random back-off is used before retransmitting. The average communication time depends on the network utilization, i.e. the percentage of time the network is being used (see Section 5 for more details). Modeling the CSMA network as a M/D/1 queue, the average communication time per node is

$$\tau_{CSMA} = \frac{\rho T_m}{2(1-\rho)} + T_m \tag{63}$$

where T_m is the time to send a single message (or service time of the queue). For simplicity of comparison with the TDMA network, we will assume that $T_m = \tau$. In reality, the message length T_m in a TDMA network must be slightly less than the time slot τ. The utilization factor is

$$\rho = \frac{T_m(\Delta + \epsilon)}{T_d} \tag{64}$$

where T_d is the delay time between each new message that a node initiates. For a CSMA network with degree Δ, the communication sample period is approximately given by

$$T_{CSMA} = \tau_{CSMA}(\Delta + \epsilon) \tag{65}$$

Figure 20 shows how the network utilization changes as the number of robots increases in our one-dimensional dispersion example. The resulting communication sampling period for both the TDMA and CSMA colored networks are shown in Figure 21.

Notice that for smaller numbers of vehicles both the colored TDMA and CSMA networks have a shorter communication sample period than both linear and polylogarithmic broadcasts. However, when the utilization of the CSMA network reaches 0.9, the CSMA network starts to substantially degrade in performance. This is caused by flooding the network with messages as the density of the robots increase while the back-off time of communication stays the same. It should be noted that this flooding can be alleviated if the nodes where to adjust how often they send out messages based on the maximum degree of the network. Ideally, the nodes should adjust their communication sample period so that $T_d = T_{CSMA}$.

These results show that a colored TDMA network appears to perform the best. However, the disadvantage of the colored TDMA network is that there is an initialization time that is required to determine the color of each node whenever the network

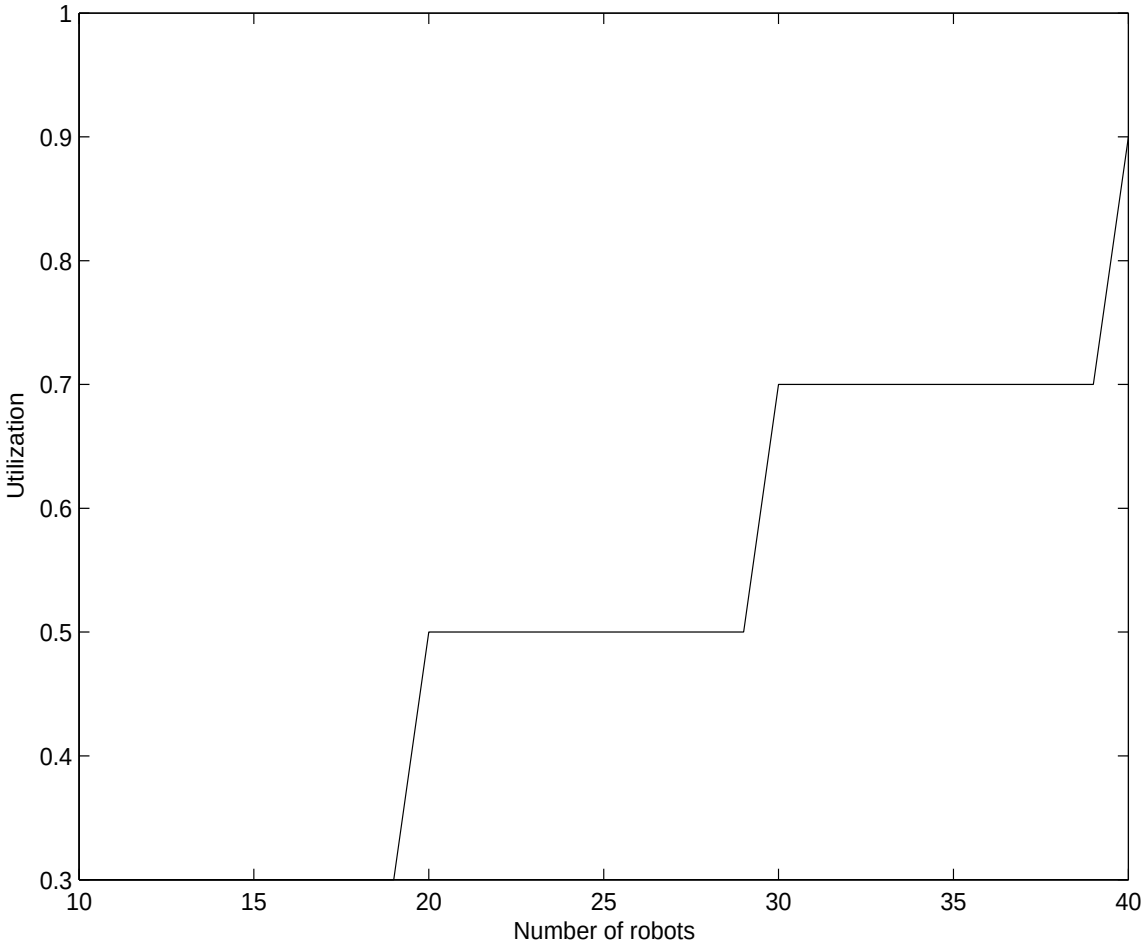

Figure 20: Utilization for a CSMA network when $\frac{R_c}{L} = 0.1s$, $\tau = 0.1s$, $\epsilon = 1$, and $T_d = 1s$.

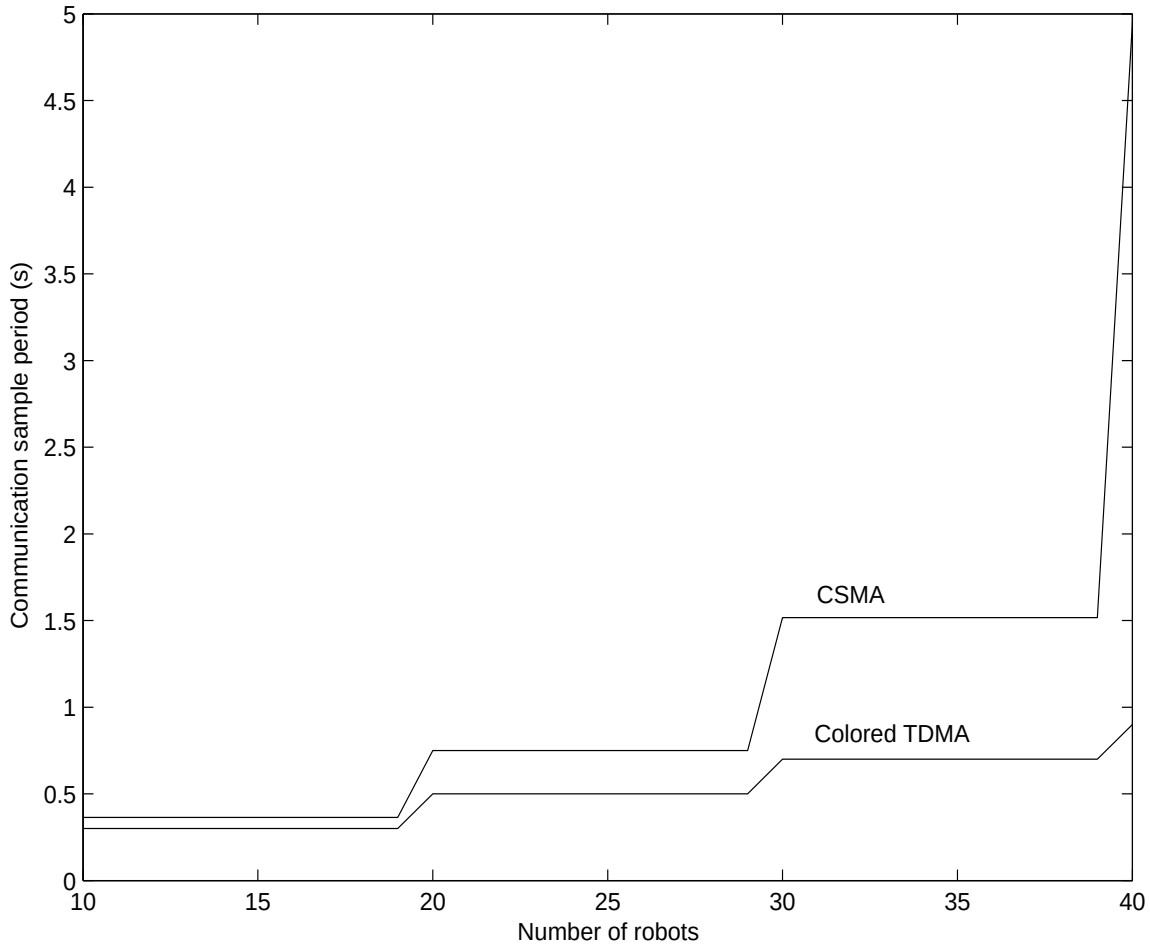

Figure 21: Communication sample period for TDMA and CSMA reconfigurable coloring when $\frac{R_c}{L} = 0.1s$, $\tau = 0.1s$, $\epsilon = 1$, and $T_d = 1s$.

topology changes. The other algorithms have the advantage that they do not require a network initialization time. Using an algorithm by [33], this initialization time is the time required to broadcast their own identification number, their 1^{st} nearest neighbor lists (after which each node can determine 2^{nd} nearest neighbor), and $k+2$ additional messages for coloring. In addition to these messages, each neighboring node must acknowledge the messages containing the neighbor lists and coloring information to make ensure that the messages were received. Since time slots are typically not assigned before hand, this initialization process occurs using CSMA protocols, and it should be performed whenever the topology of the network changes, i.e. when robots move. Assuming that all messages are the same length, the resulting initialization time is given by

$$\begin{aligned} t_{init} &= \tau_{CSMA}\left[2(\Delta+\epsilon)+(\Delta+\epsilon+2)(\Delta+\epsilon)\right]+ \\ &\quad \tau_{CSMA}\left[(\Delta+\epsilon)(\Delta+\epsilon-1)+(\Delta+\epsilon)(\Delta+\epsilon+2)(\Delta+\epsilon-1)\right] \end{aligned} \tag{66}$$

where ϵ is small. This initialization time is plotted as a function of the number of robots in Figure 22. This figure shows that this initialization time can be substantial. Adding the initialization time in Figure 22 to the communication sample period in Figure 21, we see that the linear, polylogarithmic, and CSMA networks have a shorter sample period than the colored TDMA network. In general, it is best to use the TDMA mode only if the network does not reconfigure; otherwise, it is best to use a CSMA mode of communications.

3.3 Utilization versus Delay in CSMA Networks

If a control system is implemented with a CSMA communications scheme, there is a tradeoff between network utilization and message delay. Ideally, one would like to maximize network utilization while minimizing the delay seen by each message. Modeling the CSMA network as an M/D/1 queue, the normalized delay (where T_m is the message length) is given by

$$\frac{\tau_{CSMA}}{T_m} = \frac{\rho}{2(1-\rho)} + 1 \tag{67}$$

where τ_{CSMA} is the average communication time. Using the cost function $J(\rho)$ below, the gains K_1 and K_2 may be used to weight the penalty associated with utilization and delay.

$$J(\rho) = \frac{K_1}{\rho} + \frac{K_2\rho}{2(1-\rho)} + K_2 \tag{68}$$

The optimal solution is then obtained by minimizing the cost function over $0 \leq \rho < 1$.

$$J(\rho^*) = \min_{0\leq\rho<1} J(\rho) = \min_{0\leq\rho<1}\left\{\frac{K_1}{\rho} + \frac{K_2\rho}{2(1-\rho)} + K_2\right\} \tag{69}$$

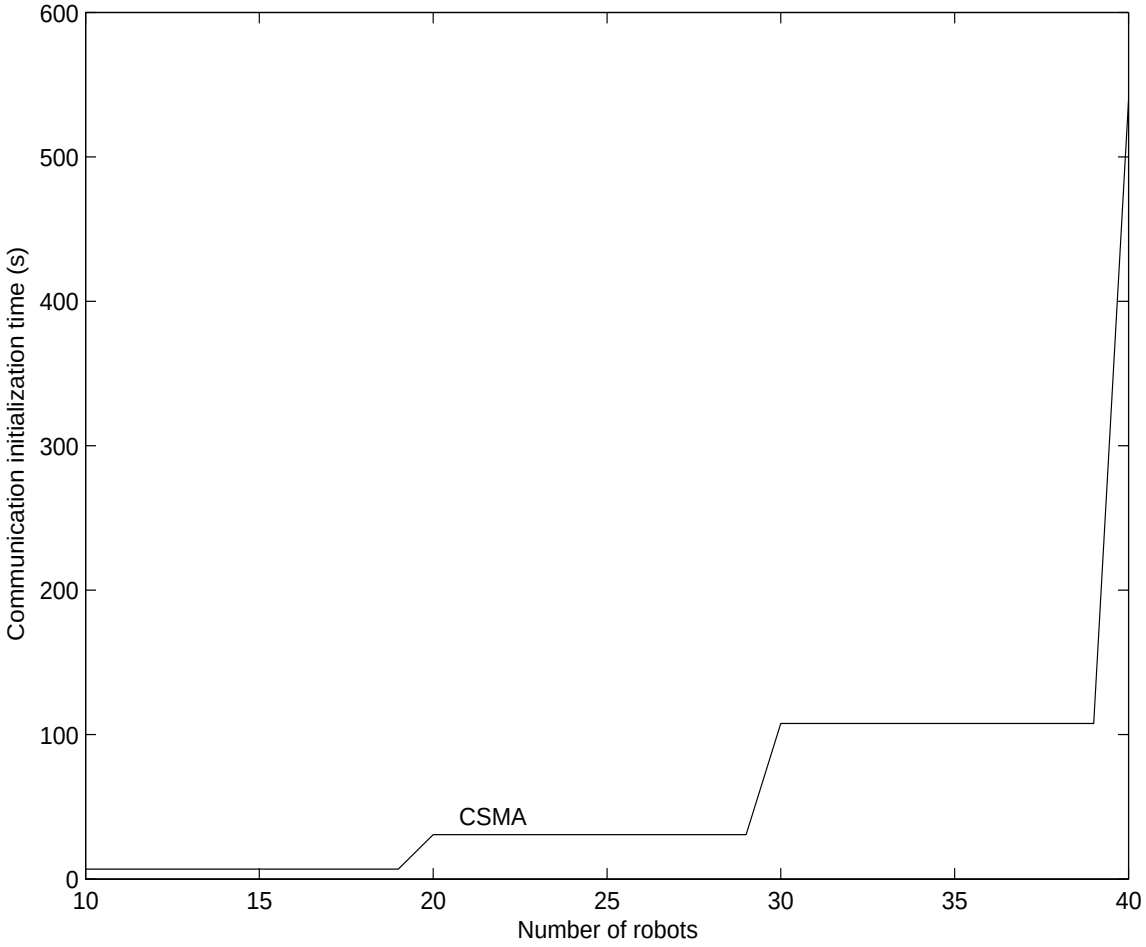

Figure 22: Communication overhead for reconfigurable coloring when $\frac{R_c}{L} = 0.1$, $\tau = 0.1s$, $\epsilon = 1$, and $T_d = 1s$.

Weighting Gains	**Optimal Utilization, ρ^***
$K_1 = K_2 = 1$	$\rho^* = 0.5858$
$K_1 = 1,\ K_2 = 10$	$\rho^* = 0.309$
$K_1 = 0.1,\ K_2 = 10$	$\rho^* = 0.124$

Table 2: Optimal Utilization for Different Values of K_1 and K_2

Differentiating equation (69) with respect to ρ yields

$$\frac{\partial J(\rho)}{\partial \rho} = -\frac{K_1}{\rho^2} + \frac{2(1-\rho)K_2 + 2K_2\rho}{4(1-\rho)^2} \tag{70}$$

Solving equation (70) for the minimum gives the following optimal value for network utilization ρ^* based on the weighting values K_1 and K_2.

$$\rho^* = \frac{-2K_1 \pm \sqrt{2}\sqrt{K_1 K_2}}{(K_2 - 2K_1)},\ 0 \leq \rho^* < 1 \tag{71}$$

If utilization and average delay are equally weighted, the optimal value of ρ is $\rho^* = 0.5858$. Optimal utilization values for several values of K_1 and K_2 are summarized in Table 2. These results are intuitive - if average message delay is critical, then network utilization should be kept to a minimum.

This section illustrates the tight coupling that exists between communications and controls when designing large-scale cooperative robotic systems. A connective stability analysis shows that local feedback control gains and communication sample periods are inversely related. If the communication sample period increases, then the local feedback control gains must decrease. The communication sample period is a function of the protocol, and the protocol with the shortest communication sample period depends on the density of robots and the communication radius. By assuming worst-case conditions for robot density and communication range, this analysis can be used off-line to determine conservative control gains required for stable control. In the future, it might also be possible to use this analysis on-line to adjust control gains and/or communication range as the robot density changes. The next section presents simulation results for the node organization algorithm described in [33].

4 Analysis and Simulation of TDMA and Coloring

This section presents OPNET^{TM} simulation results for the node organization algorithm described in [33]. The algorithm, proposed by Chlamtac and Pinter, was

intended to provide collision free channel allocation in a multihop radio network. The algorithm describes the use of a graph coloring algorithm to be used for channel access allocation in an ad hoc wireless network. This algorithm will complete when each node in the network has been assigned a "color" or channel for use. Once the algorithm is complete, nodes can communicate in the network using a TDMA scheme based on the channels assigned in the algorithm.

There are several assumptions made about the radio network in order to guarantee the proper operation of the algorithm [33].

A1: Nodes have distinct identities and know the identities of their neighbors. (We assume the existence of a physical layer for mutual location and identification of stations).

A2: Links are bidirectional. (The case of directional links is discussed in [33]).

A3: A message sent by a node is received correctly within a finite time by all its neighbors.

A4: Control messages arrive in finite but undetermined time.

A5: Network topology does not change during the algorithm execution (this may be relaxed).

Using these assumptions, each node keeps the following lists:

LOCAL.Neighbors which contains all the neighbors of the node (1-hop).

LOCAL.Receive_Neighbors consists of a flag for every neighbor indicating whether a neighbor has already sent its list of neighbors.

LOCAL.Neighbor&2-Neighbors stores for every neighbor and 2-neighbor, its (assigned) Slot_Number (DUMMY at initialization) and a flag Slot_Assigned indicating that a slot assignment has been accomplished.

LOCAL.ID unique local identification number.

LOCAL.Slot slot number for the node that is assigned with the algorithm.

LOCAL.Node_Awake boolean variable that is **false** at initialization.

A node starts participating in the algorithm either on receiving a **WAKE** message or by the reception of a message from another node executing the algorithm. On entering the algorithm each node executes the following sequence:

- Each node broadcasts a message NEIGHBORS(LOCAL.Neighbors and LOCAL_ID) consisting of its list of neighbors. Nodes which receive this message use it for constructing LOCAL.Neighbor&2-Neighbors.

- On completion of the list LOCAL.Receive_Neighbors, the node is waiting for the development of proper conditions for selecting a slot. At a node i, this condition is obtained when i becomes the node with the highest ID in LOCAL.Receive_Neighbors whose Slot_Assigned flag is **false**. Following the slot selection, node i transmits a SLOT message to its neighbors who forward it to i's 2-neighbors.

Mechanisms for dynamically adding and deleting nodes are also presented in [33].

In order to implement this algorithm three steps must be taken:

Step 1 Announcement - each node announces itself by broadcasting its network ID. Every node receiving a node announcement packet will add the sending node to its list of neighbors.

Step 2 Neighbor Broadcast - all nodes broadcast their neighbor list to the network. Nodes receiving a neighbor list process the list and add any new nodes in the received list as a level two neighbor.

Step 3 Channel Allocation - when i becomes the node with the highest ID in LOCAL.Receive_Neighbors whose Slot_Assigned flag is **false**. Following the slot selection, node i transmits a SLOT message to its neighbors who forward it to i's 2-neighbors.

Step 1 is required to meet assumption **A1** which requires knowledge of neighbors. Step 3 is the actual channel allocation step. The node with the largest ID number in its list of level 1 and level 2 neighbors gets to assign itself a channel. The node picks the smallest channel that has not been used by any of its level-one neighbors. This channel assignment is then sent to the neighbors who receive, process, and then forward the assignment. (Only level 1 neighbors need to forward the assignment). As each node becomes the largest unassigned node in its list, they will assign themselves a channel.

Once all nodes have completed channel assignment, data communication in the network can begin.

The algorithm presented in the paper included an enhancement for mobile networks. The complication of getting the standard algorithm to work precluded any development into this area. The next section addresses the difficulties experienced when simulating the algorithm.

4.1 Implementation Issues

The algorithm as written makes an assumption that every packet which is sent in the network is received by all necessary recipients. Unfortunately, in a network, this assumption does not hold true unless there is a mechanism to guarantee the receipt of messages in the physical layer. Because of this, packets which should have been received (i.e. neighbor lists and slot announcements) get lost in the network usually due to collisions. The problem becomes one of assumptions; a node does not know

what it doesn't know. In simulating this algorithm in a wireless network, packets were lost due to collisions. If a node received a node announcement, but not a neighbor list from this node; the next step of the algorithm (channel selection) would be inaccurate due to missing information.

Because of these difficulties, a system of acknowledgments was needed to ensure nodes had the information they needed before proceeding to the next step. In Step 2 (neighbor list exchange), a node receiving a neighbor list packet will then send a receipt or acknowledgment packet to the sender. Node will not leave Step 2 until they have received receipts from all nodes they consider level-one neighbors. A similar situation occurs in Step 3 where level-one neighbors will send a receipt when they receive a channel allocation packet from a level-one neighbor. Nodes receiving forwarded channel allocation packets will not send receipts.

In order to make the network more stable, the receipt packets are given a higher priority than other setup message packets. This ensures that nodes will have to re-send their message packets less often.

Another situation created by the lossy nature of the communication channel is the necessity of retries. If a node does not receive all of the receipts from its neighbors, it will need to re-send the packet which did not get acknowledged.

If a node misses an allocation packet, there is a mechanism for requesting the packet built into the simulation as well. This situation arises when a node does not receive a forwarded channel allocation packet. A node will request a channel if it receives a channel allocation for a node which is smaller than the node without a channel. This works as a result of the ordered assignment of channels from largest node ID to smallest.

4.2 Simulation Results

After achieving a working simulation, the following results were calculated.

- Setup Time: the time it takes for all nodes to assign and share their channel value
- Degree of network
- Number of Collisions comparing CSMA and Colored TDMA
- Communication Sample Period

4.2.1 Setup time

As the degree of the network increase, the amount of time it takes for a network to become fully setup will also increase. This is due to the increased number of packets being sent and the higher probability of collisions. As can be seen in Figure 23, the setup times increase greatly as the number of nodes increase. This is due to the change in the degree of the network which will be shown in the next plot.

Radius of Communication	250 *meters*
Length	2500 *meters*
R_C/L	0.1
Backoff	uniform distribution between 50 and 150 *msec*
Packet Size	1000 *bits*
Data Rate	10,000 *bps*

Table 3: Simulation Parameters

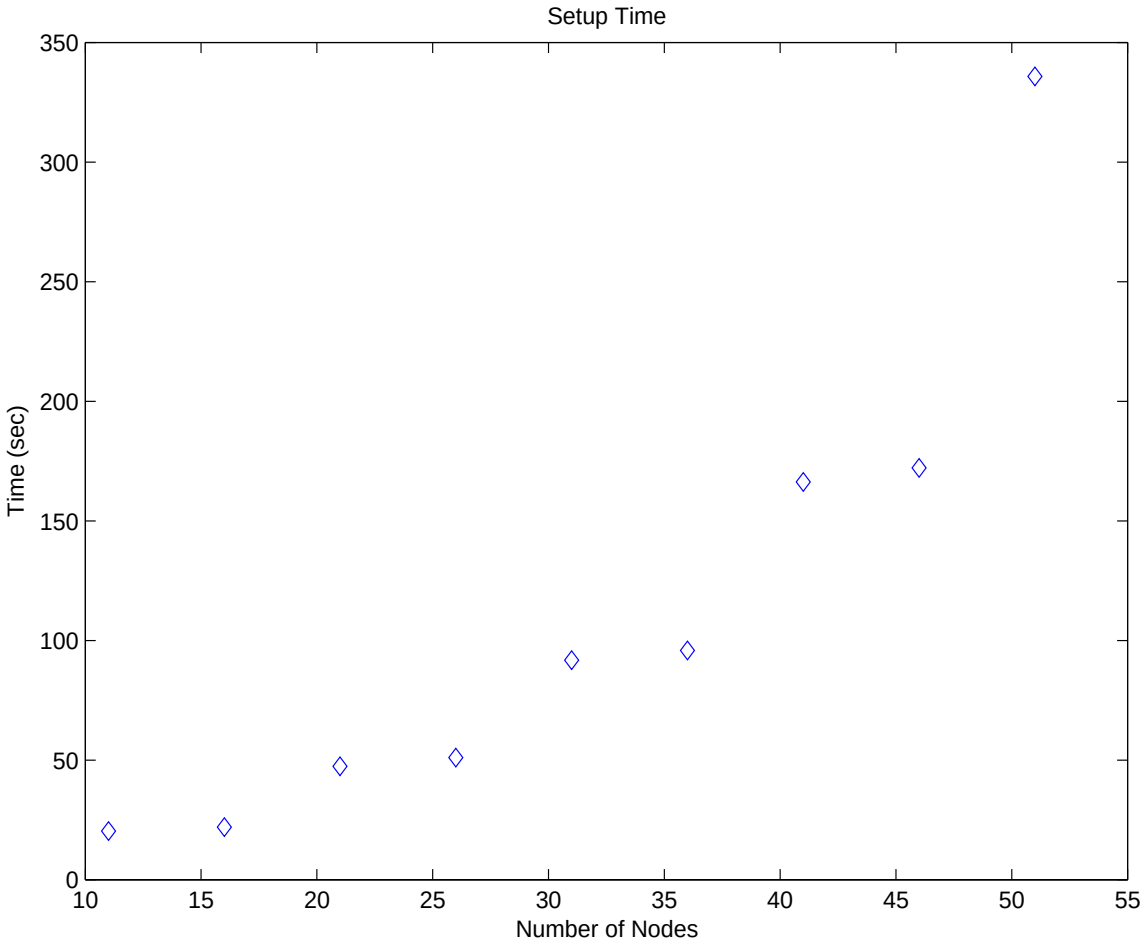

Figure 23: Setup Time

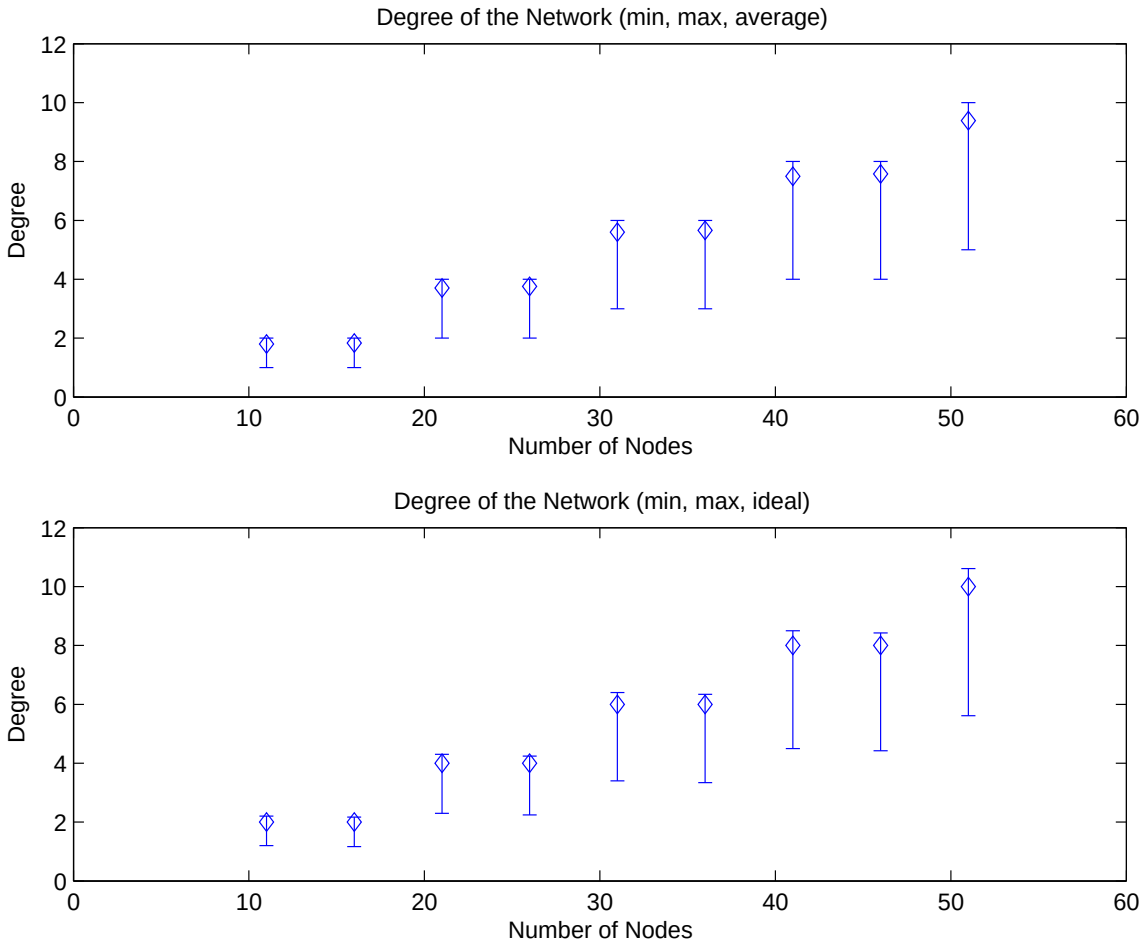

Figure 24: Degree of the Network

4.2.2 Degree of Network

The degree of the network, shown in Figure 24, was measured by taking the degree of each node in the network. The min and max values are used for confidence bars. Both the average and mean values for the degree are shown as well.

The ideal degree was calculated using

$$\text{ideal degree} = 2 \left\lfloor \frac{NR_C}{L} \right\rfloor \tag{72}$$

where N is the number of nodes, R_C is the radius of communications, L is the length of the line (assume that nodes are distributed in a line).

In the 10 node case, the degree of the network was 0 due to the distance between the nodes. When distributing 10 nodes on a 2500 meter line, the distance between each node is 277.77 meters. The simulation enforced a hard communication range of 250 meters. Therefore, none of the nodes was able to hear the other nodes.

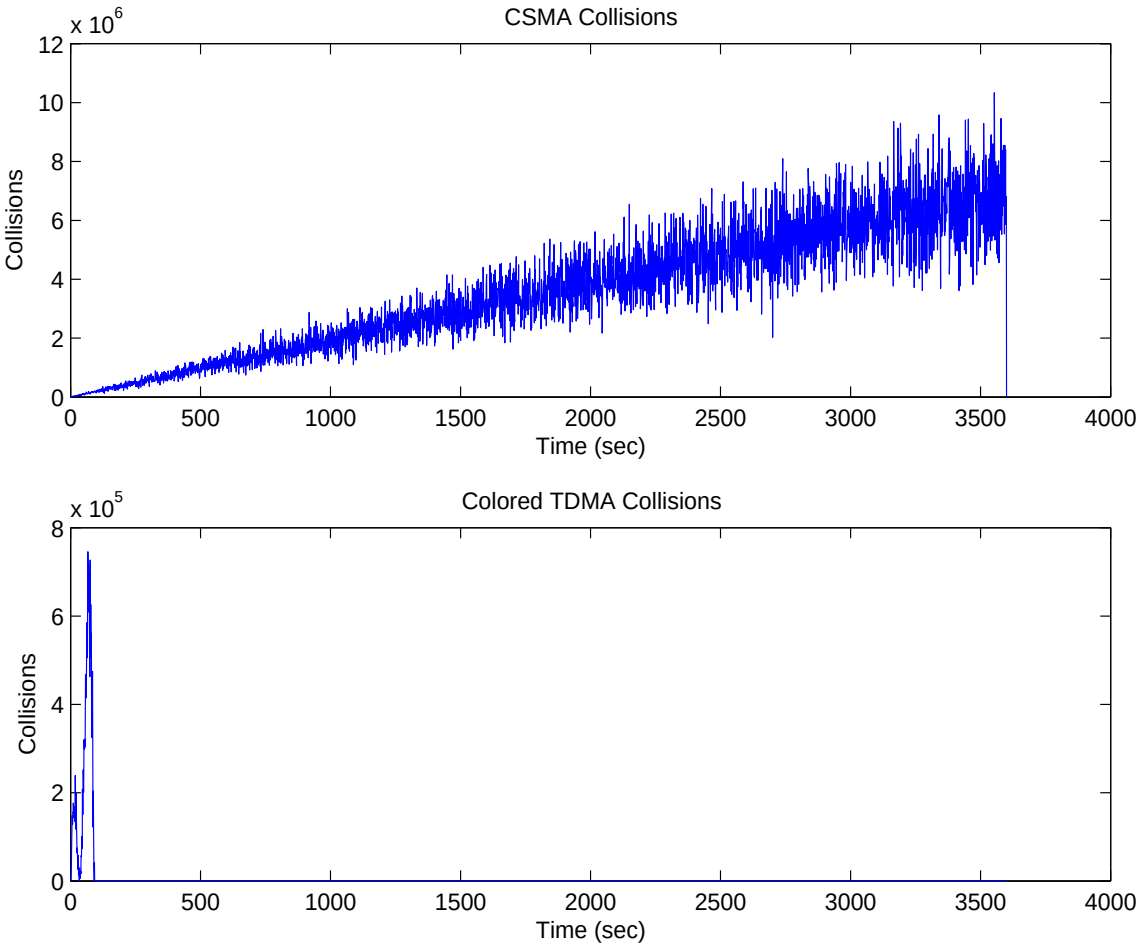

Figure 25: Number of Collisions

4.2.3 CSMA Collisions

The number of CSMA collisions is shown in Figure 25.

As this plot clearly shows, the true advantage of Colored TDMA is in the low rate of collisions after the coloring has been performed.

4.2.4 Communication Sample Period

A comparison of the communications sample period for CSMA versus colored TDMA appears in Figure 26.

The main advantage of the network organization algorithm proposed by Chlamtac and Pinter is that once configured, it provides efficient collision free channel allocation in a multi-hop radio network. However, one of the assumptions for the algorithm is that a message sent by a node is received correctly within a finite time by all neighbors (**A3**). An additional assumption (**A1**) is that nodes know the identities of their neighbors. In RF networks, where there are often collisions or dropouts, these two

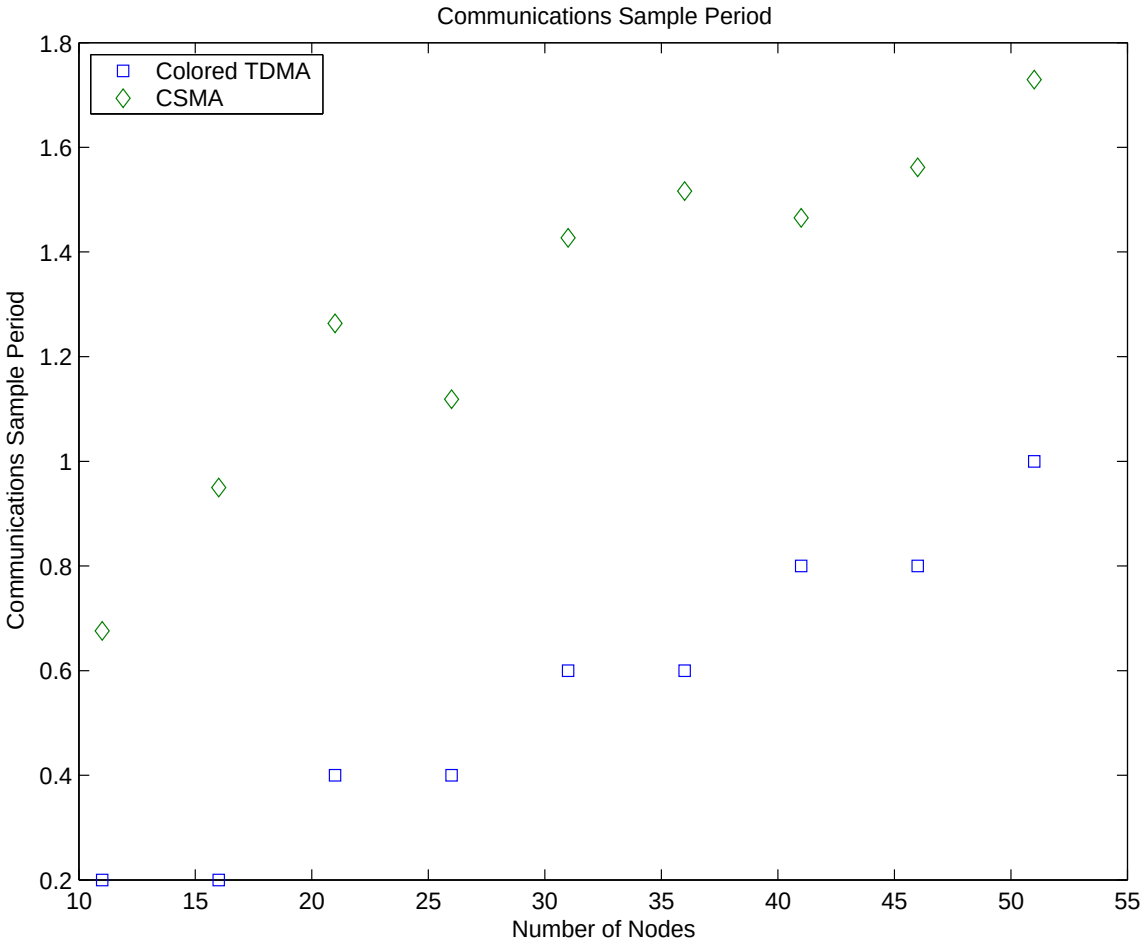

Figure 26: Communication Sample Period

assumptions require that the physical layer or a higher layer incorporate robustness to guarantee the delivery of messages. If these assumptions are not met, the algorithm encounters many difficulties. However, meeting these assumptions in a practical system adds a lot of communications overhead (e.g. message acknowledgments) that decreases the simplicity and efficiency of the algorithm.

The next section discusses the delay characteristics of CSMA communication networks.

5 CSMA Delay Characteristics and Simulation Results

This section focuses on the delay characteristics of Carrier Sense Multiple Access (CSMA) networks because the delay characteristics affect the performance of any type of networked control system. CSMA networks allow multiple users to efficiently share a single channel. The main advantage of this scheme is that each user can access the channel at any time, provided that an attempt is made to avoid collisions by listening to (e.g. sensing) the carrier due to another user's transmission [34]. Based on the state of the channel, there are different actions, often referred to as protocols, that may be taken by the user awaiting tranmission. Some common protocols include: $1-persistent$ CSMA, $p-persistent$ CSMA, and $nonpersistent$ CSMA.

For $nonpersistent$ CSMA, the user senses the channel and performs the following actions:

Step 1. If the channel is sensed idle, the user transmits the packet.

Step 2. If the channel is sensed busy, the user reschedules the transmission of the packet to some later time according to the retransmission delay distribution. At this point, Step 1 is repeated.

The retransmission delay distribution is used to decrease the chance of two users attempting to access the channel at virtually the same time and generating a collision. Common delay distributions include a uniform distribution or an exponential distribution.

The $1-persistent$ protocol is a special case of the $p-persistent$ protocol with $p = 1$. For $1-persistent$ CSMA, the user senses the channel and performs the following actions:

Step 1. If the channel is sensed idle, the user transmits the packet with probability 1.

Step 2. If the channel is sensed busy, the user waits until the channel goes idle and only then transmits the packet (with probability 1).

$p-persistent$ CSMA is a generalization of the $1-persistent$ protocol. It is assumed that the time axis is finely slotted where the (mini) slot size is τ seconds.

For simplicity of analysis, the system is synchronized such that all packets begin their transmission at the beginning of the slot [34]. For $p-persistent$ CSMA, the user senses the channel and performs the following actions:

Step 1. If the channel is sensed idle, the user transmits the packet with probability p, or with probability $1-p$ the user delays the transmission by τ seconds. If at this new time, the channel is still idle, the same process is repeated. If the channel is sensed busy, the action in Step 2 is performed.

Step 2. If the channel is sensed busy, the user reschedules the transmission of the packet to some later time according to the retransmission delay distribution. At this point, Step 1 is repeated.

5.1 CSMA Network Modeled as an M/D/1 Queue

One of the basic assumptions for analysis of CSMA networks is that the traffic source consists of an infinite number of users who collectively form an independent Poisson source with an aggregate mean packet generation rate of λ packets/sec. For a Poisson process, the probability that k arrivals occur during the time interval $(0,t)$ is given by $P_k(t)$ [35].

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad k \geq 0,\ t \geq 0 \tag{73}$$

For a random variable $\tilde{t}$, which represents the time between adjacent arrivals, the probability density function (pdf) in the Poisson case is given by

$$a(t) = \lambda e^{-\lambda t} \tag{74}$$

which is the well known exponential distribution [35].

Modeling the channel as a Poisson source enables analysis of the channel characteristics. For control applications, one of the most important characteristics is message delay. Another parameter of interest is the utilization factor ρ [35]. For a single-server system, the definition of ρ is

$$\rho = (\text{average arrival rate of customers}) \times (\text{average service time}) \tag{75}$$

For a single-channel communications network with a Poisson distribution, the average arrival rate is λ packets/second, and the average service time is the average message length T_m. Thus, equation (75) can be expressed as

$$\rho = \lambda T_m \tag{76}$$

If a CSMA network is modeled as an $M/D/1$ queue, the average message delay as a function of ρ is given by

$$\tau_{CSMA} = \frac{\rho T_m}{2(1-\rho)} + T_m \tag{77}$$

where T_m is the message length (or service time of the queue). The message delay is composed of two components, the queuing time and the message length. The notation $A/B/m$ is often used to succinctly describe the characteristics of queuing systems. The A represents the probability distribution of interarrival times of customers, the B represents the probability distribution of the service time, while the m represents the number of servers. For an $M/D/1$ queue, the interarrival times are modeled as a Markov process (exponential distribution for states), the message time is constant (Deterministic), and there is one channel so the number of servers m equals one. If the message time has an exponential distribution (M/M/1 queue) the average delay increases roughly by a factor of two. The normalized average delay may be obtained by dividing by the message length T_m which yields

$$\frac{\tau_{CSMA}}{T_m} = \frac{\rho}{2(1-\rho)} + 1 \tag{78}$$

A plot of normalized average delay as a function of network utilization ρ appears in Figure 27. While the average network delay is certainly an important parameter for a networked control system, the distribution of the delay is also a critical parameter. If the system performance or stability can be guaranteed for any delay less than δ, then the delay distribution is required to determine the probability that the delay will be less than δ. A method for predicting the delay characteristics of CSMA networks is described in [36]. An alternate method is to perform Monte Carlo simulations of the network of interest. A simple Matlab program was developed to simulate CSMA networks and analyze their delay characteristics. The m-file appears in Appendix A.

5.2 Simulation Results

One of the assumptions for an M/D/1 queue is that the traffic source consists of an infinite number of users who collectively form an independent Poisson source. In many practical networks the traffic source consists of a finite number of users that come close to collectively forming an independent Poisson source. Monte Carlo simulations were performed to investigate how closely several different networks can be modeled by the M/D/1 queue. The network parameters used in the simulations are summarized in Table 4. The message length (ML) was held constant at 10 ms for all of the simulations. Each agent attempted to broadcast a message at a periodic rate of once per second (e.g. at a constant sampling rate that would be typical of a network control system). The number of agents was varied between 1 and 80 to vary the channel utilization ρ. The backoff scheme employed a uniform distribution proportional to message length, and three variations were tested: 0-ML, 0-2.5ML, and 0-5ML. Approximately 10^4 messages were used in each simulation run. Two different types of collision detection were tested: $a = 0$ and $a = 0.1$ML. The case $a = 0$ assumes "perfect" collision detection, which is not really feasible. The case $a = 0.1$ML assumes that if one agent is scheduled to start a message within a seconds of the previous message there is a fatal collision - both messages are lost. Plots of the Monte Carlo simulation results appear in Figures 28 -57. Each data point

message length (ML)	0.01 sec
message frequency	1 Hz
number of agents	1-45
backoff scheme	uniform distribution proportional to message length 0-ML, 0-2.5ML, 0-5ML
number of messages	10^4
collision detection	$a = 0$ perfect (not really feasible) $a = 0.1$ML

Table 4: Monte Carlo Simulation Network Parameters

represents approximately 10^4 simulated messages. Data is presented for each of the six test cases: ($a = 0$, backoff = 0-ML),($a = 0$, backoff = 0-2.5ML),($a = 0$, backoff = 0-5ML), ($a = 0.1$ML, backoff = 0-ML), ($a = 0.1$ML, backoff = 0-2.5ML), and ($a = 0.1$ML, backoff = 0-5ML). The first plot compares the normalized average delay from the Monte Carlo simulation to the ideal M/D/1 queue. The second plot shows the variance of the delay (not normalized). The third plot displays the statistics for the transmission retries. The fourth plot shows the statistics for the channel access time. The fifth plot contains the fatal collision statistics. For the ideal case $a = 0$ there will never be any fatal collisions because of the perfect (unrealistic) collision detection.

Overall, the Monte Carlo simulations agree with the M/D/1 approximation. However, the accuracy of the agreement depends on the simulation parameters as well as the range of utilization ρ. For example, the test case ($a = 0$, backoff = 0-ML) agrees well for $\rho < 0.5$. For another test case ($a = 0.1ML$, backoff = 0-ML), the M/D/1 queue approximation is accurate for $0.1 < \rho < 0.7$. In general, the longer the backoff scheme (e.g. 0-5ML), the larger the delay at higher utilizations. The differences in the M/D/1 queue approximation and the Monte Carlo simulations may be traced to the channel access statistics. The closer these are approximated by an independent Poisson source, the more accurate the M/D/1 approximation.

The next section describes a Linear Matrix Inequality approach for stable control of multiple cooperative robotic vehicles.

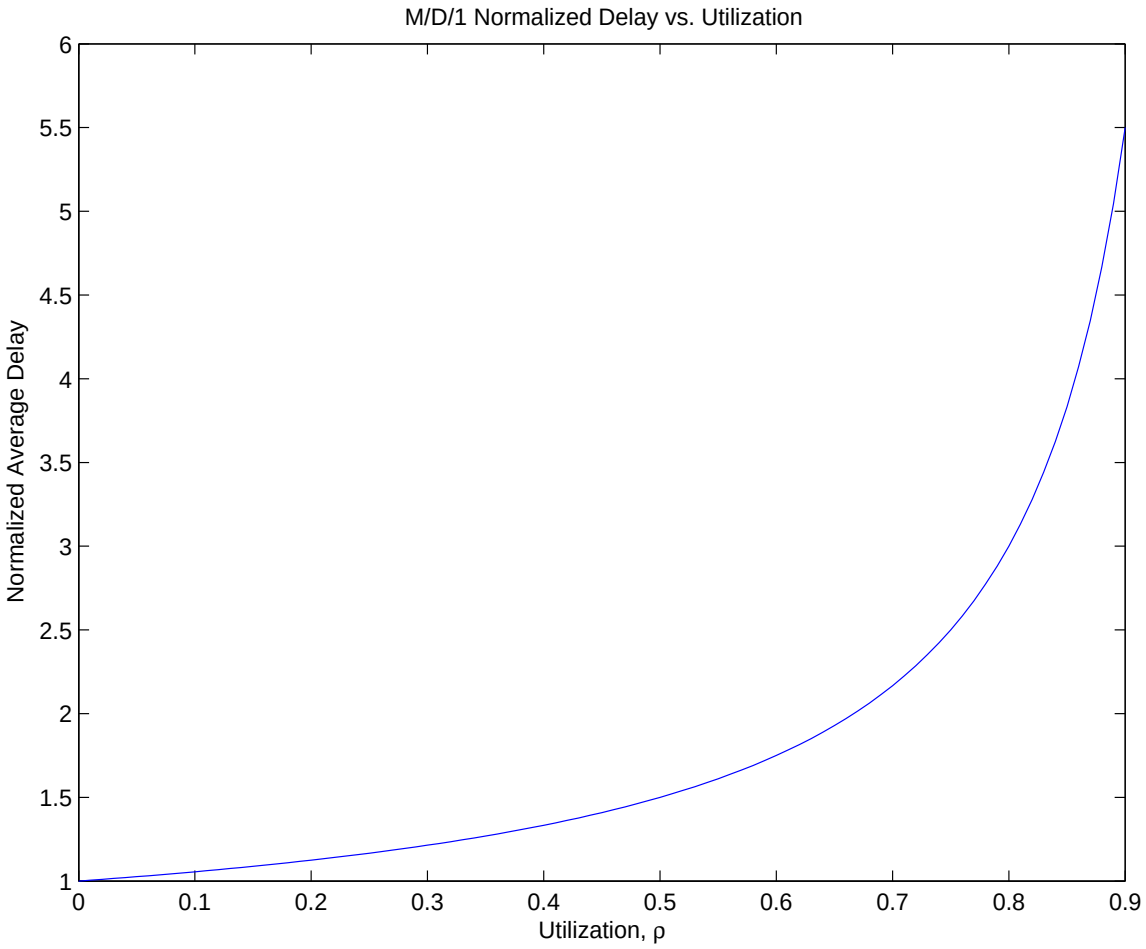

Figure 27: M/D/1 Normalized Average Delay versus Utilization

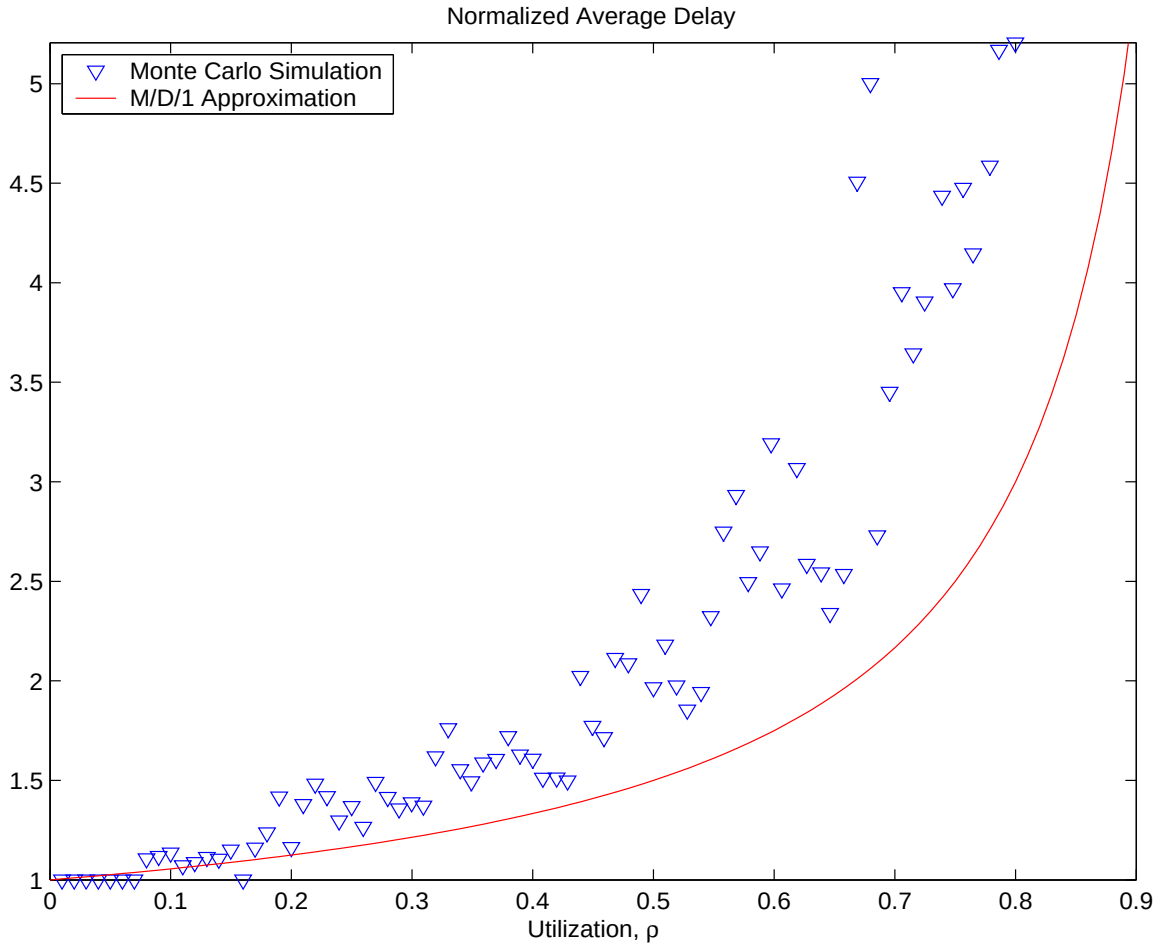

Figure 28: Monte Carlo Simulation: Delay Statistics, $a = 0$, Backoff 0-ML

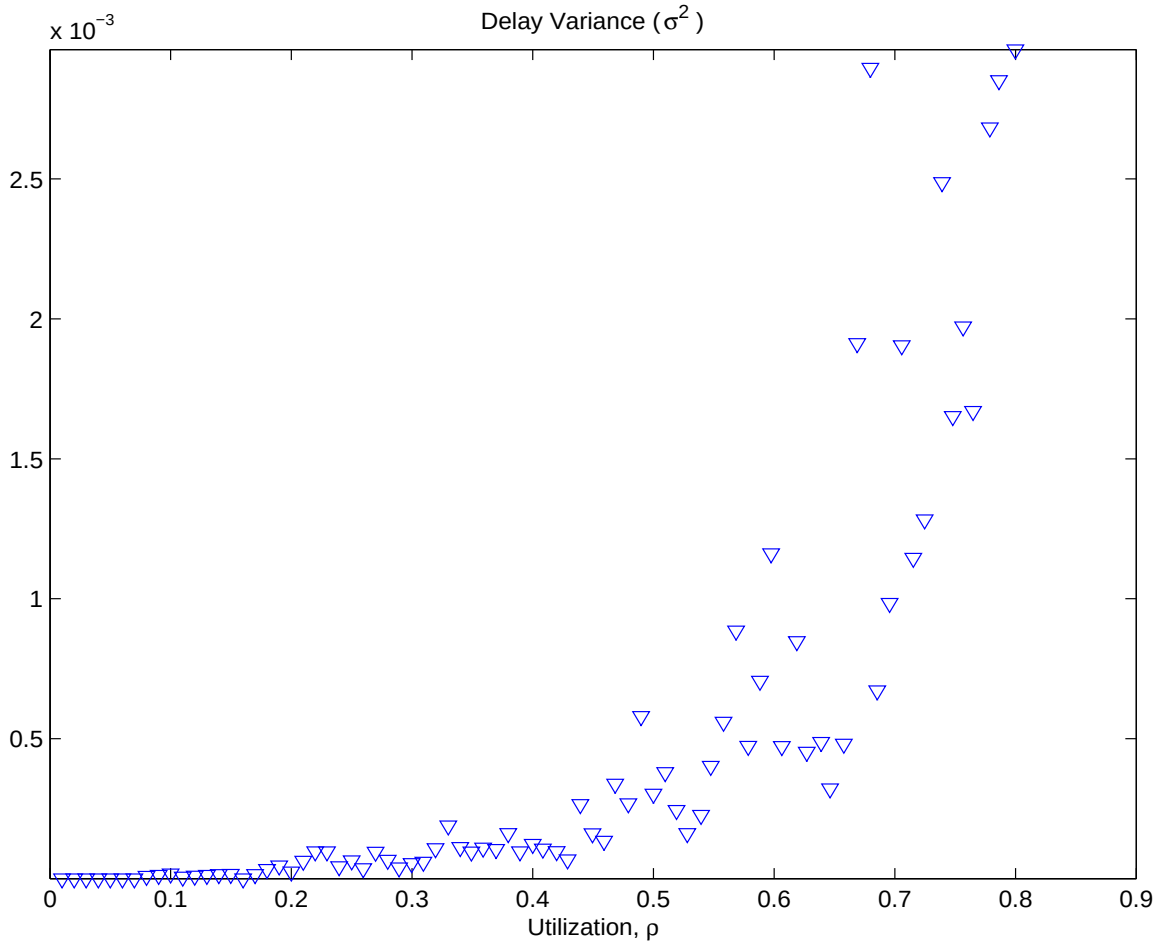

Figure 29: Monte Carlo Simulation: Delay Statistics, $a = 0$, Backoff 0-ML

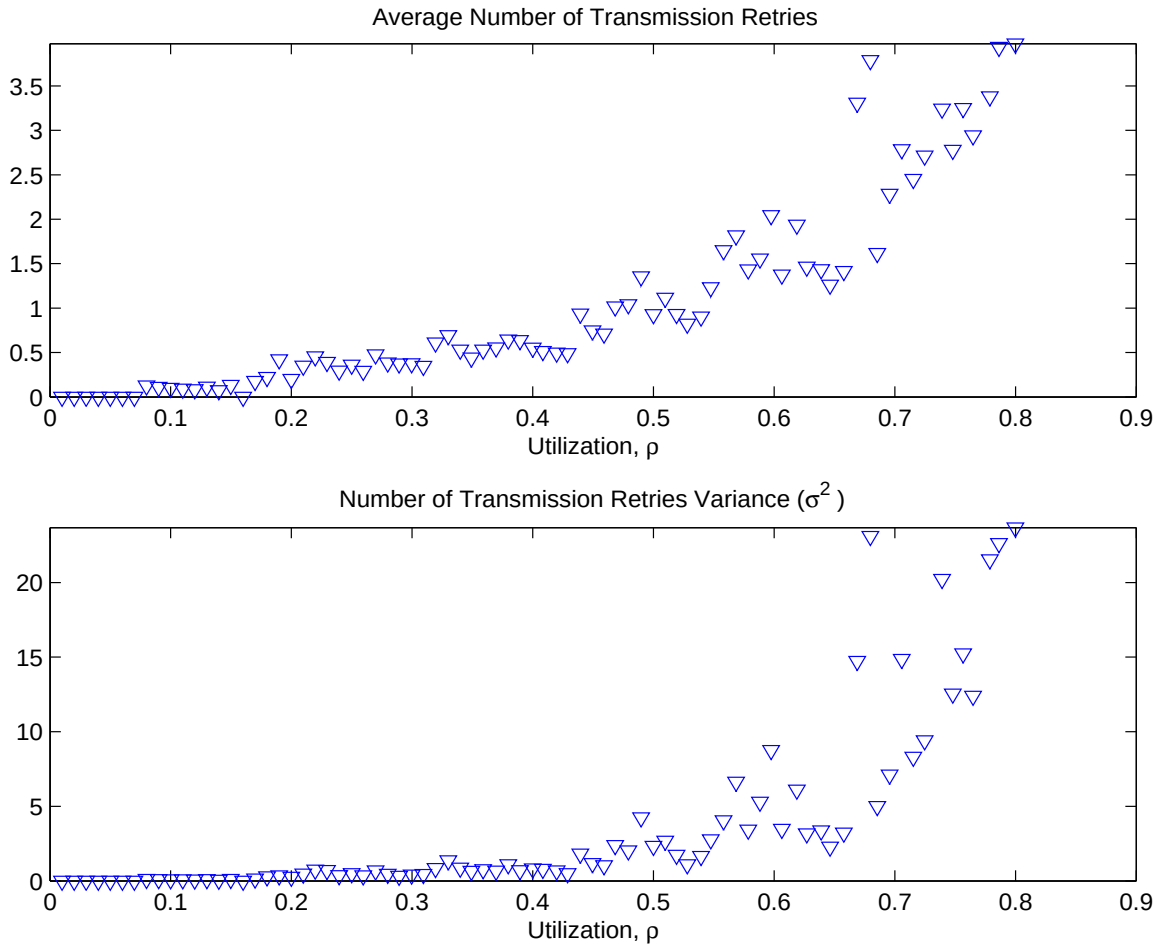

Figure 30: Monte Carlo Simulation: Transmission Retry Statistics, $a = 0$, Backoff 0-ML

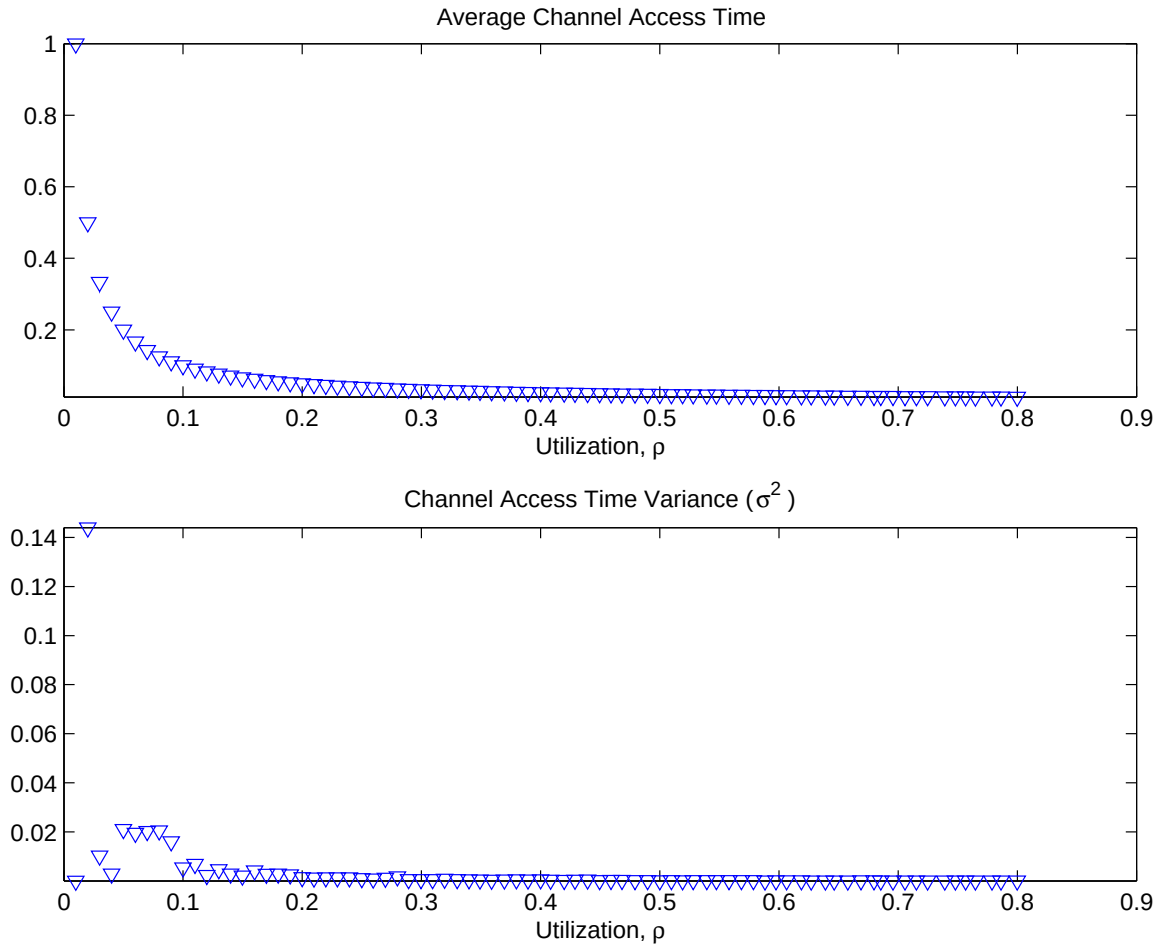

Figure 31: Monte Carlo Simulation: Channel Access Time Statistics, $a = 0$, Backoff 0-ML

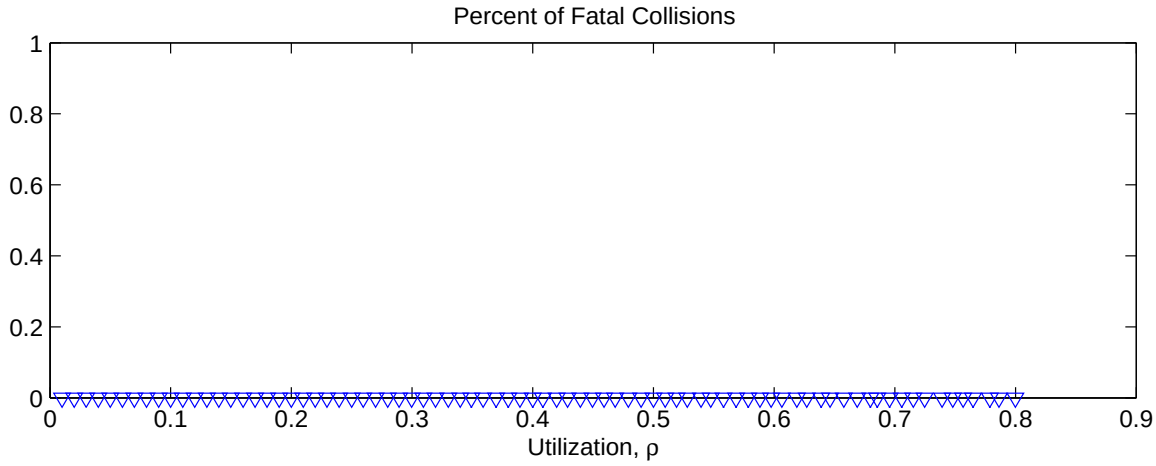

Figure 32: Monte Carlo Simulation: Fatal Collision Statistics, $a = 0$, Backoff 0-ML

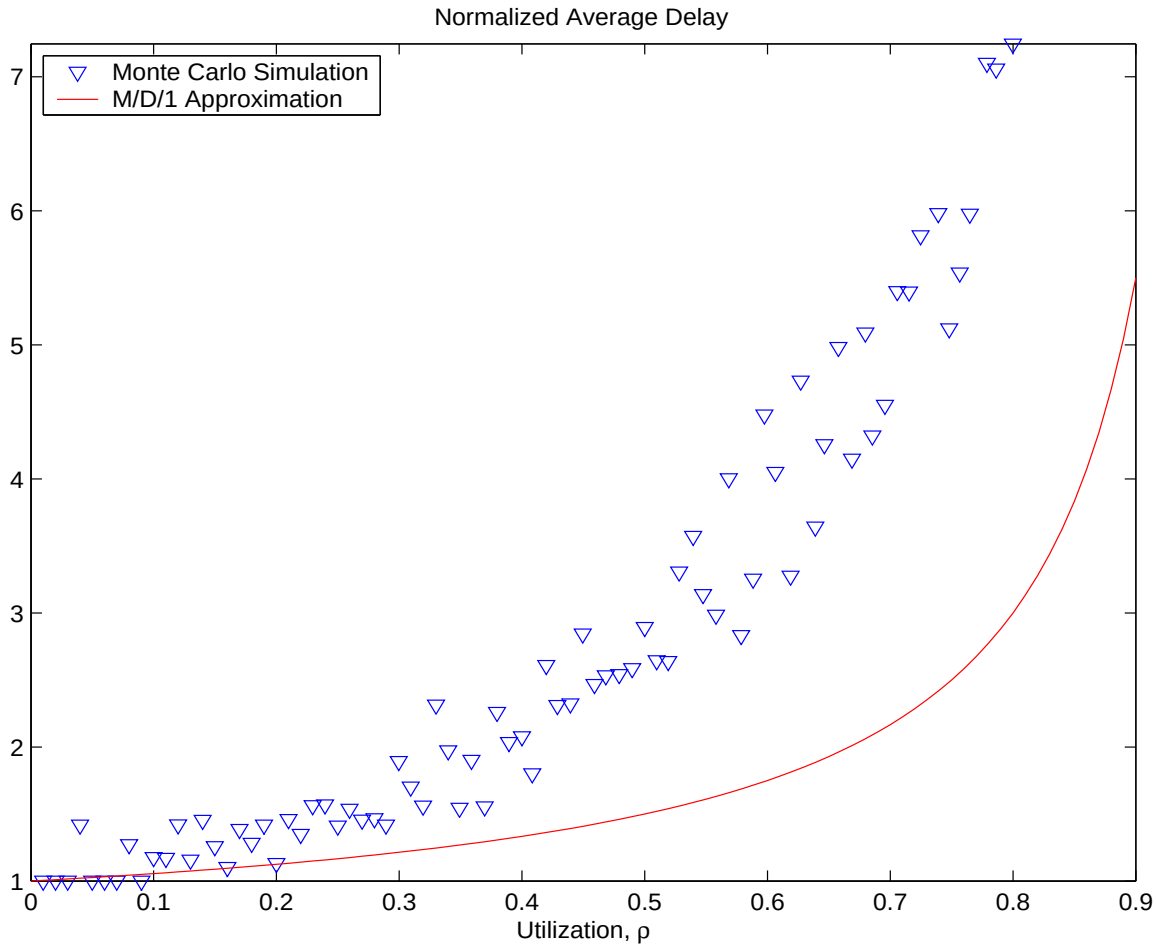

Figure 33: Monte Carlo Simulation: Delay Statistics, $a = 0$, Backoff 0-2.5ML

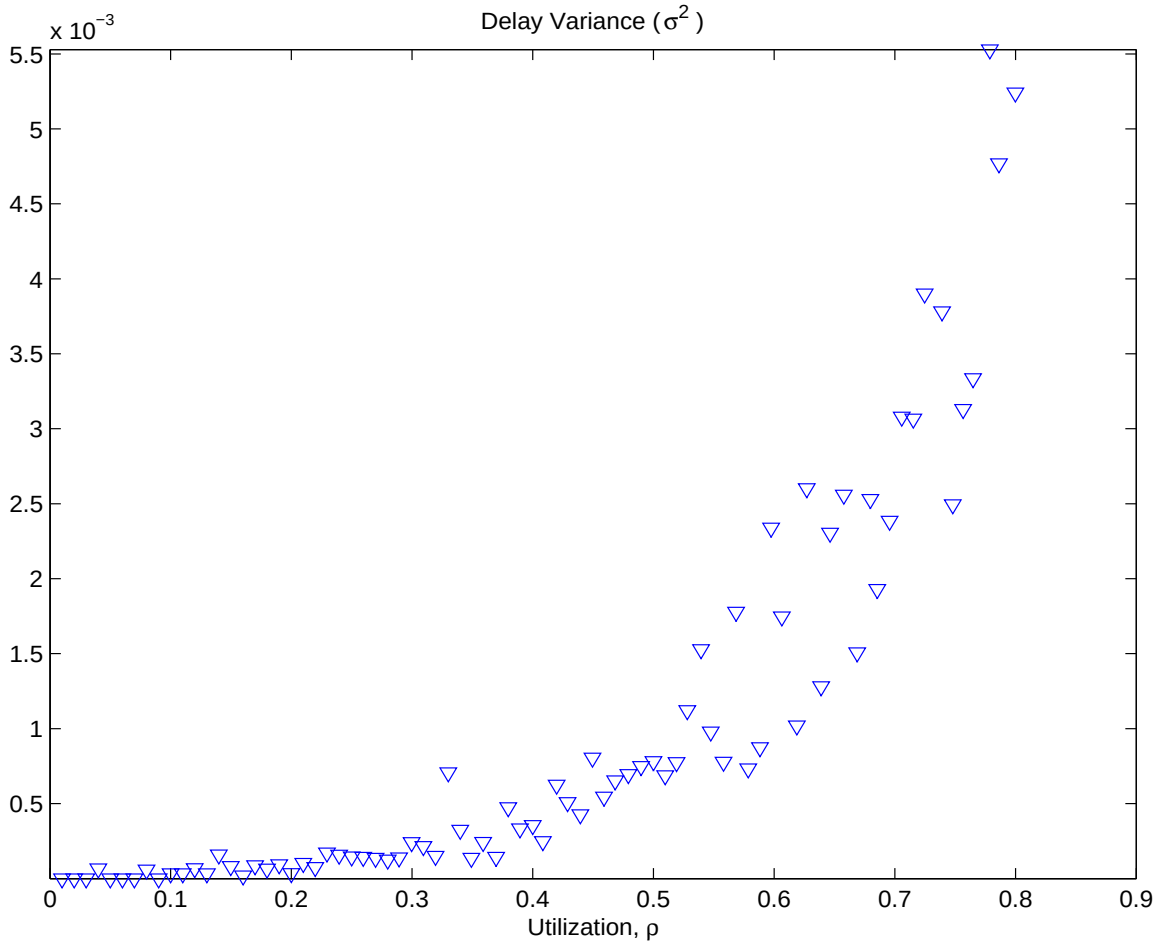

Figure 34: Monte Carlo Simulation: Delay Statistics, $a = 0$, Backoff 0-2.5ML

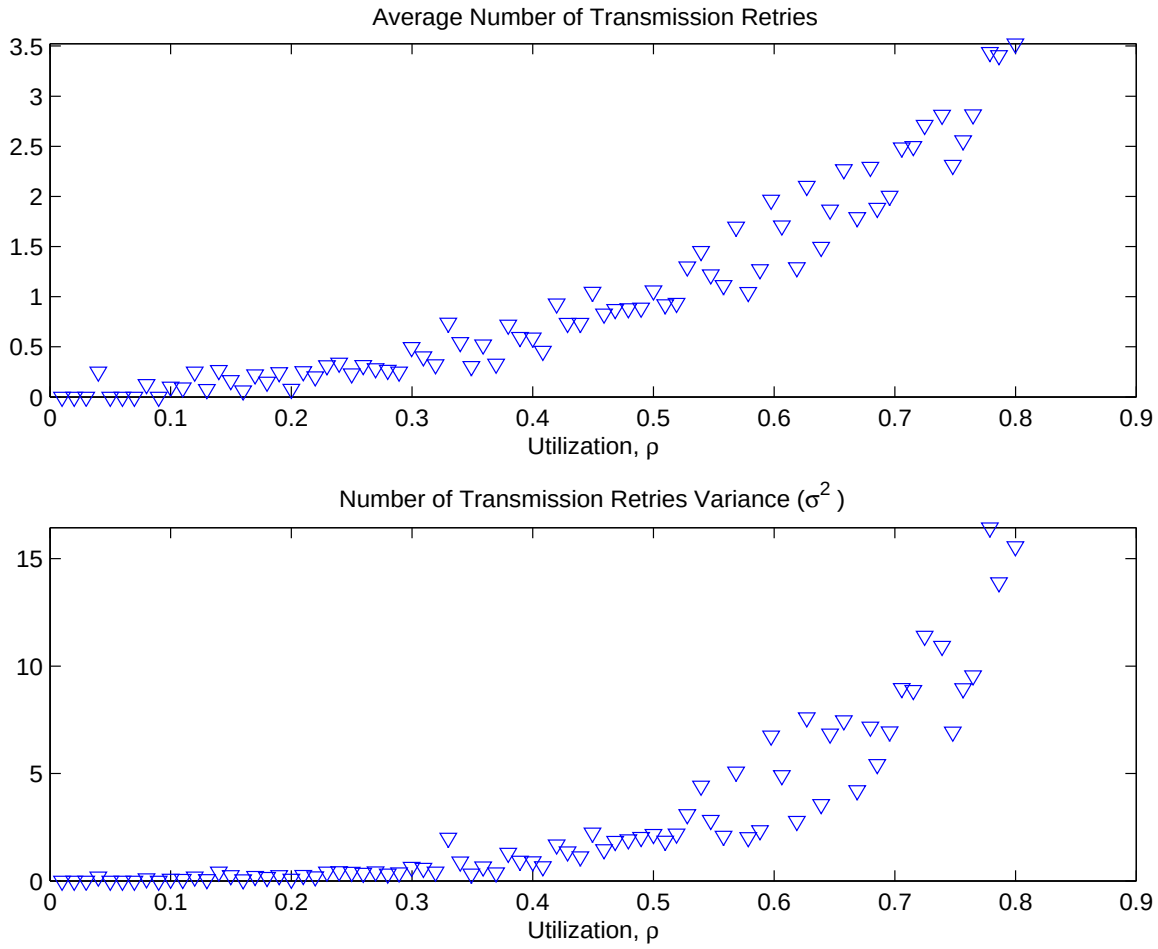

Figure 35: Monte Carlo Simulation: Transmission Retry Statistics, $a = 0$, Backoff 0-2.5ML

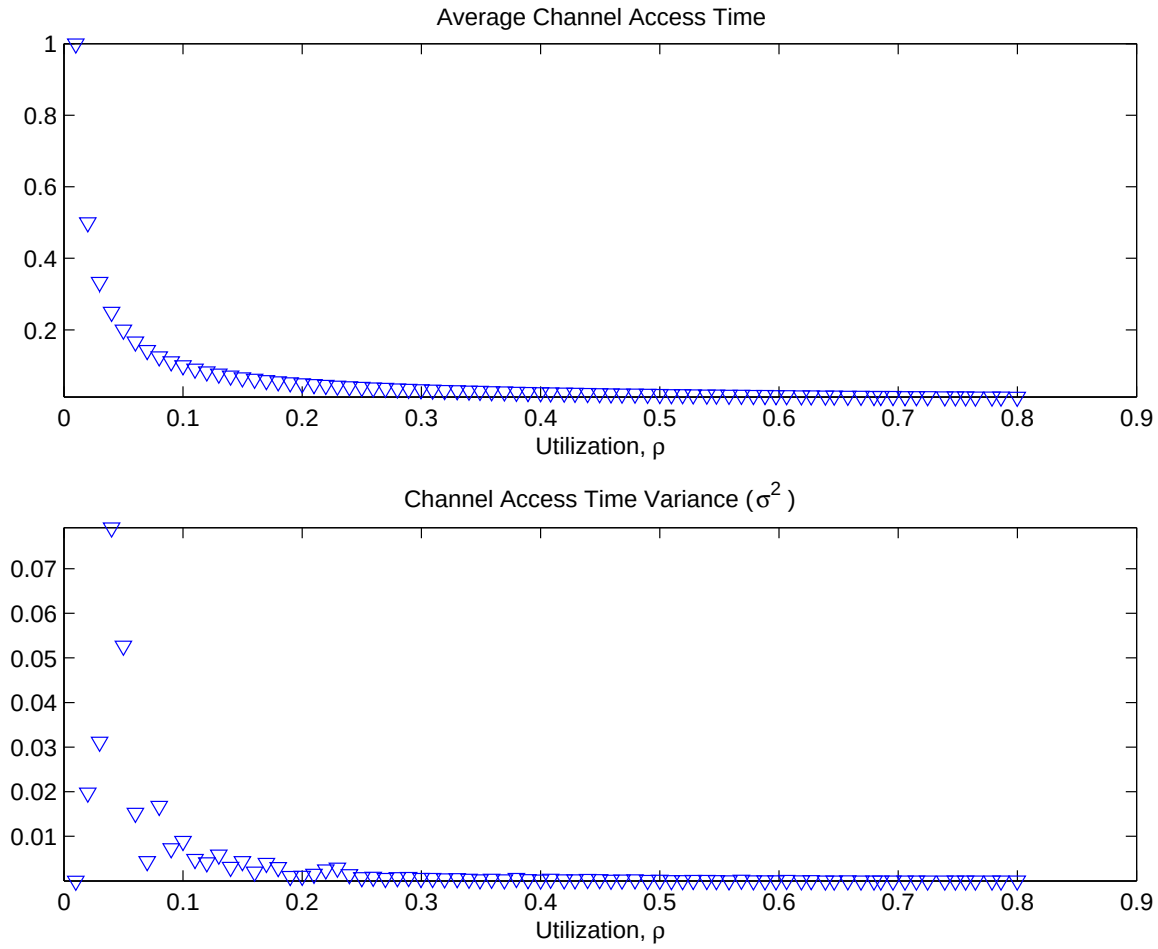

Figure 36: Monte Carlo Simulation: Channel Access Time Statistics, $a = 0$, Backoff 0-2.5ML

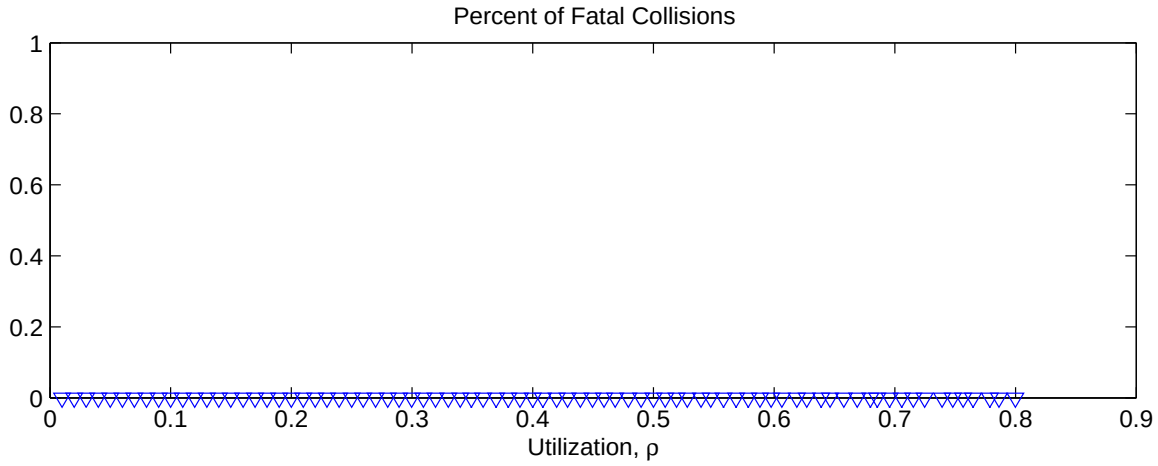

Figure 37: Monte Carlo Simulation: Fatal Collision Statistics, $a = 0$, Backoff 0-2.5ML

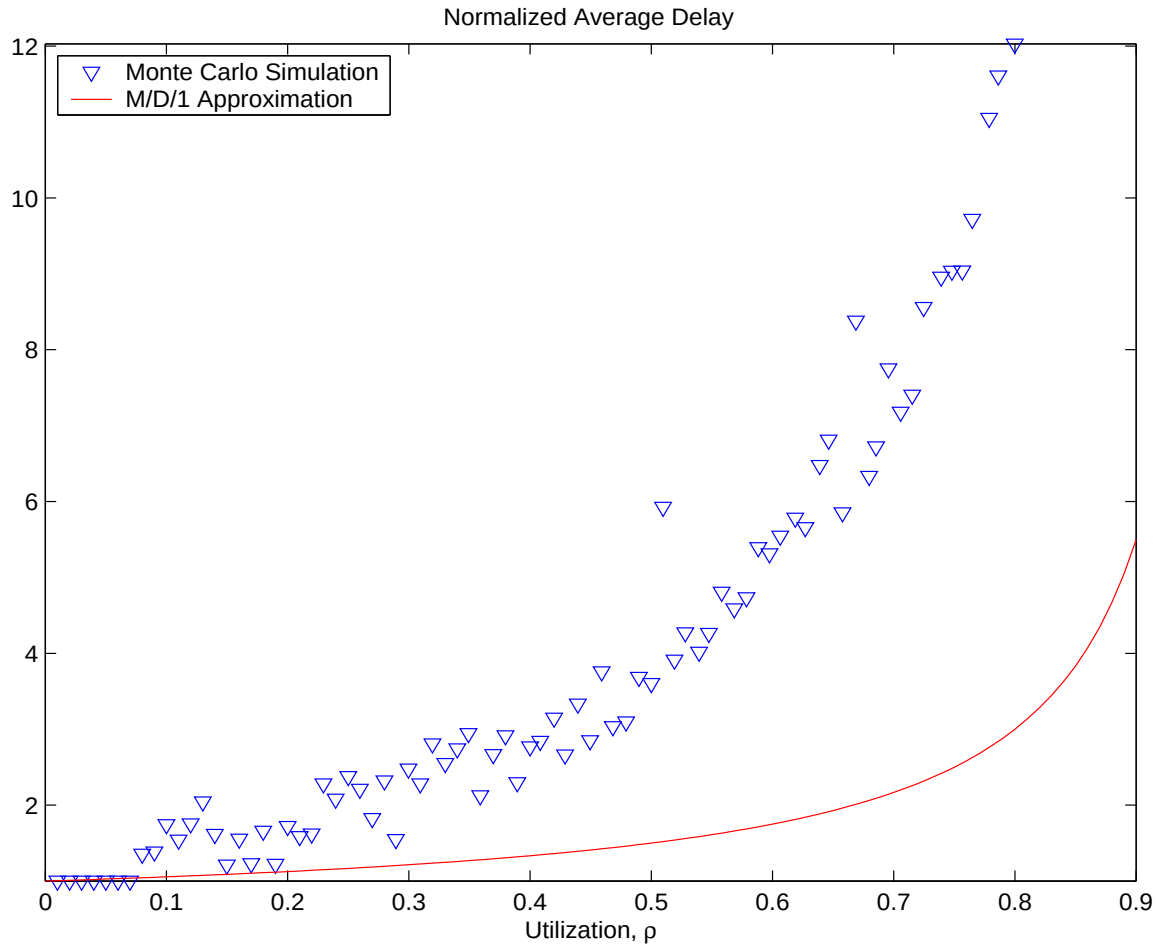

Figure 38: Monte Carlo Simulation: Delay Statistics, $a = 0$, Backoff 0-5ML

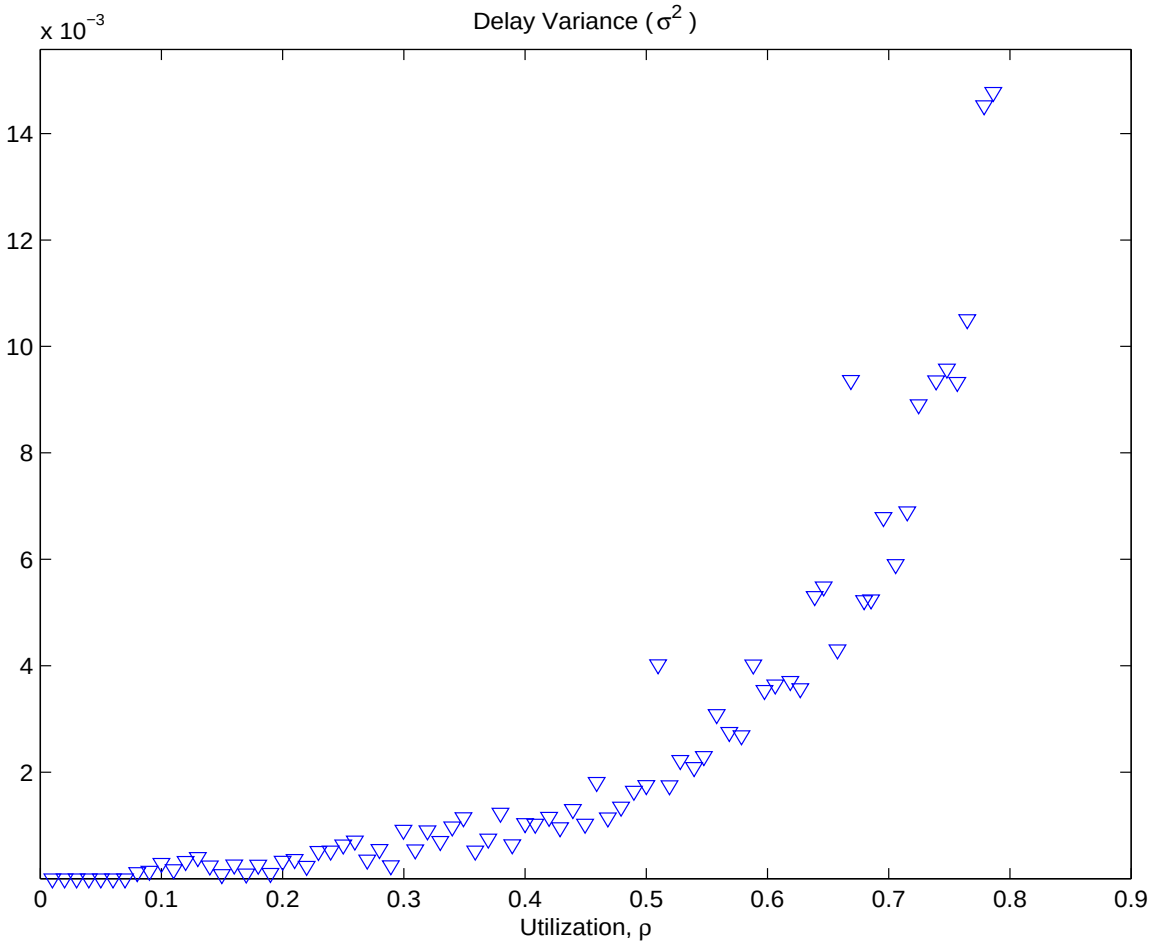

Figure 39: Monte Carlo Simulation: Delay Statistics, $a = 0$, Backoff 0-5ML

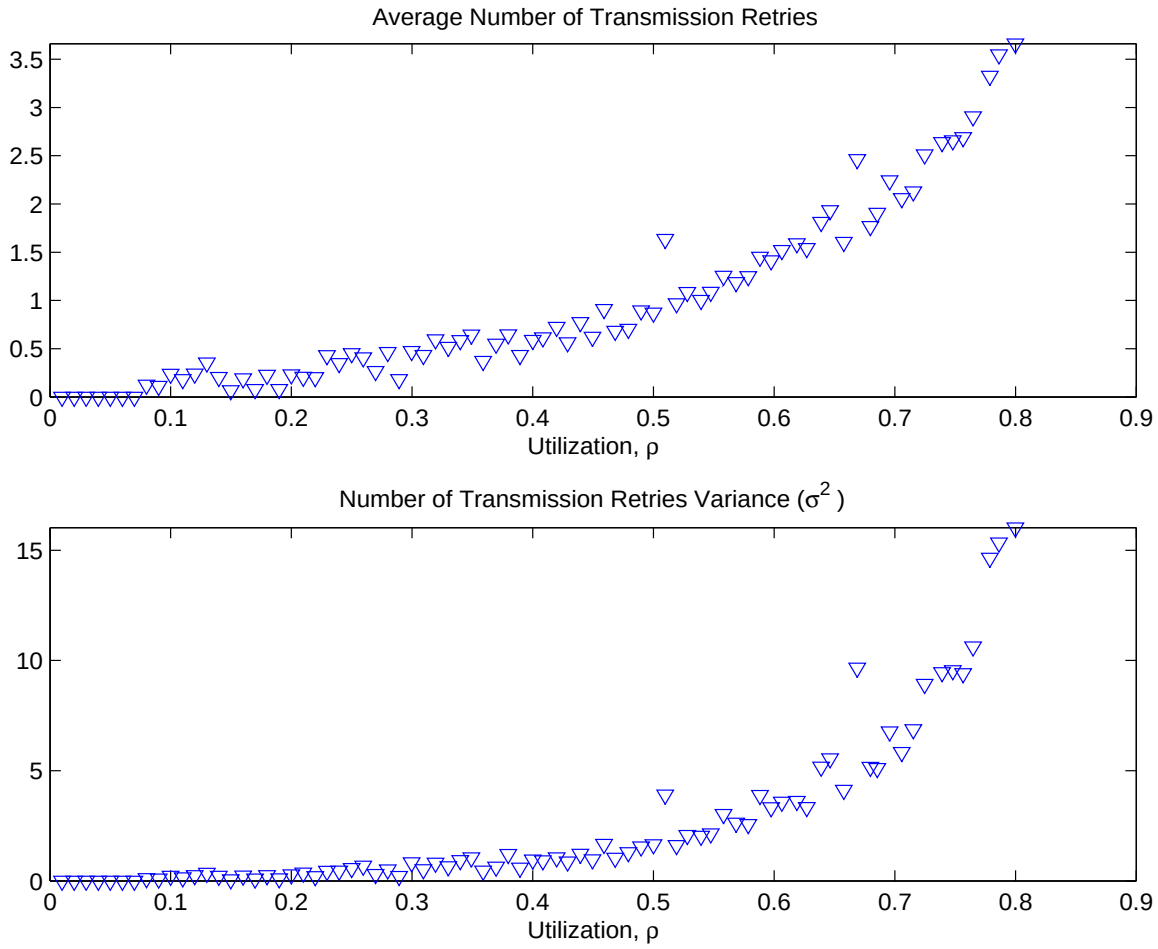

Figure 40: Monte Carlo Simulation: Transmission Retry Statistics, $a = 0$, Backoff 0-5ML

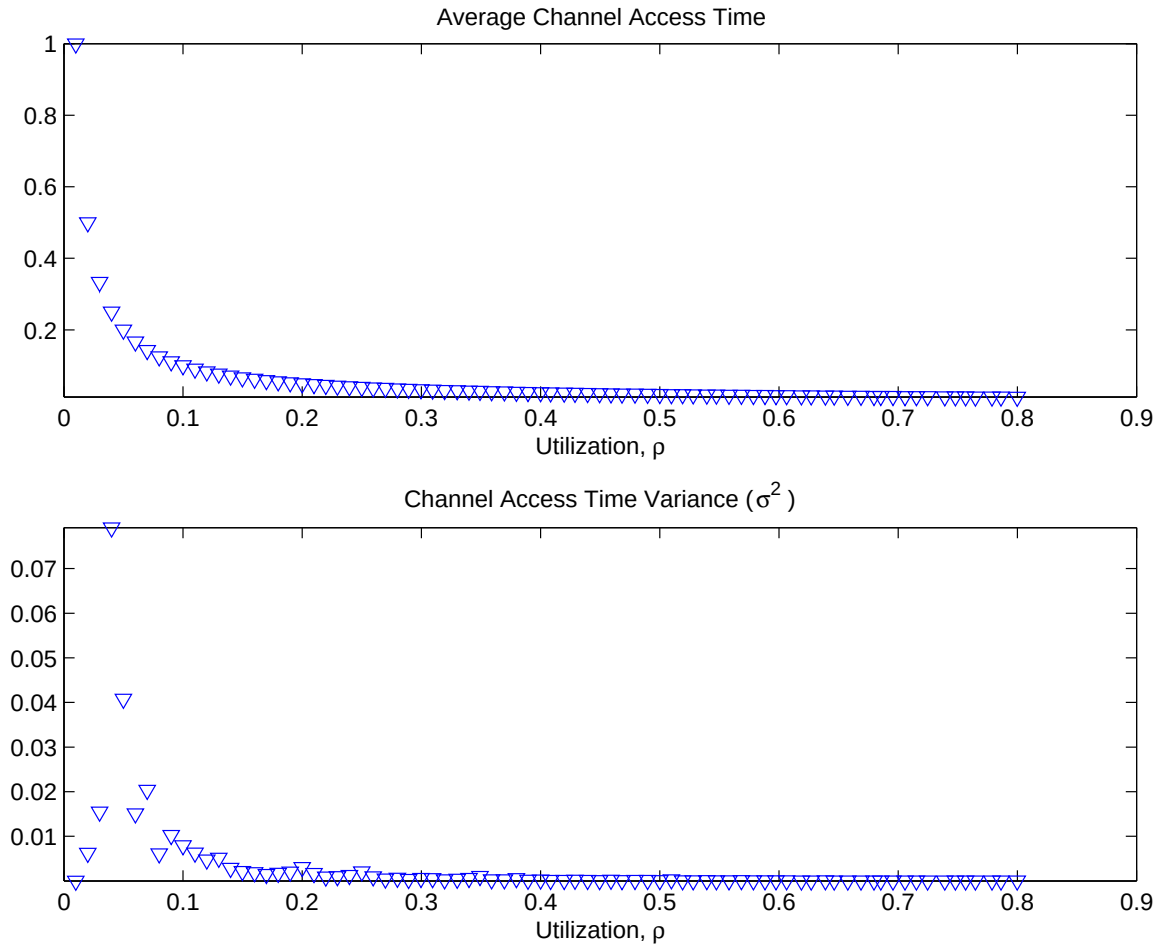

Figure 41: Monte Carlo Simulation: Channel Access Time Statistics, $a = 0$, Backoff 0-5ML

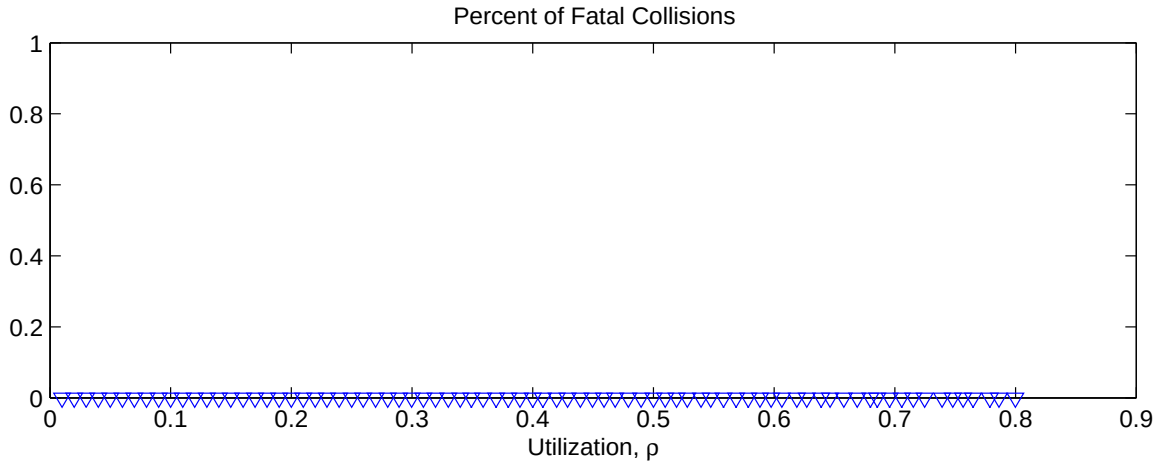

Figure 42: Monte Carlo Simulation: Fatal Collision Statistics, $a = 0$, Backoff 0-5ML

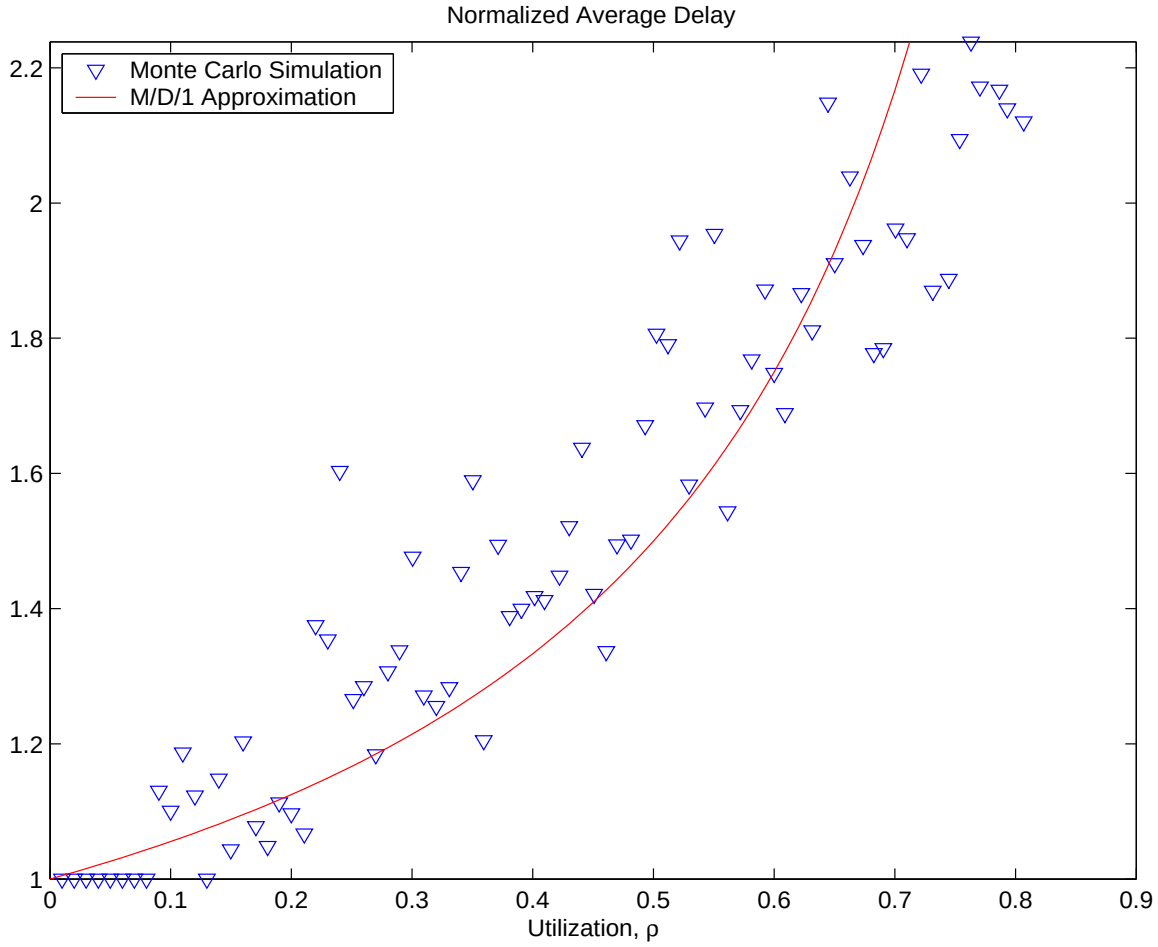

Figure 43: Monte Carlo Simulation: Delay Statistics, $a = 0.1$ML, Backoff 0-ML

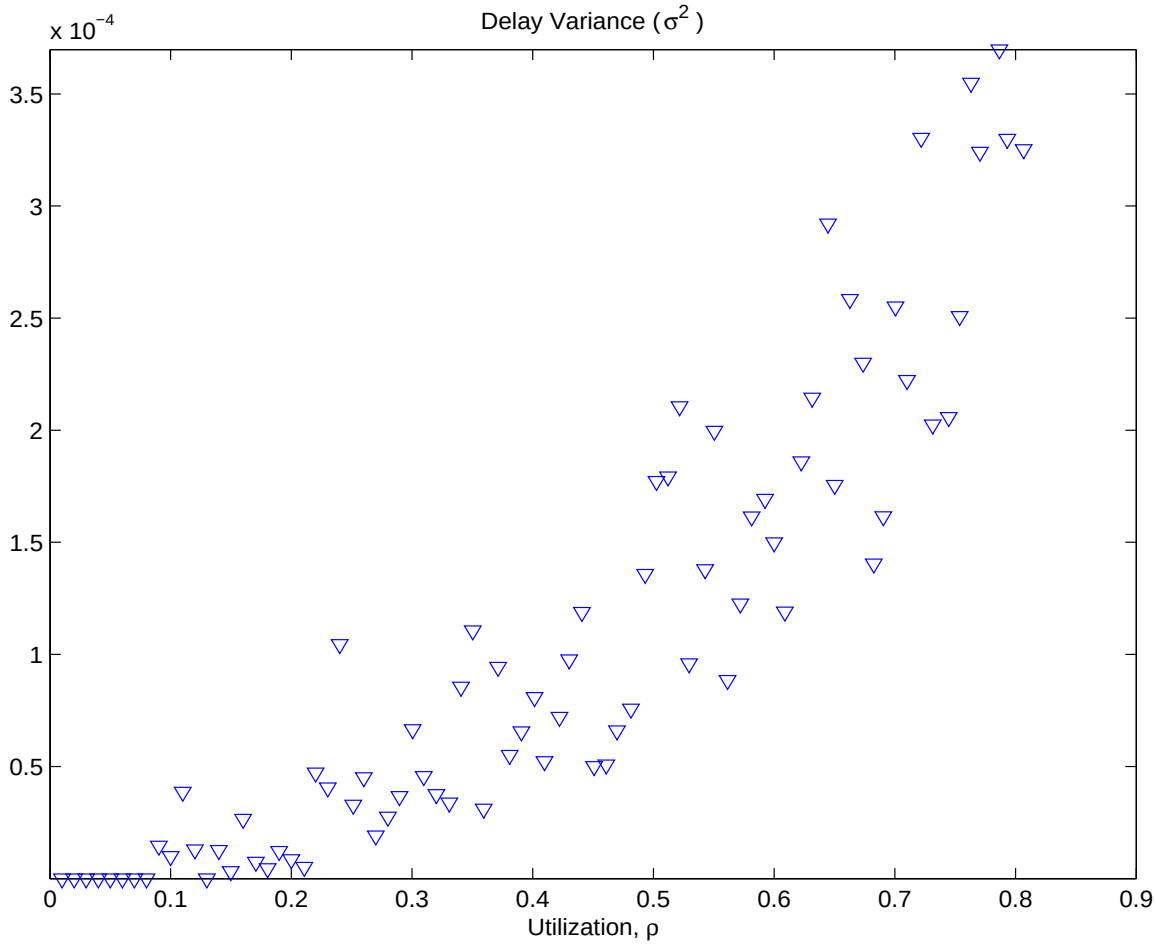

Figure 44: Monte Carlo Simulation: Delay Statistics, $a = 0.1$ML, Backoff 0-ML

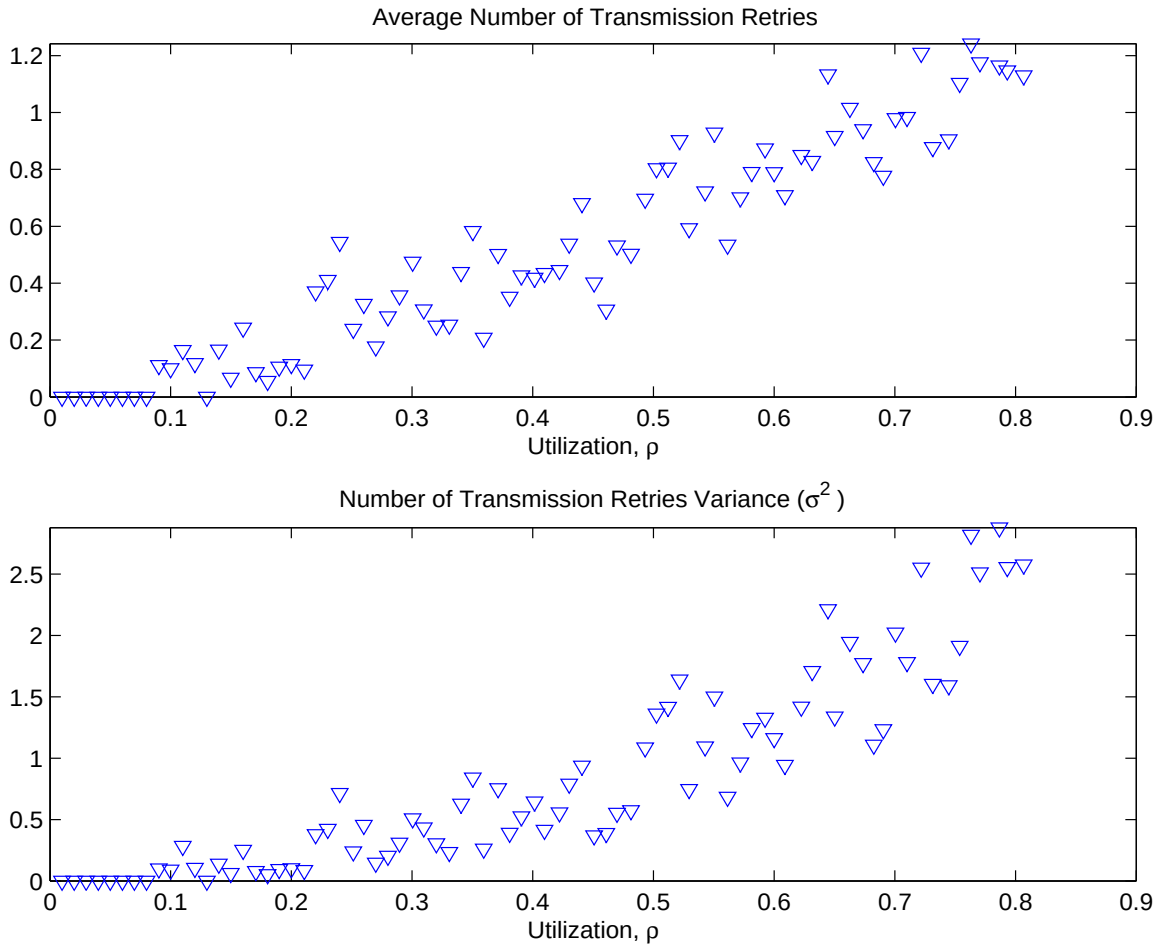

Figure 45: Monte Carlo Simulation: Transmission Retry Statistics, $a = 0.1$ML, Back-off 0-ML

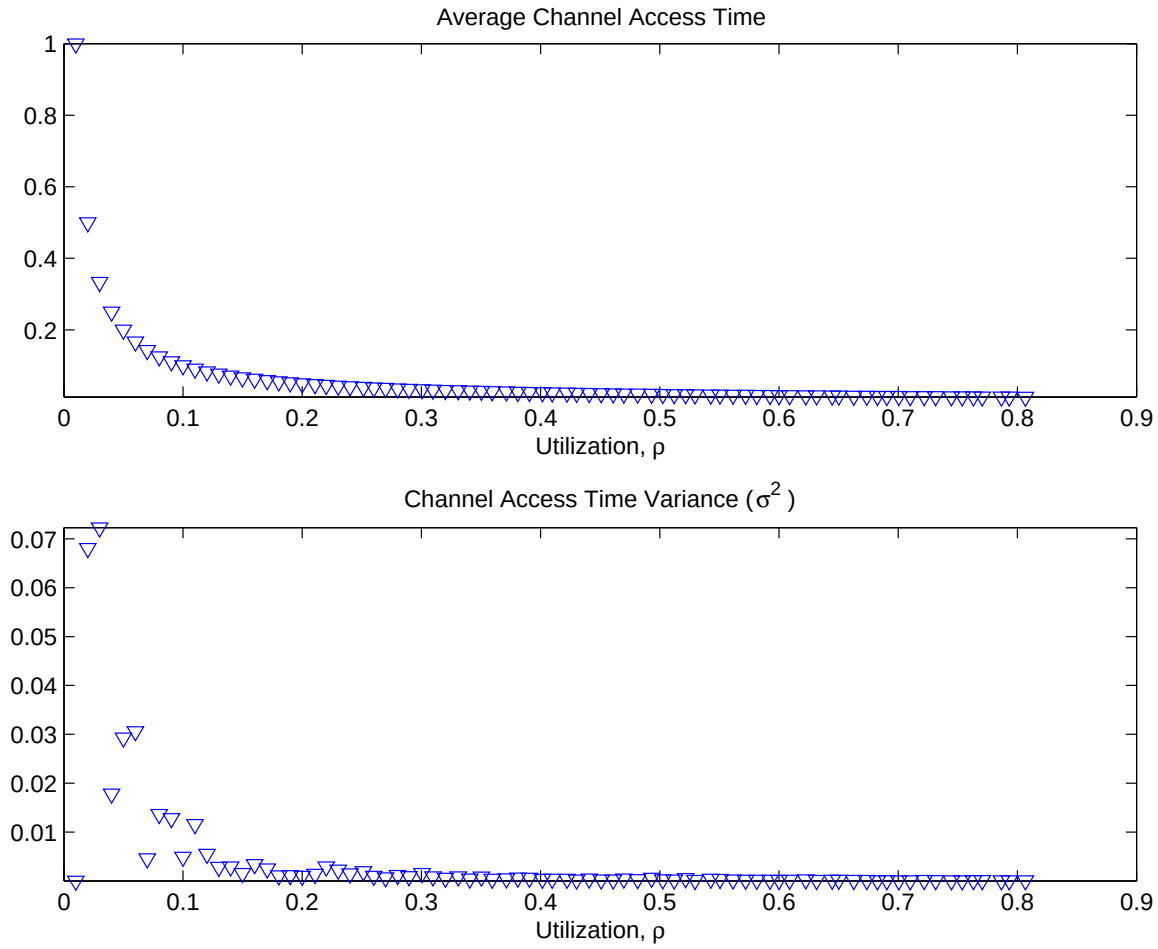

Figure 46: Monte Carlo Simulation: Channel Access Time Statistics, $a = 0.1$ML, Backoff 0-ML

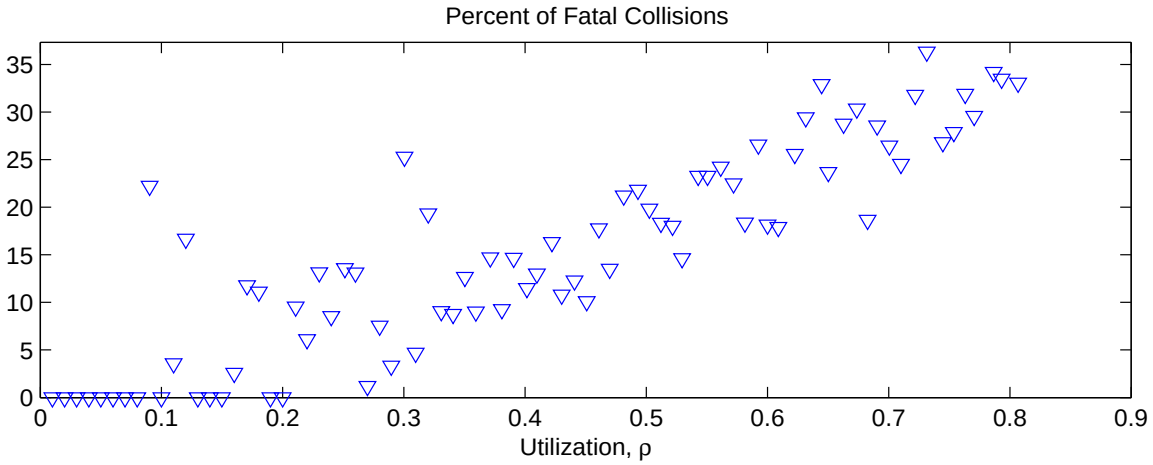

Figure 47: Monte Carlo Simulation: Fatal Collision Statistics, $a = 0.1$ML, Backoff 0-ML

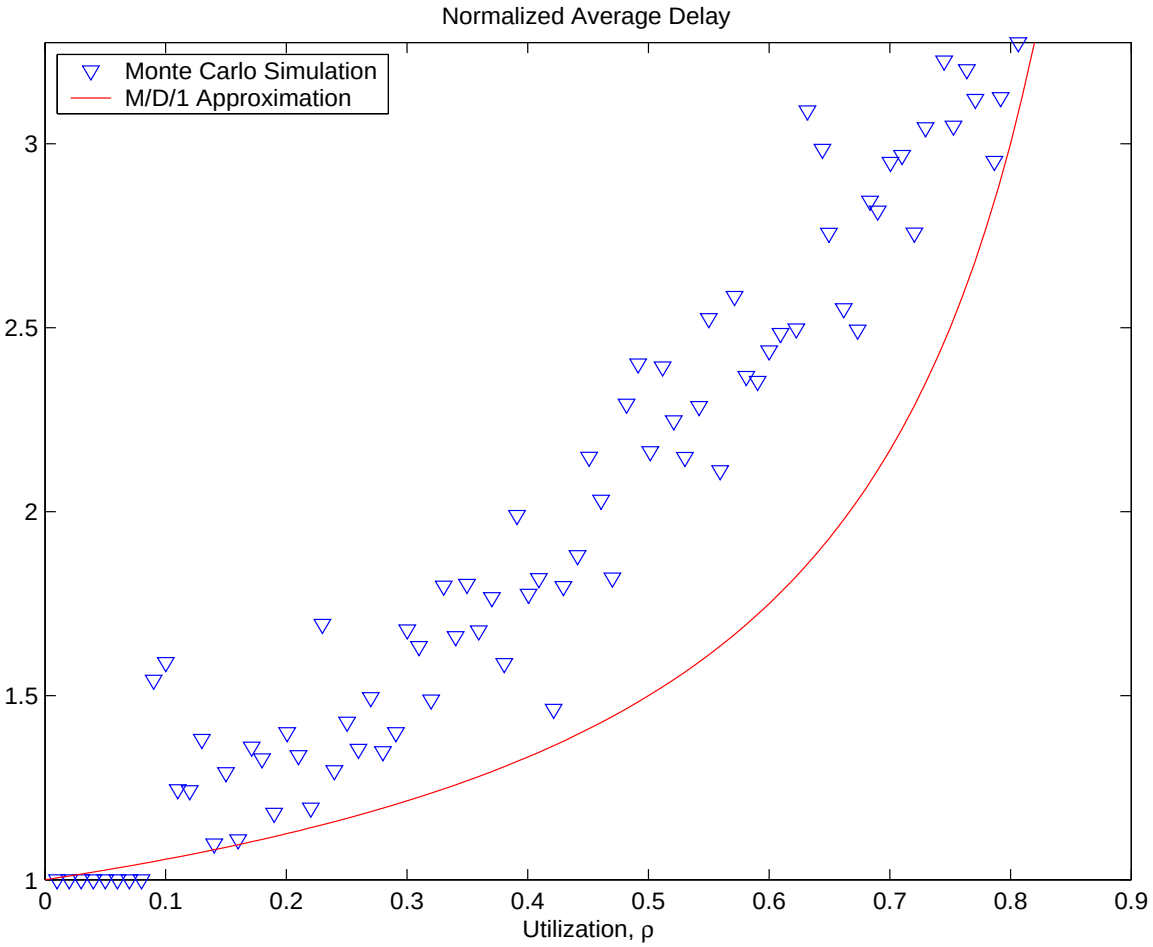

Figure 48: Monte Carlo Simulation: Delay Statistics, $a = 0.1$ML, Backoff 0-2.5ML

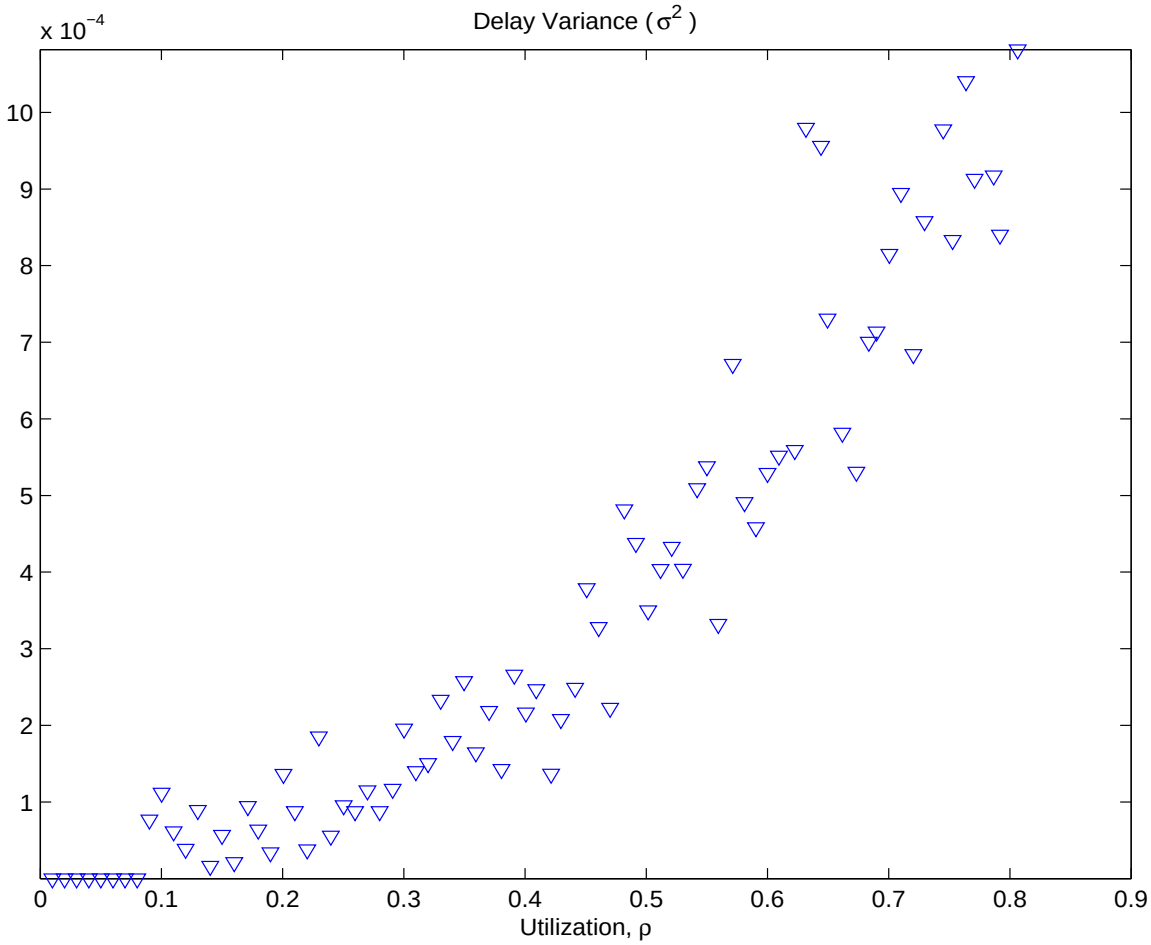

Figure 49: Monte Carlo Simulation: Delay Statistics, $a = 0.1$ML, Backoff 0-2.5ML

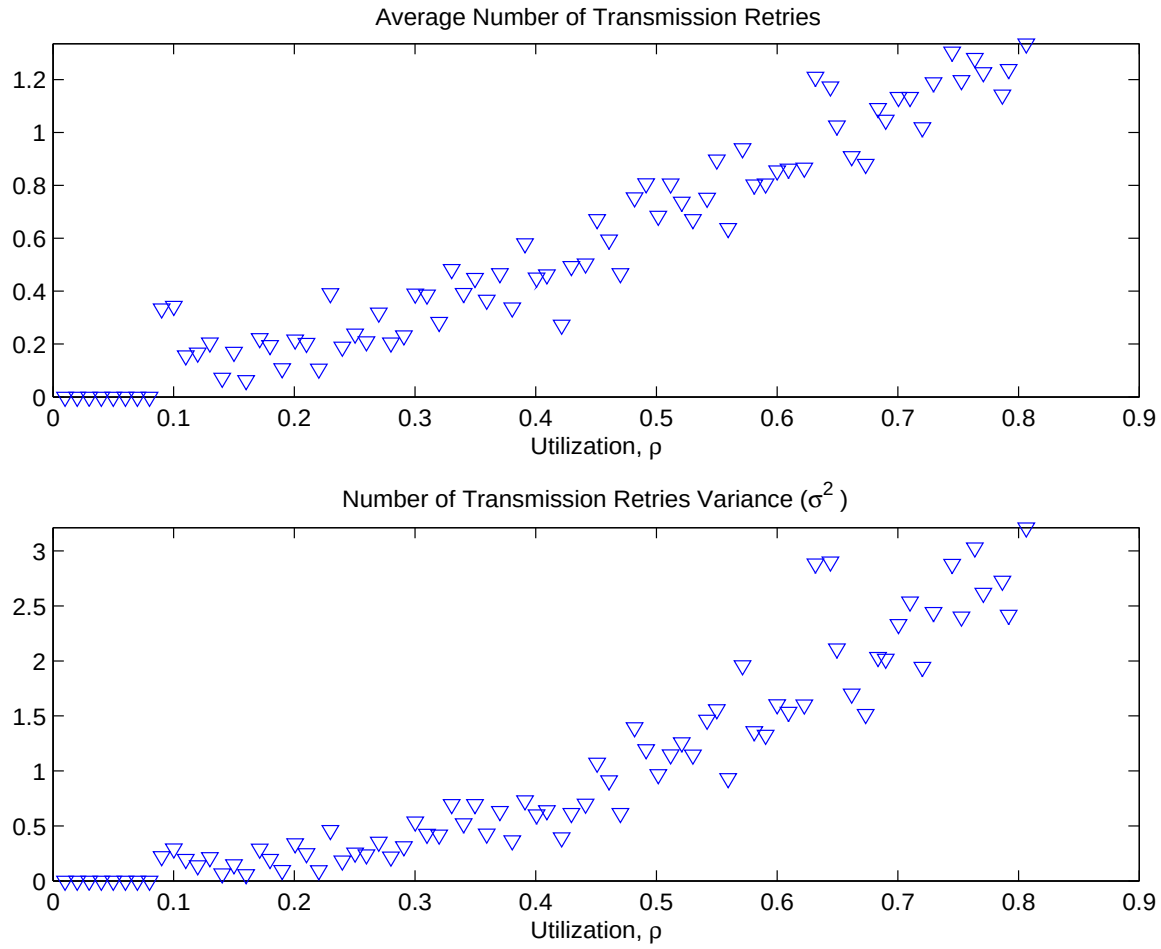

Figure 50: Monte Carlo Simulation: Transmission Retry Statistics, $a = 0.1$ML, Back-off 0-2.5ML

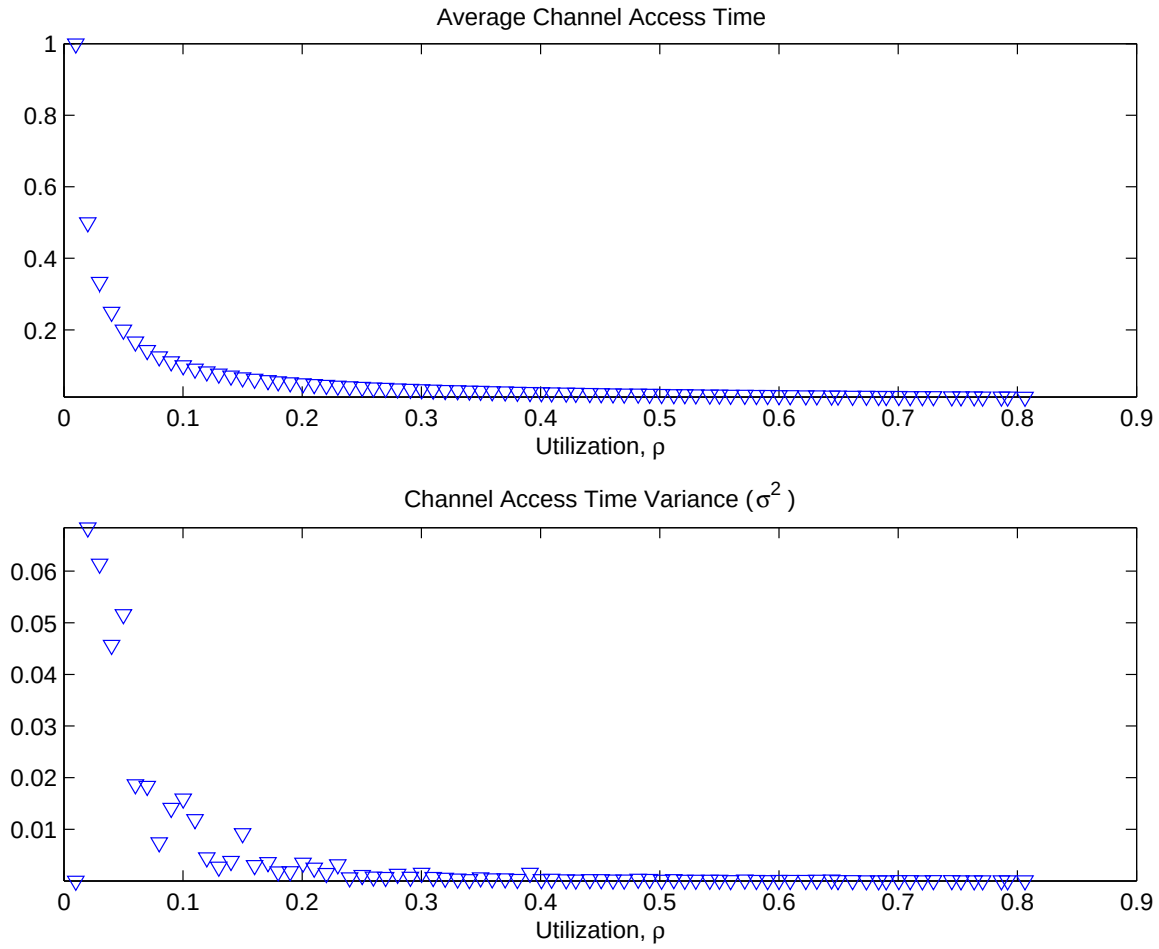

Figure 51: Monte Carlo Simulation: Channel Access Time Statistics, $a = 0.1$ML, Backoff 0-2.5ML

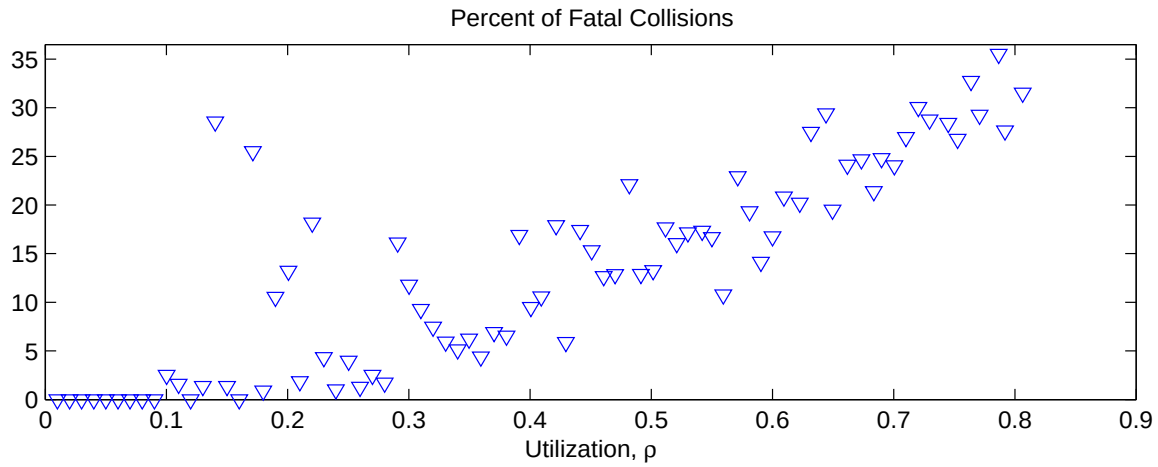

Figure 52: Monte Carlo Simulation: Fatal Collision Statistics, $a = 0.1$ML, Backoff 0-2.5ML

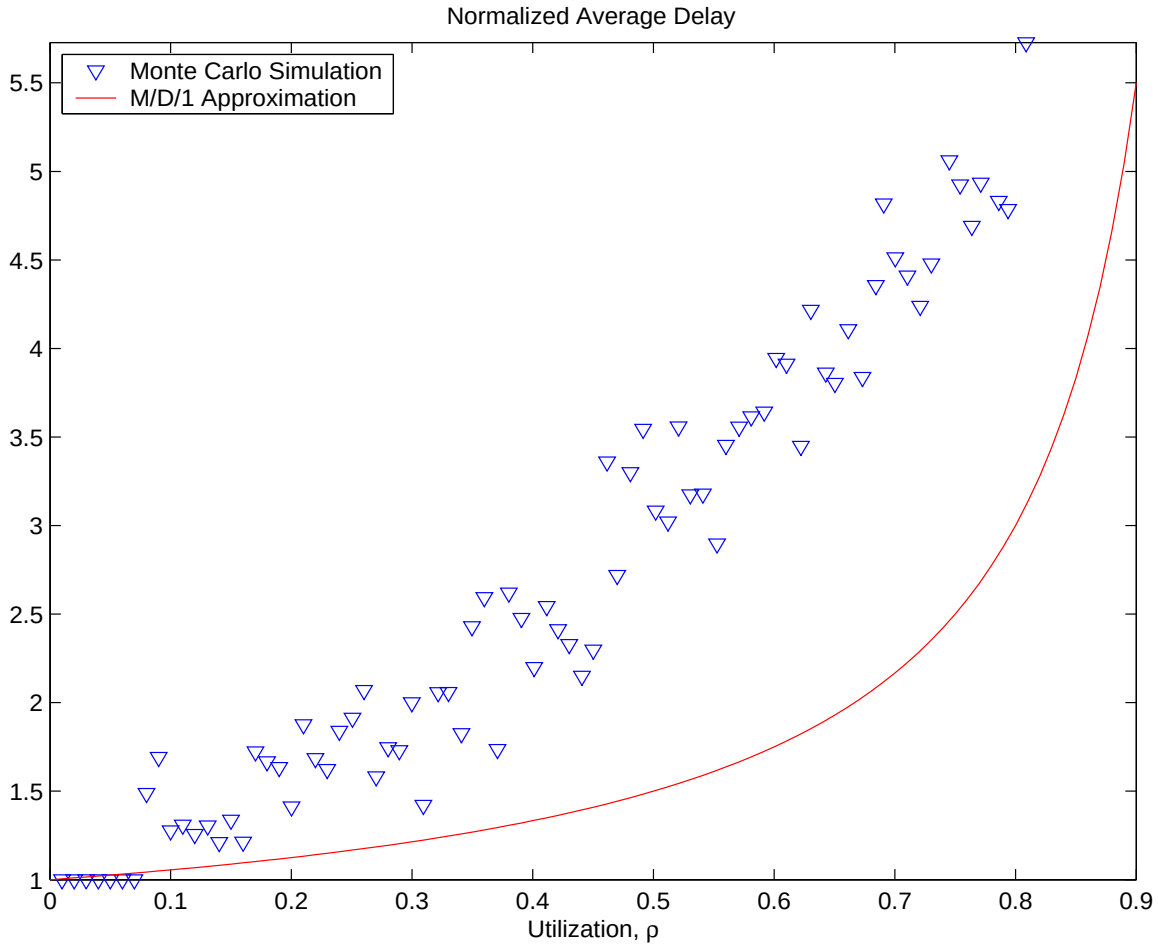

Figure 53: Monte Carlo Simulation: Delay Statistics, $a = 0.1$ML, Backoff 0-5ML

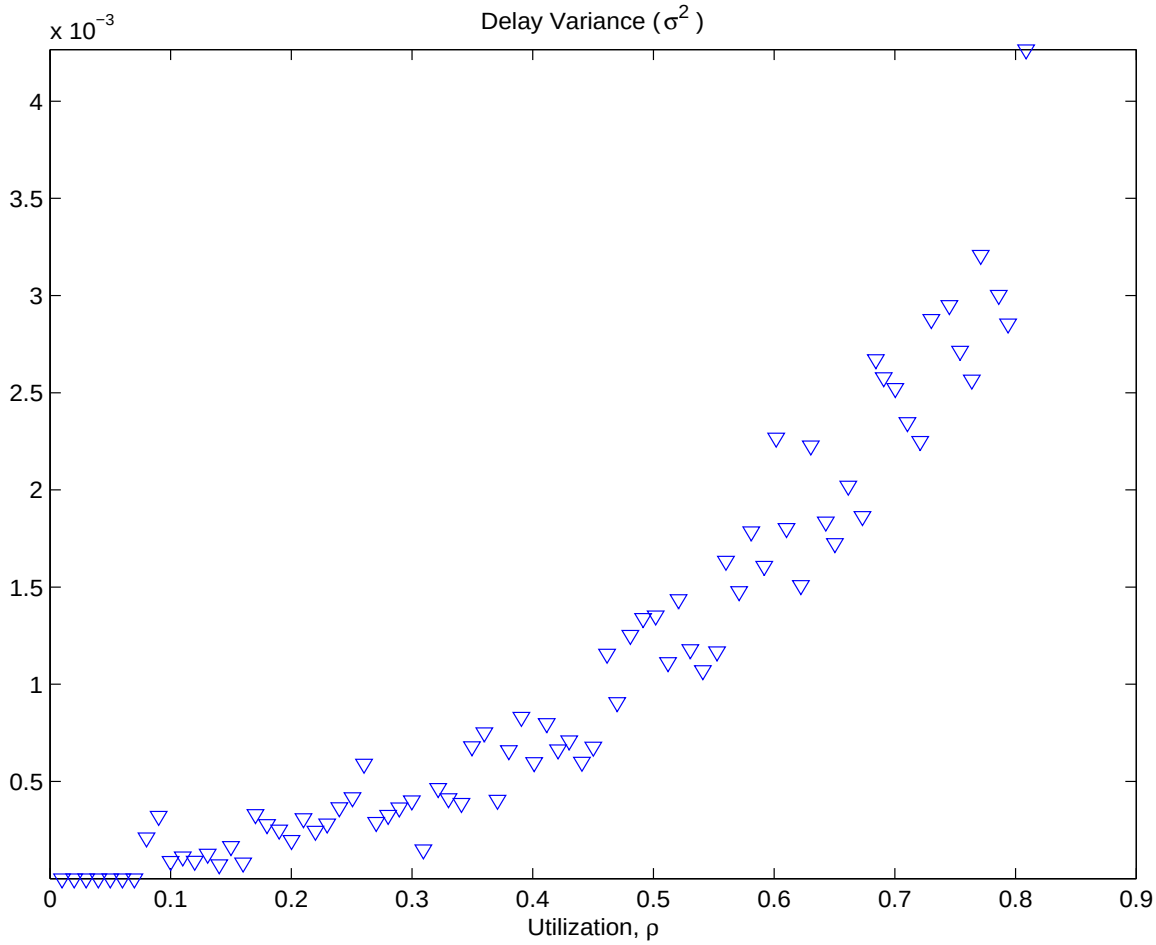

Figure 54: Monte Carlo Simulation: Delay Statistics, $a = 0.1$ML, Backoff 0-5ML

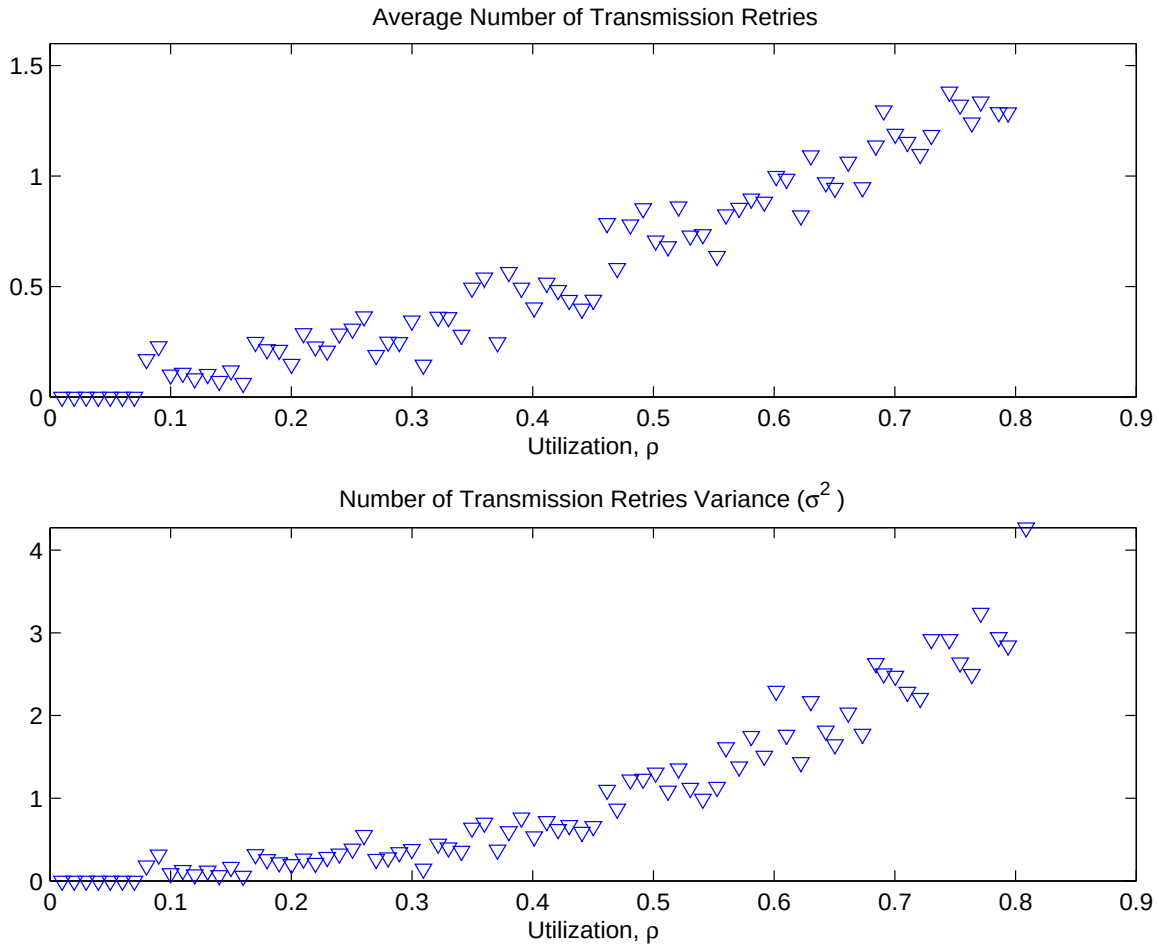

Figure 55: Monte Carlo Simulation: Transmission Retry Statistics, $a = 0.1$ML, Back-off 0-5ML

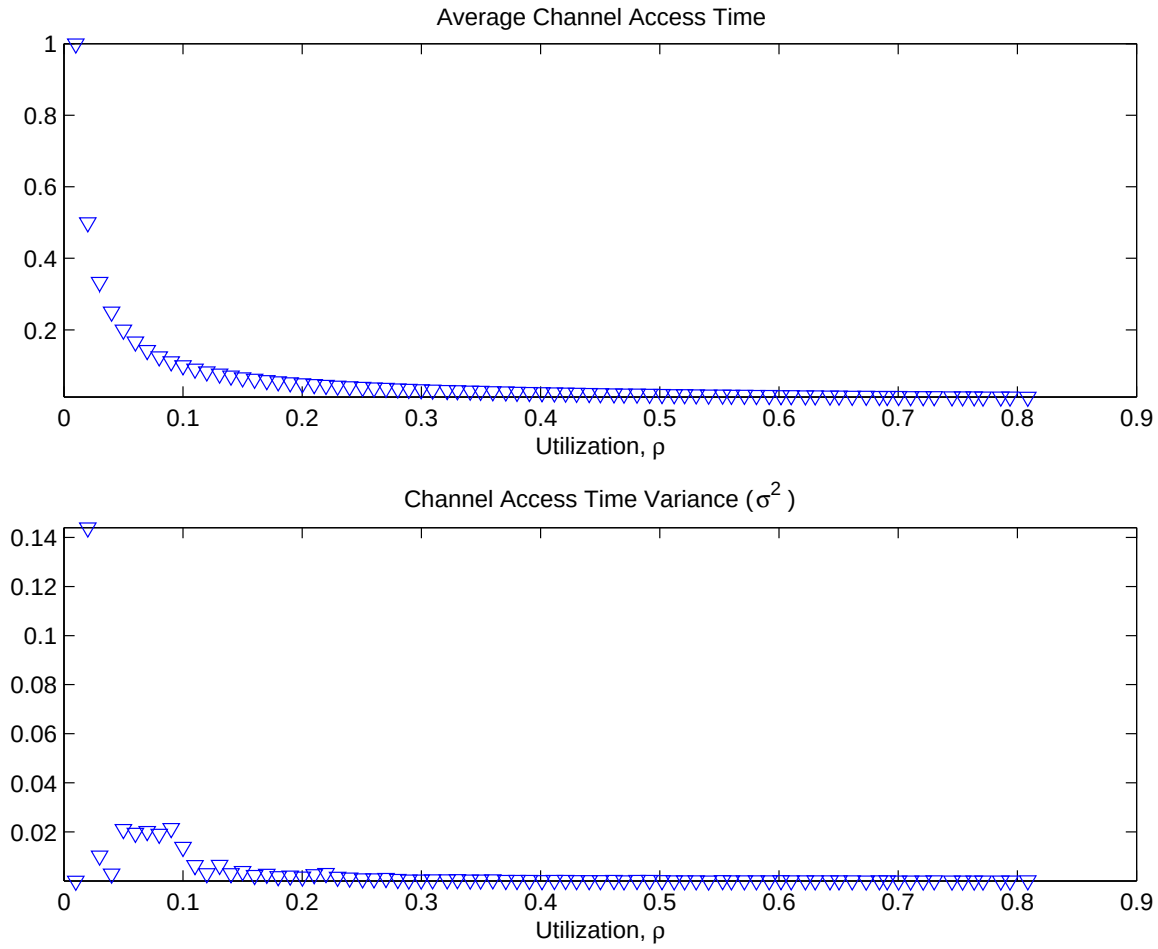

Figure 56: Monte Carlo Simulation: Channel Access Time Statistics, $a = 0.1$ML, Backoff 0-5ML

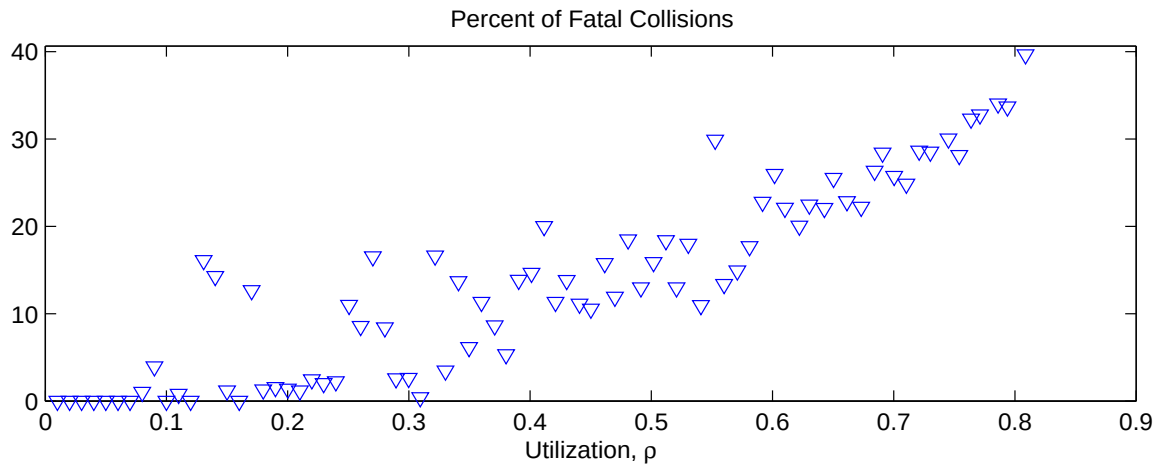

Figure 57: Monte Carlo Simulation: Fatal Collision Statistics, $a = 0.1$ML, Backoff 0-5ML

6 Linear Matrix Inequality

This section describes an alternative solution to the stable control of multiple cooperative robotic vehicles. Instead of ensuring that the test matrix W is an M-matrix as in Section 3.1, we can also solve the same problem by formulating it as a Linear Matrix Inequality (LMI) problem [37]. We originally thought that the LMI formulation would allow for a faster solution. However, after much analysis and testing, we have found that this formulation actually results in longer computation times since it is solving a more general problem than our simple example given in Section 3.1. In fact, the size of the Semi-Definite Program (SDP) used to solve the LMI problem grows proportional to the number of robots squared. Although efficient interior point methods exist for medium-size SDPs, this stability problem becomes intractable as the number of robots grows large. In the following analysis, a cutting plane algorithm was developed as an alternative approach to the interior point algorithm. This cutting plane algorithm uses a semiinfinite reformulation of the SDP and solves it using a cutting plane scheme [38].

6.1 SDP Formulation of Multiple Robot Vehicle Stability Problem

The LMI optimization problem is given by

$$\begin{aligned} \min \quad & \theta \\ \text{subj} \quad & \tilde{P} \succ 0 \\ & \begin{bmatrix} -\tilde{P} & A^T\tilde{P} & A^T\tilde{P} & H^T \\ \tilde{P}A & \tilde{P}-I & 0 & 0 \\ \tilde{P}A & 0 & -\tilde{P} & 0 \\ H & 0 & 0 & -\theta I \end{bmatrix} \prec 0 \end{aligned}$$

where

- H: Describes the interaction between the robots, i.e. $(H)_{ij} = A_{ij}$ in Equation (46).
- A: Describes the internal dynamics of each robot, i.e. $A = diag(A_{11}, ..., A_{NN})$ in Equation (46).
- r: The number of robots.
- $m = \frac{r(r+1)}{2} + 1$: The number of variables.
- $n = 4r$: The size of the semidefinite constraint.
- $P \in S^{r\times r}$: Variable matrix.
- $\theta \in \Re$: Scalar variable $\theta = \frac{1}{e_{ij}^2}$ in Equation (46). The system is robustly stable with degree $\alpha = \sqrt{\frac{1}{\theta}}$ if the problem is feasible.

This may be reformulated into

$$\begin{aligned} \min \quad & \theta \\ \text{subj} \quad & X = \begin{bmatrix} \tilde{P} & -A^T\tilde{P} & -A^T\tilde{P} & -H^T \\ -\tilde{P}A & I-\tilde{P} & 0 & 0 \\ -\tilde{P}A & 0 & \tilde{P} & 0 \\ -H & 0 & 0 & \theta I \end{bmatrix} \succ 0 \end{aligned}$$

from the negation of the semi-definite constraint and the Schur complement theorem.

The primal problem in standard form is given by

$$\begin{aligned} \min \quad & \theta \\ \text{subj} \quad & X = F_m\theta + \sum_{i=1}^{r}\sum_{j=1}^{i} F_{\beta(i,j)}p_{\beta(i,j)} - F_0 \succeq 0 \end{aligned}$$

where

- $\beta(i,j) = \frac{i}{2}(1+2r-i)+j-r$: Indexing function
- $P_{i,j} = p_{\beta(i,j)}$
- $F_0 = \begin{bmatrix} 0 & 0 & 0 & -H^T \\ 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -H & 0 & 0 & 0 \end{bmatrix}$
- $F_{\alpha(i,j)} = \begin{bmatrix} E_{i,j} & -(E_{i,i}A)^T & -(E_{i,j}A)^T & 0 \\ -E_{i,j}A & -E_{i,j} & 0 & 0 \\ -E_{i,j}A & 0 & E_{i,j} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$
- $F_{m+1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$
- $E_{i,j} = \begin{cases} e_i e_j^T + e_j e_i^T & i \neq j \\ e_i e_j^T & otherwise \end{cases}$

6.2 The Cutting Plane Algorithm

The original cutting plane algorithm was developed by Krishnan and Mitchell[38]. This formulation duplicates many of their results using the same problem formulation as the solver SDPA[39].

6.2.1 Semidefinite Program

The SDP primal problem is given by

$$\begin{array}{lll} \min & f(x) = c^T x & \\ \text{subj} & X = \sum_{i=1}^{m} F_i x_i - F_0 \succeq 0 & \text{(SDP)} \end{array}$$

where

- $X \in \mathcal{S}^{n\times n}$: variable matrix.
- $F_i \in \mathcal{S}^{n\times n}$: constraint matrices.
- $c \in \Re^{m\times 1}$: cost vector.
- $x \in \Re^{m\times 1}$: variable vector.

The SDP dual problem is given by

$$\begin{array}{lll} \max & F_0 \bullet Y & \\ \text{subj} & F_i \bullet Y = c_i \qquad i = 1, \ldots, m & \text{(SDD)} \\ & Y \succeq 0 & \end{array}$$

where

- $Y \in \mathcal{S}^{n\times n}$: variable matrix.

Assumption 1. *Both (SDP) and (SDD) has strictly feasible solutions.*

Assumption 2. *The matrices F_i are all linearly independent.*

6.2.2 Semiinfinite Program Reformulation of SDP

Recall that $X \succ 0$ is equivalent to $dd^T \bullet C > 0 \, \forall d \in B$ where B is a compact set. This allows (SDP) to be reformulated as a semiinfinite programming problem.

The SIP primal problem is given by

$$\begin{array}{lll} \min & f(x) = c^T x & \\ \text{subj} & \sum_{i=1}^{m} (dd^T \bullet F_i) x_i \geq dd^T \bullet F_0 \qquad \forall d \in B & \text{(SIP)} \end{array}$$

where

- B: compact set in $\Re^{n\times 1}$.

6.2.3 Linear Program Reformulation of SIP

Taking a finite subset of the infinite number of constraints from (SIP) gives a linear programming problem.

The LP primal problem is given by

$$\begin{aligned} \min \quad & f(x) = c^T x \\ \text{subj} \quad & \sum_{i=1}^{m} (d_j d_j^T \bullet F_i) x_i \geq d_j d_j^T \bullet F_0 \qquad j = 1, \ldots, p \end{aligned} \tag{LPP}$$

where

- p is arbitrary.

The LP dual problem is given by

$$\begin{aligned} \max \quad & F_0 \bullet (\sum_{j=1}^{p} y_j d_j d_j^T) \\ \text{subj} \quad & F_i \bullet (\sum_{j=1}^{p} y_j d_j d_j^T) = c_i \qquad i = 1, \ldots, m \end{aligned} \tag{LPD}$$

where

- $y \in \Re^+ \bigcup \{0\}$

Since (LPP) is a relaxation of (SDP), a feasible solution of (LPP) isn't necessarily feasible in (SDP). Conversely, (LPD) is more constrained than (SDD), so all feasible solutions of (LPD) are feasible in (SDD). This is given formally by the following theorem.

Theorem 1. *Any feasible solution to (LPD) gives a feasible solution to (SDD).*

Proof. Let, $Y = \sum_{j=1}^{p} y_j d_j d_j^T$

I must show that $F_i \bullet Y = c_i$ and $Y \succeq 0$.

The first part following immediately from (LPD).

The second part can be seen by noting

$$\begin{aligned} d^T Y d &= d^T (\sum_{j=1}^{p} y_j d_j d_j^T) d \\ &= \sum_{j=1}^{p} y_j d^T (d_j d_j^T) d \\ &= \sum_{j=1}^{p} y_j (d^T d_j)^2 \\ &\geq 0 \end{aligned}$$

since $y_j \geq 0 \forall j$. □

A solution to (SDP) is optimal when the duality gap between (SDP) and (SDD) is 0. The following theorem shows that if the optimal solution of (LPP) is feasible, then it is also an optimal solution to (SDP).

Theorem 2. *Let x^* be an optimal solution to (LPP) and y^* be an optimal solution to (LPD). If x^* is feasible in (SDP), then x^* is optimal in (SDP).*

Proof. Let $Y = \sum_{j=1}^{p} y_j^* d_j d_j^T$ and $X = \sum_{i=1}^{m} F_i x_i^* - F_0$.

The duality gap is given by

$$\begin{aligned} tr(YX) &= tr((\sum_{j=1}^{p} y_j^* d_j d_j^T)(\sum_{i=1}^{m} F_i x_i^* - F_0)) \\ &= (\sum_{j=1}^{p} y_j^* d_j d_j^T) \bullet (\sum_{i=1}^{m} F_i x_i^* - F_0) \\ &= \sum_{j=1}^{p} y_j^* (d_j d_j^T \bullet (\sum_{i=1}^{m} F_i x_i^* - F_0)) \\ &= \sum_{j=1}^{p} y_j^* (\sum_{i=1}^{m} (d_j d_j^T \bullet F_i) x_i^* - d_j d_j^T \bullet F_0) \\ &= 0 \end{aligned}$$

from complementary slackness at optimality for the dual pair (LPP) and (LPD).

Since x^* and y^* are both feasible and the duality gap is 0, x^* is an optimal solution to SDP. □

6.2.4 Bounding the LP

Assuming that the original SDP is bounded, there exists M such that $c^T x > M \, \forall x$. Initializing the LP with this constraint insures that the LP is bounded. The constant M depends on the individual problem.

6.2.5 Optimal Set of Constraints

Assume that the current optimal solution is infeasible. Then, adding more constraints to (LPP) should produce a solution closer to feasibility. However, not all constraints will produce an equivalent improvement. Some constraints are redundant and others produce a very marginal gain. So, an intelligent choice of constraints allows the method to converge faster.

Theorem 3 (Valid Cut). *A cut generated from*

$$\sum_{i=1}^{m} (d^{(k)} d^{(k)T} \bullet F_i) x_i \geq d^{(k)} d^{(k)T} \bullet F_0$$

where

- $x^{(k)}$*: optimal solution of (LPP) at iteration* k*.*
- $X^{(k)} = \sum_{i=1}^{m} F_i x_i^{(k)} - F_0 \not\succeq 0$
- $d^{(k)} \in B \ni d^{(k)} X^{(k)} d^{(k)} < 0$.

is deep.

Proof.

$$d^{(k)} X^{(k)} d^{(k)} < 0 \Longrightarrow \sum_{i=1}^{m} (d^{(k)} d^{(k)T} \bullet F_i) x_i^{(k)} - d^{(k)} d^{(k)T} \bullet F_0 < 0$$

So, the current point is solution is infeasible and the cut is deep. □

The following theorem describes how to find $d^{(k)}$ quickly and efficiently.

Theorem 4. *Let* $B = \{d \ni \|d\|_2^2 \leq 1\}$*. Every eigenvector* $d^{(k)}$ *that corresponds to a negative eigenvalue of* $X^{(k)}$ *generates a valid cut.*

Proof. Let $\lambda^{(k)}$ be the eigenvalue of $X^{(k)}$ that corresponds to $d^{(k)}$.
Notice,

$$\begin{aligned} d^{(k)T} X^{(k)} d^{(k)} &= d^{(k)T} \lambda^{(k)} d^{(k)} \\ &= \lambda^{(k)} \end{aligned}$$

However, $\lambda^{(k)} < 0$ by assumption. So, the cut is valid. □

Although adding multiple cuts per iteration could potentially increase performance, the next theorem describes the optimal cut.

Theorem 5 (QSP). *A cut generated by letting*

$$d^{(k)} = \arg\min_{d} d^T X^{(k)} d$$

subject to

$$d^{(k)} \in B$$

describes the most violated constraint.

Proof. The amount a constraint is violated is given by $dd^T \bullet F_0 - \sum_{i=1}^{m} (dd^T \bullet F_i) x_i^{(k)}$ for some d.

Since,

$$\begin{aligned} d^{(k)} &= \arg\min_d d^T X^{(k)} d \\ &= \arg\min_d \sum_{i=1}^{m} (dd^T \bullet F_i) x_i^{(k)} - dd^T \bullet F_0 \\ &= \arg\max_d dd^T \bullet F_0 - \sum_{i=1}^{m} (dd^T \bullet F_i) x_i^{(k)} \end{aligned}$$

the violation is maximized. □

The maximally violated constraint is easily solved by the following theorem.

Theorem 6. *Let $B = \{d \ni \|d\|_2^2 \leq 1\}$. The eigenvector that corresponds to the smallest algebraic eigenvalue of $X^{(k)}$ minimizes (QSP).*

Proof. There are two cases two consider.

First, assume that the constraint is inactive. This implies that the minimum is an unconstrained minimum. However, this minimum can't exist unless $X^{(k)}$ is positive definite. But, this is a contradiction since by assumption $X^{(k)} \not\succeq 0$.

Second, assume that the constraint is active. From the first order necessary conditions, $X^{(k)}d = \lambda d$. So, all eigenvectors of $X^{(k)}$ are stationary points. Let $d^{(k)}$ be an eigenvector with corresponding eigenvalue $\lambda^{(k)}$. Then, $d^{(k)T} X^{(k)} d^{(k)} = d^{(k)T} \lambda^{(k)} d^{(k)} = \lambda^{(k)}$. Thus, the smallest algebraic eigenvalue's eigenvector minimizes (QSP). □

6.2.6 Stopping Conditions

The algorithm has the option to use four different stopping conditions. The first two stopping conditions consider the progress the method is making toward an optimal solution. However, these stopping conditions perform very poorly. The second two stopping conditions consider the feasibility of the current solution.

The method has converged when it is no longer gaining progress toward the optimal solution. Notice, the objective value at every iteration converges monotonically to the optimal solution. So, when $f(x^{(k)}) - f(x^{(k-1)}) \leq \epsilon_1(1 + f(x^{(k)}))$, the method has converged. Unfortunately, there is no guarantee that the objective value will change when a cut is taken. So, this stopping conditions often fails before a good solution is reached.

The method will also converge when the distance between two successive solutions converges to 0, $\|x^{(k-1)} - x^{(k)}\| \leq \epsilon_2(1 + \|x^{(k)}\|)$. Unfortunately, the cutting plane method requires a very large number of cuts before a good solution is obtained. So, the difference between two sucessive solutions tends to be very small.

Since the relaxation provides a superoptimal solution to the original problem, as the solution approaches feasibility, it also approaches optimality. Recall, since $X \succeq 0$ when all its eigenvalues are nonnegative, the smallest algebraic eigenvalue gives a measure of how close the matrix is to feasibility. So, if λ_n is the smallest

eigenvalue, then the method has converged when $\lambda_n > -\epsilon_3$. Unfortunately, the smallest eigenvalue tends to converge to zero very slowly even the solution is near optimal. Further, the smallest eigenvalue does not necessarily converge monotonically to 0.

The final stopping condition provides the best estimate of optimality, but is much more difficult to determine. Let $\hat{X}^{(k)} = X^{(k)} + E^{(k)} \succeq 0$. Since all feasible solutions have the form $X = \sum_{i=1}^{m} F_i x_i - F_0$, an appropriate matrix $E^{(k)}$ must be found such that $E^{(k)} = \sum_{i=1}^{m} F_i \hat{x}_i - F_0$ and the norm of $E^{(k)}$ converges to 0 as the method converges toward optimality. Once found, $E^{(k)}$ provides an upper bound to the problem that converges toward optimality. So, the gap between the upper and lower bounds tends to 0 as the method converges toward optimality.

6.2.7 Algorithm

1. Initialize: Determine the constraint needed to bound the LP.
2. Find the lower bound: Solve the LP for its optimal solution
3. Check for stopping conditions
4. Remove inactive constraints.
5. Add new cutting planes
6. Find a new lower bound and repeat.

6.2.8 Efficiency of the Algorithm

The two parts of the method that take the most computational time are solving the LP and finding the cutting planes.

Solving the LP The LP is solved using the simplex method. So, it is important to determine whether it is more efficient to solve the primal or dual problem.

The primal LP has m variables. Unfortunately, it has a variable number of constraints. In practice, the primal has far fewer constraints than variables since inactive constraints are eliminated. Conversely, the dual LP has these characteristics reversed. It contains a variable number of variables, but a fixed number of constraints.

Unfortunately, there is no formula that describes the exact running time of the simplex method. However, it is commonly accepted that the time it takes to solve a problem grows more quickly with the number of constraints than the number of variables. Since the number of constraints that the primal problem possess is far less than the number of variables, it is much more efficient to solve the primal problem.

Solving the Quadratic Subproblem Calculating all of the eigenvalues of a matrix requires $O(n^3)$ operations. However, typically only a few eigenvalues per iteration are needed. So, since $X^{(k)}$ is symmetric, Lanczos method can be used to find the smallest eigenvalues. The exact running time of Lanczos method is difficult to determine. The effort per iteration depends on the difficulty of the matrix multiplications and the sparsity of the system. Further, the number of iterations required to converge varies depending on the eigenvalue structure of the matrix. However, in this case, for a limited number of eigenvalues, it is much more efficient to use Lanczos method than a general scheme.

6.3 Bounding the LP Relaxation for the Multiple Robot Problem

Recall that the LP relaxation isn't always bounded. However, for this problem, this simple constraint insures the problem is bounded.

Theorem 7. *The constraint $\theta \geq 0$ bounds the LP. Further, $\theta \geq 0$ for all feasible solutions to (SDP).*

Proof. The proof has two parts.

First, an LP that attempts to minimize θ subject to $\theta \geq 0$ is obviously bounded with a minimum solution of 0.

Second, let

$$\mathcal{A} = \begin{bmatrix} \tilde{P} & -A^T\tilde{P} & -A^T\tilde{P} \\ -\tilde{P}A & I-\tilde{P} & 0 \\ -\tilde{P}A & 0 & \tilde{P} \end{bmatrix}$$

$$\mathcal{B} = \begin{bmatrix} -H^T \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{C} = \theta I$$

From the Schur complement theorem

$$X = \begin{bmatrix} \mathcal{A} & \mathcal{B} \\ \mathcal{B}^T & \mathcal{C} \end{bmatrix} \succeq 0 \Longleftrightarrow \mathcal{A} \succeq 0, \mathcal{C} - \mathcal{B}^T\mathcal{A}^{-1}\mathcal{B} \succeq 0$$

Assume that $X \succeq 0$ and $\theta < 0$.
Notice,

$$\begin{aligned} \mathcal{A} \succeq 0 &\Longrightarrow \mathcal{A}^{-1} \succeq 0 \\ &\Longrightarrow \mathcal{B}^T \mathcal{A}^{-1} \mathcal{B} \succeq 0 \\ &\Longrightarrow -\mathcal{B}^T \mathcal{A}^{-1} \mathcal{B} \preceq 0 \end{aligned}$$

This implies that the largest algebraic eigenvalue of $-\mathcal{B}^T \mathcal{A}^{-1} \mathcal{B}$, λ_1, is less than or equal to 0.

From the shifting property of eigenvalues, the largest algebraic eigenvalue of $\mathcal{C} - \mathcal{B}^T \mathcal{A}^{-1} \mathcal{B}$ is equal to $\theta + \lambda_1$. However, since $\theta < 0$, $\theta + \lambda_1 < 0$. But this is a contradiction, since, by assumption, $\mathcal{C} - \mathcal{B}^T \mathcal{A}^{-1} \mathcal{B} \succeq 0$. □

6.4 Benchmarks

These benchmarks were generated on a 400MHz Pentinum running WindowsNT using three different solvers: CSDP4.1[40], the LMI Toolbox[41], and the cutting plane algorithm CPX. Notice, CSDP4.1 uses a primal-dual interior point method while the LMI toolbox uses a projective method. The solver CPX indicates the cutting plane method with a maximum of X cuts per iteration. This method was implemented using LAPACK's RRR algorithm to find eigenvalues and GLPK[42] to solve the LP relaxations.

The internal dynamics of each robot was given by $A = -.6I$. The interaction between the robots was described by H where each off diagonal element was equal to 1.6.

Robots	CP2	CP4	CP8	CP16	CP32	CSDP4.1	LMI Toolbox
2	.078	.093	.093	N/A	N/A	.109	.125
4	1.484	1.187	1.187	1.218	N/A	.343	.766
8	MAX	MAX	540	514	359	.406	2.09
16	MAX	MAX	MAX	MAX	MAX	1.84	18.2
32	MAX	MAX	MAX	MAX	MAX	21.1	252

Table 5: Run Time of Each Algorithm in Seconds

Table 5 gives the running time of each algorithm in seconds with a maximum time limit of 600s. For any nontrivial problems, the cutting plane method takes far longer than either CSDP4.1 or the LMI Toolbox. For moderately sized problems, the LMI Toolbox has trouble calculating an answer quickly.

Notice, attempting to calculate more cuts per iteration doesn't necessarily increase the performance of the cutting plane algorithm. Recall, only eigenvectors that correspond to negative eigenvalues will produce valid cuts. If too many eigenvalues are calculated, many of them will be positive. So, the computational effort used to produce them is wasted.

Robots	CP2	CP4	CP8	CP16	CP32	CSDP4.1	LMI Toolbox
2	25	25	25	N/A	N/A	11	17
4	375	250	225	200	N/A	10	22
8	15775	14450	11700	10675	7000	10	24
16	575	500	450	400	350	11	26
32	375	200	100	100	75	11	26

Table 6: Number of Iterations Computed

Table 6 gives the number of iterations each method took to converge. Notice, the cutting plane algorithm typically takes a huge number of inexpensive iterations to converge while CSDP4.1 and the LMI Toolbox take a smaller number of more expensive iterations. As the size of the problem increased, the effort needed to compute an iteration of the cutting plane algorithm prevented substantial progress.

Table 7 gives the value of α calculated by each method. The results obtained from CSDP4.1 and the LMI Toolbox were identical while those obtained from the cutting plane algorithm tended to possess more error.

6.5 Future Work

The next two phases involve further optimizing the algorithm and distributing the task asynchronously to multiple agents.

The most expensive step of the algorithm is the simplex method. However, the optimal solution from one iteration to the next changes only slightly. Hence, warm starting the simplex method would probably speed up the process.

The asynchronous distribution of the problem is more difficult. Typically, parallel computers allow operations such as matrix multiplications, eigenvalues computations, or the simplex method to be distributed between machines. However, the communication cost between two robots is much higher than it is between multiple processors. So, a new scheme must be developed.

Robots	CP2	CP4	CP8	CP16	CP32
2	2.500854e-1	2.501291e-1	2.501801e-1	N/A	N/A
4	1.547215e-1	1.547063e-1	1.545778e-1	1.547429e-1	N/A
8	1.380454e-1	1.337765e-1	1.330610e-1	1.330866e-1	1.331674e-1
16	4.469729e-1	4.326303e-1	4.394678e-1	4.266139e-0	3.548570e-1
32	∞	∞	1.908568e+7	∞	∞

Robots	CSDP4.1	LMI Toolbox
2	2.500000e-1	2.500000e-1
4	1.545085e-1	1.545085e-1
8	1.330222e-1	1.330222e-1
16	1.271652e-1	1.271652e-1
32	1.255686e-1	1.255686e-1

Table 7: Value of α Calculated

7 Conclusions

This project took an integrated approach to designing communication, sensing, and control systems for fixed and mobile distributed systems. Our analysis built upon our past experience in developing provably convergent cooperative controls and upon concepts from graph theory as applied to communication networks. A common mathematical framework consisting of three steps was developed for creating decentralized cooperative controls laws. The first step is to define a global performance index for the cooperative behavior. The second step is to partition the performance index so that only local interactions are included. The third step is to create a first or second order gradient control law that minimized the partitioned performance index. After these three steps, a vector Liapunov technique is used to determine the stability constraints on the individual subsystem control gains, interaction control gains, and the communication sampling period.

Next, we evaluated which communication protocols could meet the constraint on communication sampling period. Coloring algorithms from graph theory were used to compare the performance of TDMA or CSMA communication networks. In general,

we found that TDMA networks will outperform CSMA networks if the network is stationary, and there is sufficient time to perform the network coloring algorithms. However, if the network is moving, then a CSMA network will outperform the TDMA network as long as the network utilization stays below 58 percent. Queuing theory was used to determine a closed form solution of the time response of a CSMA network, and Monte Carlo simulations were used to verify the solution. The vector Liapunov analysis shows that the collective CSMA networked system will still remain stable as long as the maximum communication time is below the maximum sampling period determined from the vector Liapunov analysis.

The report concluded by evaluating using Linear Matrix Inequality (LMI) instead of the vector Liapunov analysis. Our initial belief was that the stability problem could be solved more efficiently using LMI equations instead the M-matrix technique, and that we might be able to use this technique on-line to adjust the communication sampling period and control gains so that the system remains stable. We found that this is not true, and that in general the LMI equations actually takes more time to solve. An area of future research will be to determine more efficient methods of solving these equations and determining the bounds on the communication sample period.

Portions of this report have been published in [21] and [43].

Acknowledgments

Section 2, Common Mathematical Framework, was primarily written by John Feddema with input from Ray Byrne. Section 3, Behavior Metrics, was the work of John Feddema. Dominique Kilman performed the OPNET^{TM} simulations and authored Section 4, Analysis and Simulation of TDMA and Coloring. Ray Byrne performed the CSMA delay simulations and wrote Section 5, CSMA Delay Characteristics and Simulation Results. The LMI research was performed by Joseph Young who authored Section 6, Linear Matrix Inequality. Ray Byrne coordinated and edited the final document. Rush Robinett was the originator of many of the concepts covered in Section 2. John Harrington and Brian Van Leeuwen provided assistance with all communications sections. Chris Lewis developed most of the RATLER^{TM} demonstrations described in Section 2.

References

[1] T. Arai, E. Pagello, and P. E. Parker, "Guest editorial: Advances in multirobot systems," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 655–659, October 2002.

[2] E. Bonaneau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press, 1999.

[3] R. C. Arkin, "Cooperation without communication: Multiagent schema-based robot navigation," *Journal of Robotic Systems*, vol. 9, no. 3, pp. 351–364, 1992.

[4] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation*, vol. 14, December 1998.

[5] R. C. Kube and H. Zhang, "Collective robotics: From social insects to robots," *Adaptive Behavior*, vol. 2, pp. 189–218, Fall 1993.

[6] Q. Chen and J. Y. S. Luh, "Coordination and control of a group of small mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3, (San Diego, CA), pp. 2315–2320, May 1994.

[7] H. Yamaguchi and T. Arai, "Distributed and autonomous control method for generating shape of multiple mobile robot group," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, vol. 2, (Munich, Germany), pp. 800–807, September 1994.

[8] H. Yamaguchi and J. W. Burdick, "Asymptotic stabilization of multiple nonholonomic mobile robots forming group formations," in *Proceedings of the 1998 Conference on Robotics and Automation*, (Leuven, Belgium), pp. 3573–3580, May 1998.

[9] E. Yoshida, T. Arai, J. Ota, and T. Miki, "Effect of grouping in local communication system of multiple mobile robots," in *Proceedings of the IEEE Conference on Intelligent Robots and Systems*, vol. 2, (Munich, Germany), pp. 808–815, September 1994.

[10] P. Molnar and J. Starke, "Communication fault tolerance in distributed robotic systems," in *Distributed Autonomous Robotic Systems 4* (L. E. Parker, G. Bekey, and J. Barhen, eds.), pp. 99–108, Springer-Verlag, 2000.

[11] F. E. Schneider, D. Wildermuth, and H. L. Wolf, "Motion coordination in formations of multiple robots using a potential field approach," in *Distributed Autonomous Robotic Systems 4* (L. E. Parker, G. Bekey, and J. Barhen, eds.), pp. 305–314, Springer-Verlag, 2000.

[12] G. Beni and P. Liang, "Pattern reconfiguration in swarms - convergence of a distributed asynchronous and bounded iterative algorithm," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 485–490, June 1996.

[13] Y. Liu, K. Passino, and M. Polycarpou, "Stability analysis of one-dimensional asynchronous swarms," in *2001 American Control Conference*, (Arlington, VA), pp. 716–721, June 25-27 2001.

[14] A. Winfield, "Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots," in *Distributed Autonomous Robotic Systems* (L. E. Parker, G. Bekey, and J. Barhen, eds.), pp. 273–282, Springer-Verlag, 2000.

[15] J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proceedings of the 1998 Conference on Robotics and Automation*, (Leuven, Belgium), pp. 2864–2869, May 1998.

[16] J. P. Desai, V. Kumar, and J. P. Ostrowski, "Modeling and control of formations on nonholonomic mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 905–908, December 2001.

[17] C. Lewis, J. T. Feddema, and P. Klarer, "Robotic perimeter detection system," in *Proceedings of the SPIE Volume 3577*, (Boston, MA), pp. 14–21, November 3-5 1998.

[18] J. T. Feddema, C. Lewis, and D. A. Shoenwald, "Decentralized control of robotic vehicles: Theory and application," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 852–864, October 2002.

[19] R. D. Robinett III and J. E. Hurtado, "Stability and control of collective systems," in *Proceedings of the John L. Junkins Astrodynamics Symposium*, (College Station, TX), May 23-24 2003.

[20] B. R. Frieden, *Physics from Fisher Information*. Cambridge University Press, 1998.

[21] J. T. Feddema, R. D. Robinett, and R. H. Byrne, "An optimization approach to distributed controls of multiple robot vehicles." Workshop on Control and Cooperation of Intelligent Miniature Robots, IEEE/RSJ International Conference on Intelligent Robots and Systems, October 31, 2003.

[22] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, 1998.

[23] D. G. Luenberger, *Linear and Nonlinear Programming*. Addison Wesley Publishing Company, 2 ed., 1984.

[24] D. Schmitt, R. Byrne, J. Feddema, G. Fisher, J. Harrington, C. Little, J. Neely, and B. Rigdon, "Intelligent mobile land mine IMLM system," Tech. Rep. SAND2003-1186, Sandia National Laboratories, Albuquerque, NM, July 2003.

[25] J. T. Feddema and D. A. Schoenwald, "Stability analysis of decentralized cooperative controls," in *Multi-Robot Systems: From Swarms to Intelligent Automata* (A. C. Shultz and L. E. Parker, eds.), pp. 122–133, Kluwer Academic Publishers, 2002.

[26] J. T. Feddema and D. A. Schoenwald, "Distributed communications/navigation robot vehicle network," in *Proceedings of the World Automation Congress*, (Orlando, FL), June 9-13 2002.

[27] R. H. Byrne, D. R. Adkins, S. E. Eskridge, J. J. Harrington, E. J. Heller, and J. E. Hurtado, "Miniature mobile robots for plume tracking and source localization research," *Journal of Micromechatronics*, vol. 1, no. 3, pp. 253–261, 2002.

[28] R. H. Byrne, S. E. Eskridge, J. E. Hurtado, and E. L. Savage, "Algorithms and analysis of underwater vehicle plume tracing," Tech. Rep. SAND2003-2643, Sandia National Laboratories, Albuquerque, NM, July 2003.

[29] D. D. Siljak, *Decentralized Control of Complex Systems*. Academic Press, 1991.

[30] M. E. Sezer and D. D. Siljak, "Robust stability of discrete systems," *International Journal of Control*, vol. 48, no. 5, pp. 2055–2063, 1988.

[31] G. D. Smith, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*. Oxford University Press, 3rd ed., 1985.

[32] S. Basagni, D. Bruschi, and I. Chlamtac, "A mobility-transparent deterministic broadcast mechanism for ad hoc networks," *IEEE/ACM Transactions on Networks*, vol. 7, pp. 799–807, December 1999.

[33] I. Chlamtac and S. S. Pinter, "Distributed nodes organization algorithm for channel access in a multihop dynamic radio network," *IEEE/ACM Transactions on Networks*, vol. C-36, pp. 728–737, June 1987.

[34] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: Part I - carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Transactions on Communications*, vol. COM-23, pp. 1400–1416, December 1975.

[35] L. Kleinrock, *Queueing Systems, Volume I: Theory*. New York: John Wiley & Sons, 1 ed., 1975.

[36] S. L. Beuerman and E. J. Coyle, "Delay characteristics of csma/cd networks," *IEEE Transactions on Communications*, vol. 36, pp. 553–563, May 1988.

[37] D. Stipanovic and D. D. Siljak, "Robust stability and stabilization of discrete-time non-linear systems: the LMI approach," *International Journal of Control*, vol. 74, no. 9, pp. 873–879, 2001.

[38] K. Krishnan and J. E. Mitchell, "Properties of Cutting Plane Method for Semidefinite Programming." submitted for publication, May 2003.

[39] K. Fujisawa and M. Kojima, "SDPA(Semidefinite Programming Algorithm) : User's manual," 1995.

[40] B. Borchers, "CSDP, A C Library for Semidefinite Programming," *Optimization Methods and Software 11*, pp. 613–623, 1999.

[41] Mathworks, "LMI Control Toolbox," 1995. Software.

[42] A. Makhorin, "GNU Linear Programming Kit," May 2003. Software.

[43] J. T. Feddema, "Communication and control of large scale cooperative systems," in *Proceedings of the American Nuclear Society* 10^{th} *International Conference on Robotics and Remote Systems for Hazardous Environments*, (Gainesville, FL), March 28-31 2004.

Appendix A

CSMA Simulation Program

```
% Simulation of CSMA Comm
% Ray Byrne, 15234
% 8/30/04

PRINT_DEBUG = 0;  % 1==print each test run, otherwise only summary data

NN = 10;     % number of agents
h = 1;       % update rate in seconds
ML = 0.01;   % message length multiplier (sec)
M = 1e6;     % oscillator uncertainty
SIMS = 10^4; % number of simulations
%B = 0.1;    % back off scaling
B = 5.0*ML;      % backoff is always proportional to message length

P_ACTUAL = []; % actual utilization
actual_utilization = [];

STATS = []   % store statistics of each run

for I=1:80   % loop simulations

%L = ML*I;   % utilization varies from 0.1 to 0.9
L = ML;      % constant message length
N = round(NN*I/10)    % vary number of robots for utilization - fudge factor to get utilization spread correct
STOP = SIMS/N/h; % simulation iterations
a = L/10;    % fatal collision if difference in start time less than a



% start of Simulation

T_0 = rand(N,1);     % random start times
ROBOT_ID = (1:N)';    % N robots
T_0 = [T_0 ROBOT_ID zeros(N,1) zeros(N,1) zeros(N,1)];
T_0 = sortrows(T_0,1);   % arrange in order
T=[];                    % event, time axis
% T = [event_time robot_ID delay collision fatal_collision]

H = rand(N,1);   % add uncertainty to local oscillators
H = H./M + h;

for i=1:(STOP)
  T = [T ; T_0(:,1)+i*H T_0(:,2) L*ones(N,1) T_0(:,4) T_0(:,5)];  % include message lenth in delay
end


LAST_TIME = max(T(:,1))

actual_utilization = 0;  % clear this every run through

n = length(T);

% check ordering
T = sortrows(T,1);


i=2;
while i < (n+1)
    j=0;
    if ((T(i,1) - T(i-1,1)) < a)  % case of  fatal collision
        actual_utilization = actual_utilization + L + (T(i,1) - T(i-1,1));
```

```
        T(i-1,5) = 1;
        T(i,5) = 1;
        % DO NOTHING WITH DELAY FOR FATAL COLLISION CASE - LOSE MESSAGE!
        i = i + 1;


        %if rem(i,1000) == 0 i=i
        %end % end if
    elseif ((T(i,1) - T(i-1,1)) < L)  % case of collision
        % actual_utilization = actual_utilization + L;  % first message
        % gets through, second is rescheduled, -> no channel utilization
        T_old = T(i,1);
        T(i,1) = T(i-1,1) + L + B*rand(1);
        delay = T(i,1) - T_old;
        T(i,3) = T(i,3) + delay;  % add to delay column
        T(i,4) = T(i,4) + 1.0;
        % sort
        if  i+j+1 <= n
            while T(i+1+j,1) < T(i+j,1)
                temp = T(i+j,:);
                T(i+j,:) = T(i+j+1,:);
                T(i+j+1,:) = temp;
                if i+j+2 <= n
                    j=j+1;
                else
                    break;
                end  %end if
            end % while sort
        end % end if
    else
        i = i + 1;
        actual_utilization = actual_utilization + L;
        %if rem(i,1000) == 0 i=i
        %end % end if
    end  % end if
end  % end while

n = length(T);
D = [];
for k=2:n
    diff = T(k,1) - T(k-1,1);
    D = [D diff];
end  % end for

STOP_TIME = max(T(:,1)) + L % sec
P_ACTUAL = [P_ACTUAL; actual_utilization/STOP]

if PRINT_DEBUG == 1
  clf;   % clear graphics

  subplot(2,2,1);
  plot(0);
  axis([0 1 0 1]);
  str = sprintf('Number of robots  = %d', N);
  text(0.05, 0.8, str);
  text(0.05, 0.7, 'Communication Rate = 1/sec');
  str = sprintf('Backoff = 0-%d ms, uniform distribution', B*1000);
  text(0.05, 0.6, str);
  temp = sprintf('Message Length = %f (sec)', L)
  text(0.05, 0.5, temp);
  temp = sprintf('Offered Utilization \\rho = %f', I/10);
  text(0.05, 0.4, temp);
  temp = sprintf('Acutal Utilization \\rho = %f', actual_utilization/STOP);
  text(0.05, 0.3, temp);
  str = sprintf('10^4 Messages in Simulation');
  text(0.05, 0.2, str);
```

```
  title('Simulation Parameters');
end;  % PRINT_DEBUG


delay_avg = mean(T(:,3));
delay_var = var(T(:,3));

if PRINT_DEBUG == 1
  subplot(2,2,2);
  EDGES = 0:0.01:0.5;
  HN = histc(T(:,3),EDGES);
  BAR(EDGES,HN,'histc')
  xlabel('Delay (sec)');
  title('Delay Distribution');
  x_max = max(EDGES);
  y_max = max(HN);
  str = sprintf('Mean = %f', delay_avg);
  text(0.3*x_max, 0.7*y_max, str);
  str = sprintf('Variance = %f', delay_var);
  text(0.3*x_max, 0.6*y_max, str);
end;  % PRINT_DEBUG

trans_avg = mean(T(:,4));
trans_var = var(T(:,4));

if PRINT_DEBUG == 1
  subplot(2,2,3);
  EDGES = -0.5:1:(max(T(:,4))+0.5);
  HN = histc(T(:,4),EDGES);
  BAR(EDGES,HN,'histc');
  xlabel('Transmission Retries');
  title('Transmission Retry Distribution');
  x_max = max(EDGES);
  y_max = max(HN);
  str = sprintf('Mean = %f', trans_avg);
  text(0.3*x_max, 0.7*y_max, str);
  str = sprintf('Variance = %f', trans_var);
  text(0.3*x_max, 0.6*y_max, str);

  subplot(2,2,4);
  plot(T(:,1),T(:,3));
  xlabel('Time (sec)');
  ylabel('Transmission Delay (sec)');
  title('Transmission Delay as a Function of Time');

  fname = sprintf('D:/CSMA/Sim%i', I);
  print('-depsc2',fname);

end;  % PRINT_DEBUG

channel_avg = mean(D);
channel_var = var(D);
fatal_collisions = sum(T(:,5));
fatal_percent = fatal_collisions/10^2;

if PRINT_DEBUG == 1
  clf;
  subplot(2,1,1);
  %EDGES = -0.05:0.1:0.25;
  %HN = histc(D,EDGES);
  %BAR(EDGES,HN,'histc')
  hist(D);
  xlabel('Channel Access Time (sec)');

  str = sprintf('Channel Access Time Distribution, Mean = %f, Variance = %f', channel_avg, channel_var);
  title(str);
```

```
  subplot(2,1,2);
  plot(T(:,1),T(:,5));

  str = sprintf('Fatal Collisions, Total = %i, Percent of Messages = %0.2f %%, a = %f',
                                                          fatal_collisions, fatal_percent,a);
  xlabel('Time (sec)');
  title(str);

  fname = sprintf('D:/CSMA/Sim%ia', I);
  print('-depsc2',fname);
end;  % PRINT_DEBUG

STATS = [STATS; delay_avg delay_var trans_avg trans_var channel_avg channel_var fatal_percent];

end  % end of simulation loop


% DO MORE SIMULATIONS AT LAST VALUE OF I (~45), OR DON'T DO!

for J=1:0   % loop simulations

%L = ML*I;   % utilization varies from 0.1 to 0.9
L = ML;      % constant message length
N = round(NN*I/10)    % vary number of robots for utilization - fudge factor to get utilization spread correct
STOP = SIMS/N/h; % simulation iterations
a = L/10;    % fatal collision if difference in start time less than a

% start of Simulation

T_0 = rand(N,1);     % random start times
ROBOT_ID = (1:N)';    % N robots
T_0 = [T_0 ROBOT_ID zeros(N,1) zeros(N,1) zeros(N,1)];
T_0 = sortrows(T_0,1);   % arrange in order
T=[];                    % event, time axis
% T = [event_time robot_ID delay collision fatal_collision]

H = rand(N,1);   % add uncertainty to local oscillators
H = H./M + h;

for i=1:(STOP)
  T = [T ; T_0(:,1)+i*H T_0(:,2) L*ones(N,1) T_0(:,4) T_0(:,5)];  % include message lenth in delay
end


LAST_TIME = max(T(:,1))

actual_utilization = 0;  % clear this every run through

n = length(T);

% check ordering
T = sortrows(T,1);


i=2;
while i < (n+1)
    j=0;
    if ((T(i,1) - T(i-1,1)) < a)  % case of  fatal collision
        actual_utilization = actual_utilization + L + (T(i,1) - T(i-1,1));
        T(i-1,5) = 1;
        T(i,5) = 1;
        % DO NOTHING WITH DELAY FOR FATAL COLLISION CASE - LOSE MESSAGE!
        i = i + 1;
        %if rem(i,1000) == 0 i=i
        %end % end if
```

```
    elseif ((T(i,1) - T(i-1,1)) < L)  % case of collision, detected - re-schedule
        % actual_utilization = actual_utilization + L;  % first message
        % gets through, second is rescheduled, -> no channel utilization
        T_old = T(i,1);
        T(i,1) = T(i-1,1) + L + B*rand(1);
        delay = T(i,1) - T_old;
        T(i,3) = T(i,3) + delay;  % add to delay column
        T(i,4) = T(i,4) + 1.0;
        % sort
        if  i+j+1 <= n
            while T(i+1+j,1) < T(i+j,1)
                temp = T(i+j,:);
                T(i+j,:) = T(i+j+1,:);
                T(i+j+1,:) = temp;
                if i+j+2 <= n
                    j=j+1;
                else
                    break;
                end  %end if
            end % while sort
        end % end if
    else
        i = i + 1;
        actual_utilization = actual_utilization + L;
        %if rem(i,1000) == 0 i=i
        %end % end if
    end  % end if
end  % end while

n = length(T);
D = [];
for k=2:n
    diff = T(k,1) - T(k-1,1);
    D = [D diff];
end  % end for

STOP_TIME = max(T(:,1)) + L % sec
P_ACTUAL = [P_ACTUAL; actual_utilization/STOP]

if PRINT_DEBUG == 1
  clf;   % clear graphics

  subplot(2,2,1);
  plot(0);
  axis([0 1 0 1]);
  str = sprintf('Number of robots  = %d', N);
  text(0.05, 0.8, str);
  text(0.05, 0.7, 'Communication Rate = 1/sec');
  str = sprintf('Backoff = 0-%d ms, uniform distribution', B*1000);
  text(0.05, 0.6, str);
  temp = sprintf('Message Length = %f (sec)', L)
  text(0.05, 0.5, temp);
  temp = sprintf('Offered Utilization \\rho = %f', I/10);
  text(0.05, 0.4, temp);
  temp = sprintf('Acutal Utilization \\rho = %f', actual_utilization/STOP);
  text(0.05, 0.3, temp);
  str = sprintf('10^4 Messages in Simulation');
  text(0.05, 0.2, str);
  title('Simulation Parameters');
end;  % PRINT_DEBUG


delay_avg = mean(T(:,3));
delay_var = var(T(:,3));

if PRINT_DEBUG == 1
```

```
  subplot(2,2,2);
  EDGES = 0:0.01:0.5;
  HN = histc(T(:,3),EDGES);
  BAR(EDGES,HN,'histc')
  xlabel('Delay (sec)');
  title('Delay Distribution');
  x_max = max(EDGES);
  y_max = max(HN);
  str = sprintf('Mean = %f', delay_avg);
  text(0.3*x_max, 0.7*y_max, str);
  str = sprintf('Variance = %f', delay_var);
  text(0.3*x_max, 0.6*y_max, str);
end;  % PRINT_DEBUG

trans_avg = mean(T(:,4));
trans_var = var(T(:,4));

if PRINT_DEBUG == 1
  subplot(2,2,3);
  EDGES = -0.5:1:(max(T(:,4))+0.5);
  HN = histc(T(:,4),EDGES);
  BAR(EDGES,HN,'histc');
  xlabel('Transmission Retries');
  title('Transmission Retry Distribution');
  x_max = max(EDGES);
  y_max = max(HN);
  str = sprintf('Mean = %f', trans_avg);
  text(0.3*x_max, 0.7*y_max, str);
  str = sprintf('Variance = %f', trans_var);
  text(0.3*x_max, 0.6*y_max, str);

  subplot(2,2,4);
  plot(T(:,1),T(:,3));
  xlabel('Time (sec)');
  ylabel('Transmission Delay (sec)');
  title('Transmission Delay as a Function of Time');

  fname = sprintf('D:/CSMA/Sim%i', I);
  print('-depsc2',fname);

end;  % PRINT_DEBUG

channel_avg = mean(D);
channel_var = var(D);
fatal_collisions = sum(T(:,5));
fatal_percent = fatal_collisions/10^2;

if PRINT_DEBUG == 1
  clf;
  subplot(2,1,1);
  %EDGES = -0.05:0.1:0.25;
  %HN = histc(D,EDGES);
  %BAR(EDGES,HN,'histc')
  hist(D);
  xlabel('Channel Access Time (sec)');

  str = sprintf('Channel Access Time Distribution, Mean = %f, Variance = %f', channel_avg, channel_var);
  title(str);

  subplot(2,1,2);
  plot(T(:,1),T(:,5));

  str = sprintf('Fatal Collisions, Total = %i, Percent of Messages = %0.2f %%, a = %f',
                                                    fatal_collisions, fatal_percent,a);
  xlabel('Time (sec)');
  title(str);
```

```
  fname = sprintf('D:/CSMA/Sim%ia', I);
  print('-depsc2',fname);
end;  % PRINT_DEBUG

STATS = [STATS; delay_avg delay_var trans_avg trans_var channel_avg channel_var fatal_percent];

end  % end of simulation loop



clf;
plot(P_ACTUAL, (1/ML)*STATS(:,1),'v');
axis([0 0.9 1 max((1/ML)*STATS(:,1))]);
xlabel('Utilization, \rho');
title('Normalized Average Delay ');
hold on
p = 0:0.01:0.9;
W=p./2./(1-p) + 1;
plot(p,W,'r')

legend('Monte Carlo Simulation', 'M/D/1 Approximation',2);

fname = sprintf('D:/CSMA/F_Summary1');
print('-depsc2',fname);

clf;
plot(P_ACTUAL,STATS(:,2),'v');
axis([0 0.9 min(STATS(:,2)) max(STATS(:,2))])
xlabel('Utilization, \rho');
title('Delay Variance ( \sigma^2 )');

fname = sprintf('D:/CSMA/F_Summary1a');
print('-depsc2',fname);

clf;

subplot(2,1,1);
plot(P_ACTUAL,STATS(:,3),'v');
axis([0 0.9 min(STATS(:,3)) max(STATS(:,3))])
xlabel('Utilization, \rho');
title('Average Number of Transmission Retries');

subplot(2,1,2);
plot(P_ACTUAL,STATS(:,4),'v');
axis([0 0.9 min(STATS(:,4)) max(STATS(:,4))])
xlabel('Utilization, \rho');
title('Number of Transmission Retries Variance ( \sigma^2 )');

fname = sprintf('D:/CSMA/F_Summary2');
print('-depsc2',fname);

clf;

subplot(2,1,1);
plot(P_ACTUAL,STATS(:,5),'v');
axis([0 0.9 min(STATS(:,5)) max(STATS(:,5))])
xlabel('Utilization, \rho');
title('Average Channel Access Time');

subplot(2,1,2);
plot(P_ACTUAL,STATS(:,6),'v');
axis([0 0.9 min(STATS(:,6)) max(STATS(:,6))])
xlabel('Utilization, \rho');
title('Channel Access Time Variance ( \sigma^2 )');
```

```
fname = sprintf('D:/CSMA/F_Summary3');
print('-depsc2',fname);

clf;

subplot(2,1,1);
plot(P_ACTUAL,STATS(:,7),'v');
axis([0 0.9 min(STATS(:,7)) (max(STATS(:,7)+1))])
xlabel('Utilization, \rho');
title('Percent of Fatal Collisions');

% subplot(2,1,2);


fname = sprintf('D:/CSMA/F_Summary4');
print('-depsc2',fname);
```

This page intentionally blank.

Distribution

5	MS0741	Rush Robinett, 6200
1	MS0785	Robert Hutchinson, 5516
1	MS0785	Dominique Kilman, 5616
1	MS0785	Brian Van Leeuwen, 5616
1	MS1002	Philip Heermann, 15230
1	MS1002	Steve Roehrig, 15200
1	MS1003	Barry Spletzer, 15200
20	MS1003	John Feddema, 15234
10	MS1003	Ray Byrne, 15234
1	MS1003	Joeseph Young, 15234
1	MS1004	Elaine Hinman-Sweeney, 15231
1	MS1005	Russ Skocypec, 15240
1	MS1005	Chuck Duus, 15201
1	MS1007	Larry Shipers, 15232
1	MS1010	Kelly Hays, 15233
1	MS1125	Phil Bennett, 15244
1	MS1125	John Harrington, 15244
1	MS1176	Robert Cranwell, 15243
1	MS1188	John Wagner, 15241
1	MS9018	Central Technical Files, 8945-1
2	MS0899	Technical Library, 9616