TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No. TX-92/1245-3	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Communications in Intelligent Vehicle Highway Systems--Part II		5. Report Date November 1991 6. Performing Organization Code
7. Author(s) Costas N. Georghiades Emina Soljanin, Kuro-hsin Chang Ramaian Velidi, Luis Cavalheiro Matias		8. Performing Organization Report No. Research Report 1245-3
9. Performing Organization Name and Address Texas Transportation Institute The Texas A&M University System College Station, Texas 77843		10. Work Unit No. 11. Contract or Grant No. Study No. 2-1-90/1-1245
12. Sponsoring Agency Name and Address Texas Department of Transportation P.O. Box 5051 Austin, Texas 78701		13. Type of Report and Period Covered 14. Sponsoring Agency Code
15. Supplementary Notes Research performed with all State Funds Research Study Title: Texas Advanced Transportation Technology Project		

16. Abstract

This report is a theoretical analysis of several communications techniques that will be useful in the full implementation of IVHS technologies. The report is a continuation of the work completed in Part I. It looks at some of the communications problems that have been mentioned as bringing some difficulties to spread spectrum, such as the near-far problem. The report also investigates the simulation protocol that could be used with spread spectrum radio to allow for individualized messages to be sent from traffic management centers to vehicles. The use of infra red as a multi-access communications technique is also investigated. Infra red was chosen since it is the system that is used by Semens in their Lisb and Autoguide systems of vehicle navigation. The third chapter discussed a totally different type of communications. This chapter looks at using radio frequency transponders on vehicles to determine the exact lateral and headway between vehicles. The Path program in California is currently looking at a combination of Radar and magnetic nail in the pavement to accomplish the same result. The method suggested in the report would not require any Radar signals nor devices in the pavement. The RF signals used by the transponders would be well below FCC license requirements.

7. Key Words Telecommunications, Spread Spectrum, Vehicle Communications		18. Distribution Statement No restrictions.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 119	22. Price

Form DOT F 1700.7 (8-69)

COMMUNICATIONS IN INTELLIGENT VEHICLE HIGHWAY SYSTEMS-- PART II

by

Costas N. Georghiades
Assistant Research Specialist

and
the following Research Assistants

Emina Soljanin

Kuor-hsin Chang

Ramaiah Velidi

Luis Cavalheiro Matias

Texas Transportation Institute

Research Report 1245-3

Research Study Number 2-1-90/1-1245
Texas Advanced Transportation Technology Project

Sponsored by

Texas Department of Transportation

Texas Transportation Institute
The Texas A&M University System
College Station, Texas 77843

November 1991

METRIC (SI*) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
	LENGTH			
in	inches	2.54	centimetres	cm
ft	feet	0.3048	metres	m
yd	yards	0.914	metres	m
mi	miles	1.61	kilometres	km
	AREA			
in^2	square inches	645.2	centimetres squared	cm^2
ft^2	square feet	0.0929	metres squared	m^2
yd^2	square yards	0.836	metres squared	m^2
mi^2	square miles	2.59	kilometres squared	km^2
ac	acres	0.395	hectares	ha
	MASS (weight)			
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams	Mg
	VOLUME			
fl oz	fluid ounces	29.57	millilitres	mL
gal	gallons	3.785	litres	L
ft^3	cubic feet	0.0328	metres cubed	m^3
yd^3	cubic yards	0.0765	metres cubed	m^3

NOTE: Volumes greater than 1000 L shall be shown in m^3.

TEMPERATURE (exact)

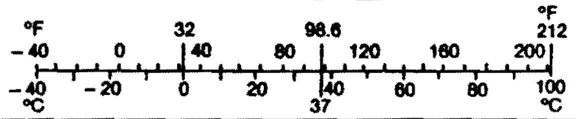
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
	LENGTH			
mm	millimetres	0.039	inches	in
m	metres	3.28	feet	ft
m	metres	1.09	yards	yd
km	kilometres	0.621	miles	mi
	AREA			
mm^2	millimetres squared	0.0016	square inches	in^2
m^2	metres squared	10.764	square feet	ft^2
km^2	kilometres squared	0.39	square miles	mi^2
ha	hectores (10 000 m^2)	2.53	acres	ac
	MASS (weight)			
g	grams	0.0353	ounces	oz
kg	kilograms	2.205	pounds	lb
Mg	megagrams (1 000 kg)	1.103	short tons	T
	VOLUME			
mL	millilitres	0.034	fluid ounces	fl oz
L	litres	0.264	gallons	gal
m^3	metres cubed	35.315	cubic feet	ft^3
m^3	metres cubed	1.308	cubic yards	yd^3

TEMPERATURE (exact)

°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



These factors conform to the requirement of FHWA Order 5190.1A.

* SI is the symbol for the International System of Measurements

IMPLEMENTATION STATEMENT

The Texas Department of Transportation is investigating methods of providing increased mobility for the major urban areas of Texas. One area being examined is to install Traffic Management Centers that would control the signal systems on freeway ramps as well as frontage roads and other major aterial in the vicinity of the freeway. This report investigates strategies of communicating with vehicles that are using the freeway and urban street system. While the findings of this report may not be installed as a part of the initial traffic management centers it should provide a context for planning the centers so they will be compatible with the most recent state-of-the-art communications techniques.

DISCLAMER

ABSTRACT

This report is a theoretical analysis of several communications techniques that will be useful in the full implementation of IVHS technologies. The report is a continuation of the work completed in Part I. It looks at some of the communications problems that have been mentioned as bringing some difficulties to spread spectrum, such as the near-far problem. The report also investigates the simulation protocol that could be used with spread spectrum radio to allow for individualized messages to be sent from traffic management centers to vehicles. The use of infra red as a multi-access communications technique is also investigated. Infra red was chosen since it is the system that is used by Semens in their Lisb and Autoguide systems of vehicle navigation. The third chapter discussed a totally different type of communications. This chapter looks at using radio frequency transponders on vehicles to determine the exact lateral and headway between vehicles. The Path program in California is currently looking at a combination of Radar and magnetic nail in the pavement to accomplish the same result. The method suggested in the report would not require any Radar signals nor devices in the pavement. The RF signals used by the transponders would be well below FCC license requirements.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

THE NEAR-FAR PROBLEM IN SPREAD-SPECTRUM ROAD-AUTOMOBILE COMMUNICATIONS

A. Introduction

Communication between roadside beacons and vehicles on the highway may be provided by a multiuser radio communication system utilizing a Code Division Multiple Access (CDMA) technique [1],[2]. In a CDMA system multiple users transmit simultaneously, asynchronously, and through the same channel. The receiver in such a system demodulates all or a subset of the transmitted messages, based on the received superposition of all the transmitted waveforms and the channel noise. The optimal demodulating strategy for these systems [3] requires a certain computational complexity which makes it almost impractical.

The detection strategy most frequently used in practice is the so-called *conventional single user detection,* which minimizes the probabability of error in a single-user channel, i.e., in the absence of interfering users. The justification for using this strategy lies in the philosophy of the CDMA technique. In this technique, each transmitter is assigned a fixed, distinct signature waveform which is used to modulate a message as in single user communication. These signature waveforms are known to the receiver which correlates the composite received signal with the replica of the signature waveform assigned to the user whose message is being demodulated. If the crosscorrelations of the signature waveforms assigned to the simultaneous users are low for all possible relative delays, and if the powers of the received signals are similar, the performance of the system utilizing the conventional single user detection technique is satisfactory as shown in the first part of this project [1], where we examined power, bandwidth, and complexity requirements for achieving a desired error-probability performance and communicating a given amount of information between a beacon and moving vehicles, under some worst-case traffic scenario.

Low crosscorrelations of signature waveforms in this system were achieved by employing spread-spectrum pseudo-noise sequences of long periodicity, and the assumption was made that the power levels received by the beacon from all the simultaneous users (vehicles) were equal. On the other hand, the fact is that, in this multiple user radio communication system, a particular roadside beacon communicates simultaneously with the vehicles that are randomly distributed in the area assigned to the beacon. This spatial distribution means different distances of different vehicles to the beacon, which causes different power attenuations. If the received signal energies are very dissimilar, i.e., some users are very weak (vehicles far form the beacon) in comparison to the others (vehicles near the beacon), then the conventional single-user detector is unable to recover the messages of the weak users reliably, even if the signature waveforms have very low crosscorrelations. This is known as the *near-far problem* and is the main shortcoming of the system treated in [1],[2].

Thus, we are faced with the problem of trading between the optimum multiuser detector, with the advantage of being able to minimize the probability of error and the disadvantage of having high computational complexity, and the conventional single-user detector with the advantage of being easy to implement and the disadvantage of not being able to handle the near-far problem. We solve this problem by resorting to recently developed *linear decorrelating detectors* [4]–[6], which are practical to implement and, although not optimal, able to handle the near-far problem.

The rest of this chapter is organized as follows: A mathematical model of a general multiuser communication system is given in the first section. Multiuser detectors are described in the second section. Results of the evaluation of the decorrelating and conventional detector performance are given in the third section. The simulation program code is given in the appendix.

B. Multiuser Communication Model

In a CDMA multiuser system [3], the receiver observes signals of the form

$$r(t) = S(t, \boldsymbol{b}) + n(t), \quad t \in \boldsymbol{R},$$

where $n(t)$ is white Gaussian noise with power spectral density σ^2 and

$$S(t, \boldsymbol{b}) = \sum_{i=-M}^{M} \sum_{k=1}^{K} b_k(i)\sqrt{w_k(i)}\tilde{s}_k(t - iT - \tau_k) \tag{1.1}$$

contains the information sequence

$$\boldsymbol{b} = \{\boldsymbol{b}(i) = [b_1(i), \ldots, b_K(i)],\ b_k(i) \in \{-1, 1\},\ k = 1, \cdots, K,\ i = -M, \cdots, M\},$$

with $\tilde{s}_k(t)$ being the normalized signature waveform of user k which is zero outside the interval $[0, T]$, and $w_k(i)$ being the received energy of user k in the i-th time slot.

Assuming all transmitted sequences to be equiprobable, the MAP sequence detector selects the sequence that maximizes

$$P[\{r(t), t \in \boldsymbol{R}\}|\boldsymbol{b}] = C \exp\left[\Omega(\boldsymbol{b})/2\sigma^2\right], \tag{1.2}$$

where C is a positive scalar independent of $\boldsymbol{b}$ and

$$\Omega(\boldsymbol{b}) = 2\int_{-\infty}^{+\infty} S_t(\boldsymbol{b})r(t)dt - \int_{-\infty}^{+\infty} S_t^2(\boldsymbol{b})dt. \tag{1.3}$$

Therefore, the maximum-likelihood sequence detector selects among the possible noise realizations the one with minimum energy.

Using definition (1.1), the first term in the right-hand side of (1.3) can be expressed as

$$\int_{-\infty}^{+\infty} S_t(\boldsymbol{b})r(t)dt = \sum_{i=-M}^{M} \boldsymbol{b}^T(i)\boldsymbol{y}(i), \tag{1.4}$$

where

$$y_k(i) = \int_{\tau_k+iT}^{\tau_k+iT+T} s_k(t-iT-\tau_k)r(t)dt$$

denotes the output of a matched filter for the i-th symbol of the k-th user. Thus the sequence of outputs $\boldsymbol{y}$ of the bank of K matched filters is a sufficient statistic for the selection of the most likely sequence $\boldsymbol{b}$. Therefore, the maximum-likelihood multiuser coherent detector, as shown in Fig. 1, consists of a front end of matched filters (one for each user) followed by a decision algorithm that selects the sequence $\boldsymbol{b}$ that maximizes (1.2) or, equivalently, (1.3).

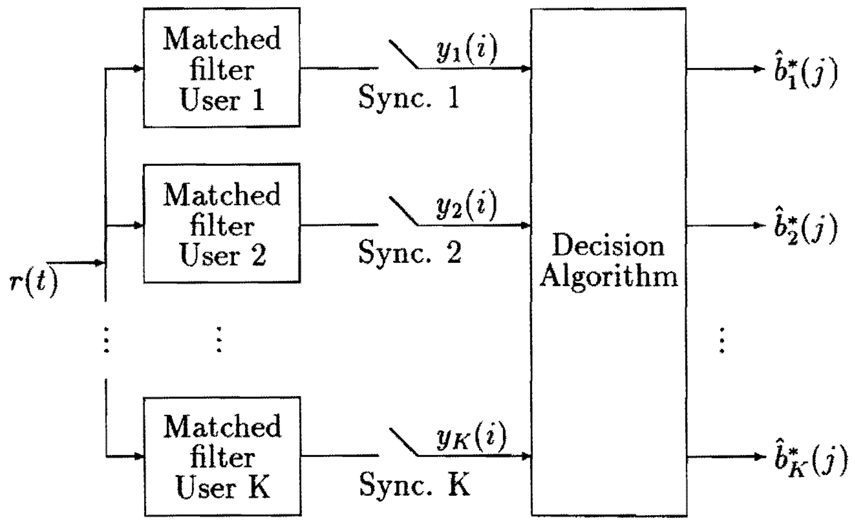


Fig. 1. Optimum K-user detector for asynchronous multiple-access Gaussian channel

The coherent K user receiver commonly employed in practice consists of a bank of optimum single-user detectors operating independently as in Fig. 2. As shown above,

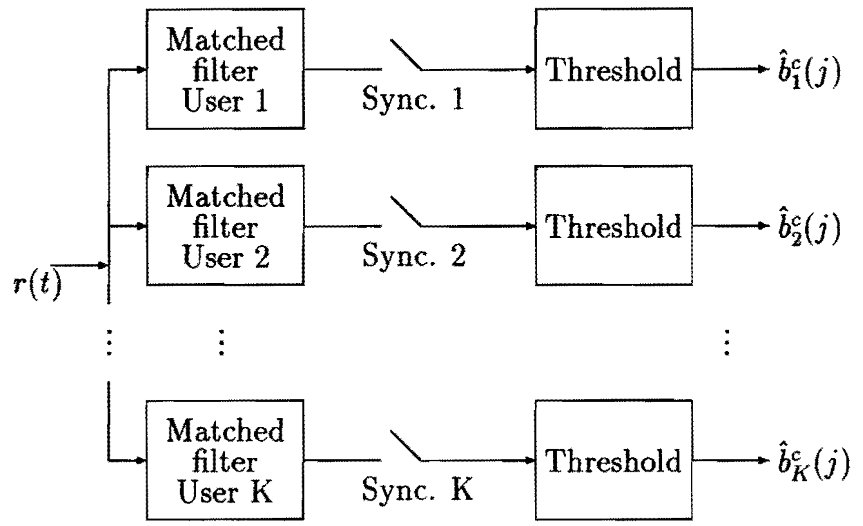


Fig. 2. Conventional K-user detector for asynchronous multiple-access Gaussian channel

the sequence $\boldsymbol{y}$ of outputs of the bank of K matched filters is a sufficient statistic for the selection of the most likely sequence $\boldsymbol{b}$. However, $y_k(i)$ is not a sufficient statistic for the detection of $b_k(i)$. Therefore, the conventional receiver of Fig. 2 is not optimal in terms of error probability. It is, however, easy to implement and, in certain cases such as the system described in [1],[2], achieves acceptable performance.

C. Multiuser Detectors

In choosing a multiuser detector for the road-automobile multiuser communication system described in [1],[2], we are faced with the problem of trading between the optimum multiuser detector, with the advantage of being able to minimize the probability of error and the disadvantage of having high computational complexity, and the conventional single-user detector with the advantage of being easy to implement and the disadvantage of not being able to handle the near-far problem. Our goal is to find a low complexity multiuser detector whose performance lies somewhere in the big gap between those of the optimum and the conventional, and is in particular capable of handling the near-far problem. We resort to recently developed linear decorrelating detectors [4]–[6].

1. Synchronous Case

If the simultaneous users maintain symbol synchronization, the receiver observes the signals of the form

$$r(t) = \sum_{k=1}^{K} b_k(j) s_k(t - jT) + n(t), \quad t \in [jT, jT + T],$$

where, as above, $n(t)$ is again Gaussian noise with power spectral density σ^2 and $\{b_k(j) \in \{-1, 1\}\}$ is the k-th user information sequence. Assuming that all possible sequences are equally likely, an information sequence is detected symbol by symbol. Therefore, we can restrict our analysis to a single symbol interval.

The likelihood function, again, depends on the observations only through the outputs of a bank of matched filters,

$$y_k(i) = \int_0^T s_k(t) r(t) dt, \quad k = 1, \ldots, K.$$

Therefore $\boldsymbol{y} = (y_1, \ldots, y_K)$ are sufficient statistics for demodulating transmitted information bits $\boldsymbol{b} = (b_1, \ldots, b_K)$, on which they depend in the following way:

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{b} + \boldsymbol{n},$$

where $\boldsymbol{H}$ is the nonnegative definite matrix of crosscorrelations between the assigned waveforms whose elements are given by

$$H_{ij} = \int_0^T s_i(t)s_j(t)dt,$$

and $\boldsymbol{n}$ is a K-dimensional vector of zero-mean Gaussian random variables whose covariance matrix is equal to $\sigma^2\boldsymbol{H}$. All multiuser detectors use sufficient statistics $\boldsymbol{y}$ to make the decisions about the transmitted bits $\boldsymbol{b}$, but they differ in the way they process these sufficient statistics.

Conventional single-user detectors ignore the multiuser interference and base their decisions about the transmitted bit of the k-th user solely on the k-th matched filter output as

$$\hat{b}_k^c = \text{sgn } [y_k].$$

In the multiple user radio communication system between vehicles on the highway and roadside beacons [1],[2], the modulation (spectrum-spreading) signals $s_k(t)$ are given by

$$s_k(t) = \sum_{j=0}^{N-1}(2x_k^j - 1)\psi(t - jT_c), \quad 1 \le k \le K,$$

where $\psi(t - jT_c)$ is a pulse (usually called a chip) of duration T_c normalized so that

$$\frac{1}{T_c}\int_0^{T_c} \psi^2(t)dt = 1,$$

and $\boldsymbol{x}_k = x_k^0, x_k^1, \ldots, x_k^N$ are binary sequences of length N. Sufficient statistics y_k are in this system of the form

$$y_k = \frac{1}{N}\sqrt{\frac{w_k}{\sigma}}\left(\phi_{kk} + \sum_{\substack{m=1\\m\neq k}}^{K}\sqrt{\frac{w_m}{w_k}}\phi_{mk}\right) + Z, \tag{1.5}$$

where Z is normalized Gaussian random variable and ϕ_{mk} is the crosscorrelation function of two sequences $\boldsymbol{x}_m$ and $\boldsymbol{x}_k$ defined as

$$\phi_{mk} = \sum_{j=0}^{N-1} (2x_m^j - 1)(2x_k^j - 1).$$

The second term in the brackets of (1.5) represents multiple access interference, which depends on two factors: the crosscorrelation functions of normalized signature waveforms and relative energies of the interfering users. In the spread-spectrum road-automobile communication system described in [1],[2], low crosscorrelations of signature waveforms were achieved by employing Gold sequences of length N for the PN sequences $\boldsymbol{x}_k$, and the assumption was made that the power levels received by the beacon from all the simultaneous users (vehicles) were equal. The performance of the conventional single user detector for this system were satisfactory [1],[2]. However, with sufficiently high interference energy, performance of the conventional detector degrades to zero, i.e., the error probability of detecting the k-th user bits increases to $1/2$ as the energies of the interfering users w_m, $m = 1, \ldots, K$, $m \neq k$, increase compared to the energy of the k-th user w_k. This is proven in [4]. Later we give some quantitive results obtained by simulation for certain practical cases treated in [1],[2].

The optimal multiuser detector selects the most likely sequence $\hat{\boldsymbol{b}}^* = (\hat{b}_1^*, \ldots, \hat{b}_K^*)$ as the one that corresponds the noise realization with minimum energy, i.e.,

$$\begin{aligned} \hat{\boldsymbol{b}}^* &= \arg \min_{\boldsymbol{b} \in \{-1,1\}^K} \int_0^T \left[r(t) - \sum_{k=1}^{K} b_k s_k(t) \right]^2 dt \\ &= \arg \min_{\boldsymbol{b} \in \{-1,1\}^K} 2\boldsymbol{y}^T \boldsymbol{b} - \boldsymbol{b}^T \boldsymbol{H} \boldsymbol{b}. \end{aligned}$$

This detector significantly outperforms the conventional single-user detector in the presence of near-far interference. However, no algorithm that solves the above minimization problem in polynomial time in K is known, which makes the optimal multiuser detector impractical.

Recently a certain class of multiuser detectors capable of handling the near-far problem was introduced [4]. This is the class of linear detectors termed *decorrelating detectors.* The decorrelating detectors detect the information bits based on the sufficient statistics in the following way:

$$\hat{\boldsymbol{b}} = \operatorname{sgn} \left[\boldsymbol{H}^I \boldsymbol{y}\right],$$

where $\boldsymbol{H}^I$ is a generalized inverse of the crosscorrelation matrix $\boldsymbol{H}$.

2. Asynchronous Case

So far we have considered multiuser systems in which the simultaneous users maintain symbol synchronization, and would like to generalize the decorrelating receiver to the asynchronous multiuser systems in which the receiver observes signals of the form

$$r(t) = \sum_{i=-M}^{M} \sum_{k=1}^{K} b_k(i)\sqrt{w_k(i)}\tilde{s}_k(t - iT - \tau_k) + n(t). \tag{1.6}$$

A straight-forward approach would be to view each of the $(2M+1)K$ symbols (transmitted by K users during $(2M+1)$ signaling intervals) as transmitted by a different user [5], and thus the K user asynchronous channel as $(2M+1)K$ synchronous channel. However, the corresponding detector would be impossible to implement since it would involve the inversion of a crosscorrelation matrix of dimension $(2M+1)K$. Fortunately, as the length of the transmitted sequence increase, the decorrelating detector approaches the K-input K-output linear time-invariant filter with transfer function

$$G(z) = [R^T(1)z + R(0) + R(1)z^{-1}]^{-1},$$

where $R(l)$, $l = 0, 1$, are $K \times K$ normalized signal crosscorrelation matrices whose entries are given by

$$R_{ij} = \int_0^T s_i(t - \tau_i) s_j(t + lT - \tau_j) dt, \quad l = 0, 1.$$

The transfer function $G(z)$ is non-causal and in practice it would be necessary to approximate it by truncation.

An alternative to the decorrelating detector for asynchronous channels can be obtained by taking a one-shot approach where each symbol interval is considered separately [6]. Suppose, without loss of generality, we wish to detect bit 0 of user k, which is send in the time interval $[0, T]$ (assuming again without loss of generality $\tau_k = 0$). This bit overlaps with bit -1 of user m over the interval $[0, \tau_m]$ and with bit 0 of user m over the interval $[\tau_m, T]$, $m = 1, \ldots, K$, $m \neq k$. We can view this situation as a $(2K+1)$-user synchronous channel (see Fig. 3) with unit energy signature waveforms $\tilde{s}_k(t) = s_k(t)$, $\tilde{s}_m(t) = e_m^{-1/2} s_m^L(t)$, $\tilde{s}_m(t) = (1 - e_m)^{-1/2} s_m^R(t)$, $m = 1, \ldots, K$, $m \neq k$, where

$$s_k^L(t) = \begin{cases} s_k(t + T - \tau_k) & 0 \leq t \leq \tau_k \\ 0 & \tau_k \leq t \leq T \end{cases} \qquad s_k^L(t) = \begin{cases} s_k(t + T - \tau_k) & 0 \leq t \leq \tau_k \\ 0 & \tau_k \leq t \leq T \end{cases}$$

and $e_m = \int_0^{\tau_m} s_m^2(t + T - \tau_m) dt$ is the partial energy of interfering signal m, $m = 1, \ldots, K$, $m \neq k$, over the left overlapping interval.

This one-shot decorrelation detector has lower complexity than the asynchronous decorrelating detector described above. Possible utilization of this detector in the spread-spectrum road-automobile communication system described in [1],[2] is investigated by testing the detector performance by simulation. The results of the simulation are presented and discussed in the next section.

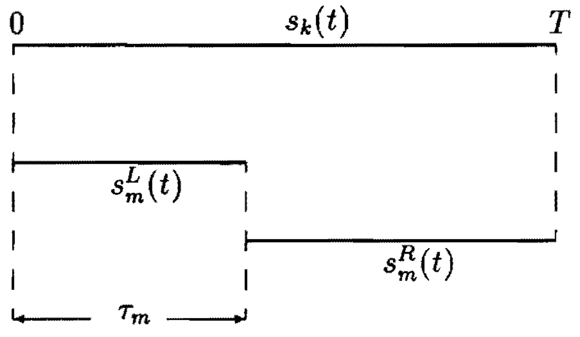


Fig. 3. Intervals for one-shot decorrelating detector

D. System Performance Evaluation

We use the Gold sequences [1],[2] for the signature waveforms, and test the error probability performance of the conventional and the linear detector for different ratios of energies of the interfering users by simulation. The results of the simulation are given in Fig. 4 for 31-length Gold sequences and two simultaneous users, in Fig. 5 for 31-length Gold sequences and six simultaneous users, and in Fig. 6 for 127-length Gold sequences and eight simultaneous users.

From these figures we can see that the error-probability performance of the conventional detector depends strongly on the ratio of energies of the interfering active users, decaying rapidly with increasing interference until the receiver becomes practically multiple-access limited. On the other hand, the error probability performance of the decorrelating detector is invariant to the energy of the interfering users.

In [1],[2] we showed that the bit error probability in CDMA systems that use conventional detectors are due to two factors: presence of the noise in the channel and presence of the multiple users in the same channel. The effect of the first factor is decreased by increasing signal to noise ration (SNR), which means increasing required power. The effect of the second factor is decreased by increasing the length of the used PN-sequences, which means increasing required bandwidth. For a given SNR, as the number of simultaneous users increases the bit-error probability increases. For a given number of simultaneous users, as the SNR increases the bit-error probability decreases down to a certain value determined by the number of the users. This means that the conventional multiuser detector becomes practically multiple-access limited with increasing number of the interfering users, even when they have the same energy as the intended one. Figure 5 shows that, with 31-length Gold sequences, it is not possible to support 6 equal energy users by using the conventional detector, but it is possible by using the decorrelating detector, which is in addition near-far resistant.

Based on the simulation results, we conclude that it is possible to handle the near-far problem by employing these more complex receivers. We also conclude that these receivers, besides being capable of handling the near-far problem, exhibit better performance in terms of the required bandwidth. This puts a new prospect to the optimization problem [1],[2] which is concerned with *transmitting as much information as possible to as many vehicles as possible as accurately as possible using as low amount of power as possible,* which is to be investigated further.

REFERENCES

[1] "Telecommunication Requirements for IVHS," *Final Report,* Texas Transprotation Institute, August 1990.

[2] E. Soljanin, W. Wills, and C. N. Georghiades, "Spread-Spectrum Raod-Automobile Communications," presented on *1991 American Control Conference,* Boston, MA, June 1991.

[3] S. Verdú, "Minimum probability of error for asynchronous Gaussian multiple-access channels," *IEEE Transactions on Information Theory,* vol. IT-32, pp. 85–96, January, 1986.

[4] R. Lupas, S. Verdú, "Linear multiuser detectors for synchronous code-division multiple-access channels," *IEEE Transactions on Information Theory,* vol. IT-35, pp. 123–136, January, 1989.

[5] R. Lupas, S. Verdú, "Near-far Resistance of multiuser detectors in asynchronous channels," *IEEE Transactions on Communications,* vol. IT-38, pp. 496–508, April, 1990.

[6] S. Verdú, "Recent progress in multiuser detection" *Lecture Notes in Control and Information Sciences Series, Springer-Verlag, Advances in Communication & Control Systems, ComCon 88,* pp. 66–77, Baton Rouge, LA, October, 1988.

APPENDIX A

SIMULATION PROGRAM CODE

```
#include <stdio.h>
#include <math.h>

#define N 127
#define m 7
#define Nu 8
#define ns 4

#define snrl 0
#define snru 21
#define I 100000

#define pi 3.14159265

double drand48();
double erfc(double x);
void srand48(long seed);
double pow(double x, double y);

int sb, rbc, rbl;

unsigned long as[ns+1];
unsigned long bs[ns+1];
long gs[N+5][ns+1];
int c_rr[Nu][Nu][N+1][N+1];
int c_lr[Nu][Nu][N+1][N+1];
int c_rl[Nu][Nu][N+1][N+1];
int c_ll[Nu][Nu][N+1][N+1];
double nc[N], nu[2*Nu-1];
int b[2*Nu];
double H[2*Nu][2*Nu];
double y[2*Nu], e[Nu];
int d[Nu];

inv_m(a,n)
double a[2*Nu][2*Nu];
int n;
{
int jr, jc, jp, j;

for(jp = 0; jp < n; jp++)
{
for(jr = 0; jr < n; jr++)
if(jr != jp)
for(jc = 0; jc < n; jc++)
if(jc != jp)
a[jr][jc] = a[jr][jc] - a[jr][jp]*a[jp][jc]/a[jp][jp];

a[jp][jp] = -1./a[jp][jp];
for(j = 0; j < n; j++)
if(j != jp)
{
a[j][jp] = a[j][jp]*a[jp][jp];
a[jp][j] = a[jp][j]*a[jp][jp];
}
}
for(jr = 0; jr < n; jr++)
{
for(jc = 0; jc < n; jc++)
{
a[jr][jc] = -a[jr][jc];
}
}
}

bitcounts(p, s1, s2)
unsigned long p[ns+1];
int s1, s2;
{
int b, bs;

rshift(p,s1);
b = 0;
for (bs = 0; bs < s2-s1; bs++)
{
if (p[0]&1)
b++;
rshift(p,1);
}
return (b);
}

bitcount1(p)
unsigned long p;
{
int b;
unsigned long pb;

pb = (p<<1)>>1;
for (b = 0; pb != 0; pb >>=1)
if (pb & 01)
b++;
return (b);
}
```

```
bitcount(p)
unsigned long p;
{
int b;

for (b = 0; p != 0; p >>=1)
if (p & 01)
b++;
return (b);
}

void coder(cs, fbc)
int cs[ns+1];
int fbc;
{
int srp = 1;
int mask;
int i, j;

mask = (N+1)/2;
for (j = 0; j < ns; j++)
{
cs[j] = 0;
for (i = 0; i < 32; i++)
{
cs[j] = cs[j] | (((srp&mask)>>(m-1)) << i);
srp = ((srp<<1) | (bitcount(srp&fbc)&1))&N;
}
}
}

void pn()
{
unsigned long z[ns+1];
unsigned long zjl[ns+1], zkl[ns+1];
unsigned long zjr[ns+1], zkr[ns+1];
int i, j;
int ju, ku, js, ks;
int f, s;

coder(as,72);
coder(bs,120);

for (j = 0; j < ns; j++)
{
z[j] = bs[j];
gs[N][j] = as[j];
gs[N+1][j] = bs[j];
}
```

```
z[ns] = z[0];

for (i = 0; i < N; i++)
{
for (j = 0; j < ns; j++)
gs[i][j] = as[j] ^ z[j];

for (j = 0; j < ns-1; j++)
z[j] = ((z[j+1]&1)<<31) | (z[j]>>1);

z[ns-1] = ((z[ns]&1)<<30) | (z[ns-1]>>1);
z[ns] = z[0];
}
for (j = 0; j < Nu; j++)
{
for (s = 0; s < ns; s++)
z[s] = gs[j][s];
for (i = 0; i < N; i++)
{
printf("%d ", (z[ns-1]&0x40000000)>>30);
lshift(z,1);
}
printf("\n");
}

printf("pocetak\n");
for (ju = 0; ju < Nu; ++ju)
for (ku = 0; ku < Nu; ++ku)
{
for (js = 0; js < N; ++js)
for (ks = 0; ks < N; ++ks)
{
for(s = 0; s < ns; ++s)
{
zjl[s] = gs[ju][s];
zjr[s] = gs[ju][s];
zkl[s] = gs[ku][s];
zkr[s] = gs[ku][s];
}
lshift(zjl,js);
rshift(zjr,js);
lshift(zkl,ks);
rshift(zkr,ks);
c_rr[ju][ku][js][ks] = cor_rr(zjr,zkr,js,ks);
c_rl[ju][ku][js][ks] = cor_rl(zjr,zkl,js,ks);
c_lr[ju][ku][js][ks] = cor_lr(zjl,zkr,js,ks);
c_ll[ju][ku][js][ks] = cor_ll(zjl,zkl,js,ks);
}
}
printf("kraj\n");
printf("%d\n", c_rr[0][1][0][13]);
```

```
for(s = 0; s < ns; ++s)
{
zkr[s] = gs[0][s];
zjr[s] = gs[1][s];
}
rshift(zjr,13);
printf("%d\n", cor_rr(zkr,zjr,0,13));
}

int cor_rr(z1, z2, i1, i2)
unsigned long z1[ns+1], z2[ns+1];
int i1, i2;
{
int f, s, j1, j2;
unsigned long z[ns];

for(s = 0; s < ns; ++s)
z[s] = z1[s]^z2[s];

f = 0;
if (i1 < i2) j2 = N-i2; else j2 = N-i1;
j1 = 0;
f = (j2-j1) - 2*bitcounts(z,j1,j2);
return(f);
}

int cor_rl(z1, z2, i1, i2)
unsigned long z1[ns+1], z2[ns+1];
int i1, i2;
{
int f, s, j1, j2;
unsigned long z[ns];

for(s = 0; s < ns; ++s)
z[s] = z1[s]^z2[s];

f = 0;
if ((N-i1) > i2)
{
j1 = i2;
j2 = N-i1;
f = (j2-j1) - 2*bitcounts(z,j1,j2);
}
return(f);
}

int cor_lr(z1, z2, i1, i2)
unsigned long z1[ns+1], z2[ns+1];
int i1, i2;
{
int f, s, j1, j2;
```

```
unsigned long z[ns];

for(s = 0; s < ns; ++s)
z[s] = z1[s]^z2[s];

f = 0;
if ((N-i2) > i1)
{
j1 = i1;
j2 = N-i2;
f = (j2-j1) - 2*bitcounts(z,j1,j2);
}
return(f);
}

int cor_ll(z1, z2, i1, i2)
unsigned long z1[ns+1], z2[ns+1];
int i1, i2;
{
int f, s, j1, j2;
unsigned long z[ns];

for(s = 0; s < ns; ++s)
z[s] = z1[s]^z2[s];

if (i1 < i2) j1 = i2; else j1 = i1;
j2 = N;
f = (j2-j1) - 2*bitcounts(z,j1,j2);
return(f);
}

lshift(ws, nsh)
unsigned long ws[ns+1];
int nsh;
{
int k, j;
if (nsh < N){
for (k = 0; k < nsh; ++k)
{
if(ns != 1)
for (j = ns-1; j > 0; j--)
ws[j] = ((ws[j-1]&0x80000000)>>31) | (ws[j]<<1);
ws[0] = ws[0]<<1;
}
}
}

rshift(ws, nsh)
unsigned long ws[ns+1];
int nsh;
{
```

```
int k, j;
if (nsh < N){
for (k = 0; k < nsh; ++k)
{
for (j = 0; j < ns-1; j++)
ws[j] = ((ws[j+1]&1)<<31) | (ws[j]>>1);

ws[ns-1] = ws[ns-1]>>1;
}
}
}

cshift(ws, nsh)
unsigned long ws[ns+1];
int nsh;
{
int k, j;
for (k = 0; k < nsh; ++k)
{
for (j = 0; j < ns-1; j++)
ws[j] = ((ws[j+1]&1)<<31) | (ws[j]>>1);

ws[ns-1] = ((ws[ns]&1)<<30) | (ws[ns-1]>>1);
ws[ns] = ws[0];
}
}

void channel(s)
double s;
{
unsigned long z[ns+1];
unsigned long zp[ns+1];
int i, j, k;
int ju, ku;
int dj, dk;
int sg, sh;

for (j = 0; j < 2*Nu-1; j++)
{
sb = 2*drand48();
if (sb == 0) b[j] = -1;
else b[j] = 1;
}
for (j = 1; j < Nu; j++)
d[j] = (N-1)*drand48()+1;
d[0] = 0;

H[0][0] = N;
for (ku = 1; ku < Nu; ++ku)
{
dk = d[ku];
```

```
H[0][ku] = e[0]*e[ku]*c_rr[0][ku][0][dk];
H[0][Nu-1+ku] = e[0]*e[ku]*c_rl[0][ku][0][N-dk];
H[ku][0] = e[ku]*e[0]*c_rr[0][ku][0][dk];
H[Nu-1+ku][0] = e[ku]*e[0]*c_rl[0][ku][0][N-dk];
}
for (ju = 1; ju < Nu; ++ju)
{
dj = d[ju];
for (ku = 1; ku < Nu; ++ku)
{
dk = d[ku];
H[ju][ku] = e[ju]*e[ku]*c_rr[ju][ku][dj][dk];
H[ju][Nu-1+ku] = e[ju]*e[ku]*c_rl[ju][ku][dj][N-dk];
H[Nu-1+ju][ku] = e[ju]*e[ku]*c_lr[ju][ku][N-dj][dk];
H[Nu-1+ju][Nu-1+ku] = e[ju]*e[ku]*c_ll[ju][ku][N-dj][N-dk];
}
}
noise();
for (j = 0; j < 2*Nu-1; j++)
y[j] = nu[j]*b[j];

for (j = 0; j < 2*Nu-1; j++)
{
for (k = 0; k < 2*Nu-1; k++)
y[j] = y[j] + s*H[j][k]*b[k];
}
inv_m(H,2*Nu-1);
}

noise()
{
int ju, jc, s;
double a, b;
unsigned long zl[ns+1], zr[ns+1];

for(jc = 0; jc < N; ++jc)
{
a = drand48();
b = drand48();
nc[jc] = sqrt(-2*log(a))*cos(2*pi*b)*sqrt(N*1.);
}
for(ju = 0; ju < Nu; ++ju)
{
for(s = 0; s < ns; ++s)
{
zl[s] = gs[ju][s];
zr[s] = gs[ju][s];
}
rshift(zr,d[ju]);
lshift(zl,N-d[ju]);
nu[ju] = 0.;
```

```
for(jc = 0; jc < N-d[ju]; ++jc)
{
s = 2*(zr[0]&1) - 1;
nu[ju] = nu[ju] + s*nc[jc];
rshift(zr,1);
}
}
for(ju = 1; ju < Nu; ++ju)
{
nu[Nu-1 + ju] = 0.;
for(jc = 0; jc < d[ju]; ++jc)
{
s = 2*((zl[ns-1]&0x40000000)>>30) - 1;
nu[Nu-1 + ju] = nu[Nu-1 + ju] + s*nc[N-1-jc];
lshift(zl,1);
}
}
}

main()
{
int k, j, snrd;
double snr;
double d, pec, pel;
unsigned long z[ns+1];
int s;

e[0] = 1.;
for (j = 1; j < Nu; j++)
e[j] = j*j/8.;
pn();

for (snrd = snrl; snrd < snru; snrd++)
{
srand(1);
snr = pow(10,snrd/10.);
snr = sqrt(2*snr);
pel = 0;
pec = 0;
for (j = 0; j < I; ++j)
{
channel(snr);
d = 0.;
for(k = 0; k < 2*Nu-1; ++k)
d = d + H[0][k]*y[k];
sb = b[0];
if (d < 0) rbl = -1;
else rbl = 1;
if (y[0] < 0) rbc = -1;
else rbc = 1;
if (sb != rbl) pel = pel + 1;
if (sb != rbc) pec = pec + 1;
}
printf("%d,", snrd);
printf("%f\n", pel/I);
printf("%d,", snrd);
printf("%f\n", pec/I);
/* printf("%f\n", 0.5*erfc(snr/sqrt(2.))); */
}
}
```

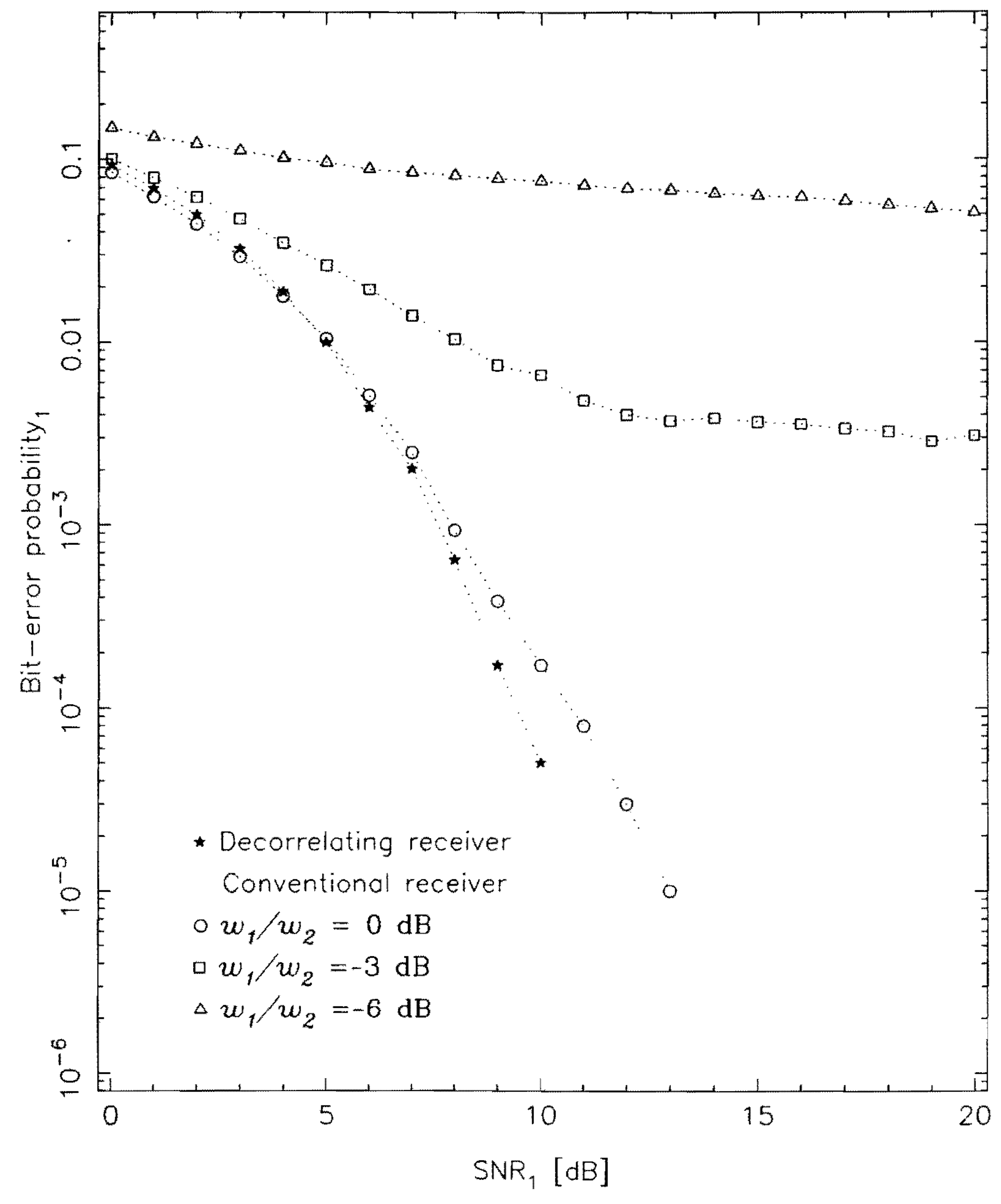


Fig. 4. Probability of error of user 1 in a CDMA system using 31-length Gold sequences, with two active users having energies w_1 and w_2, for the decorrelating and conventional receiver.

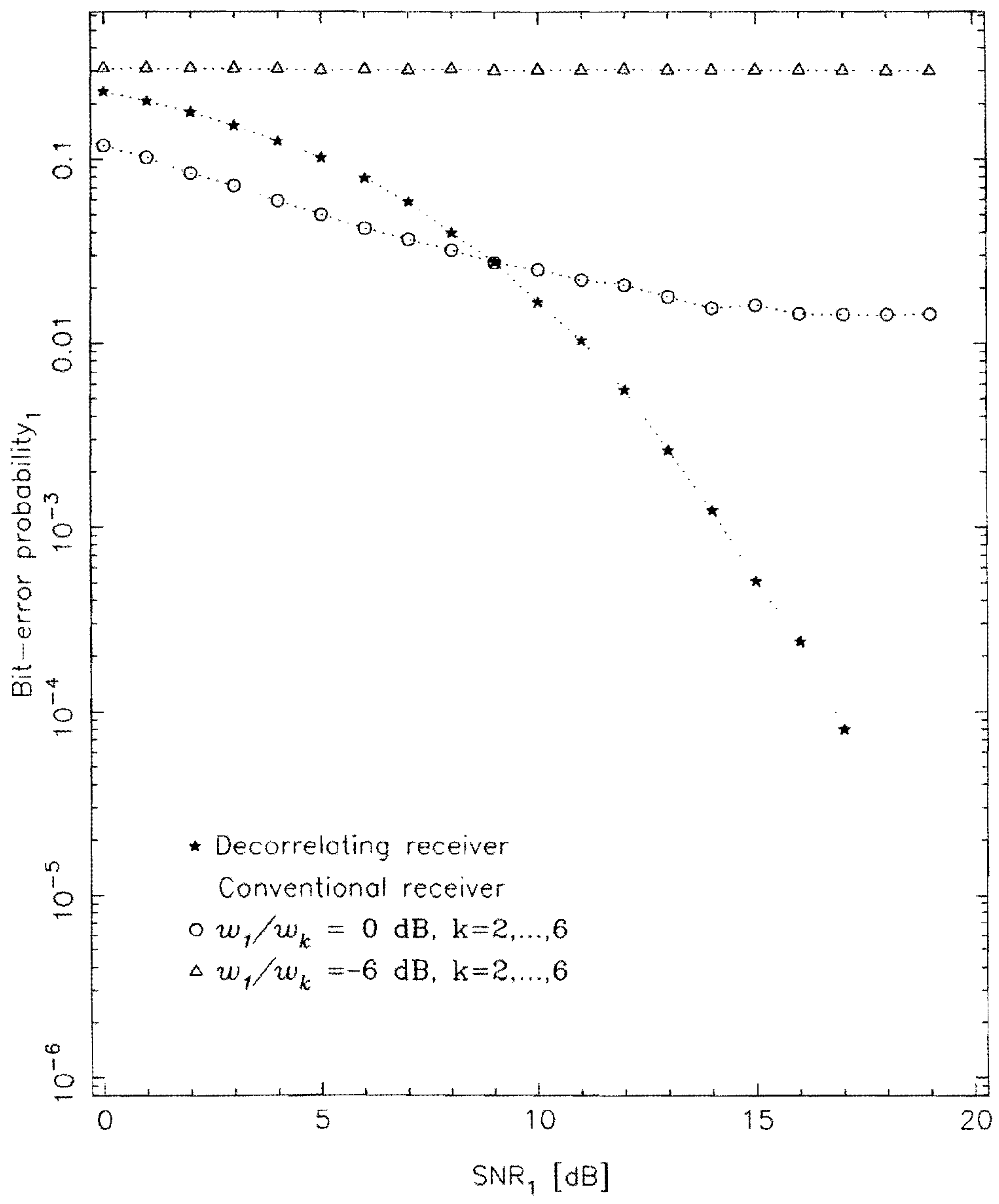


Fig. 5. Probability of error of user 1 in a CDMA system using 31-length Gold sequences, with six active users having energies $w_k, k = 1, \ldots 6$, for the decorrelating and conventional receiver.

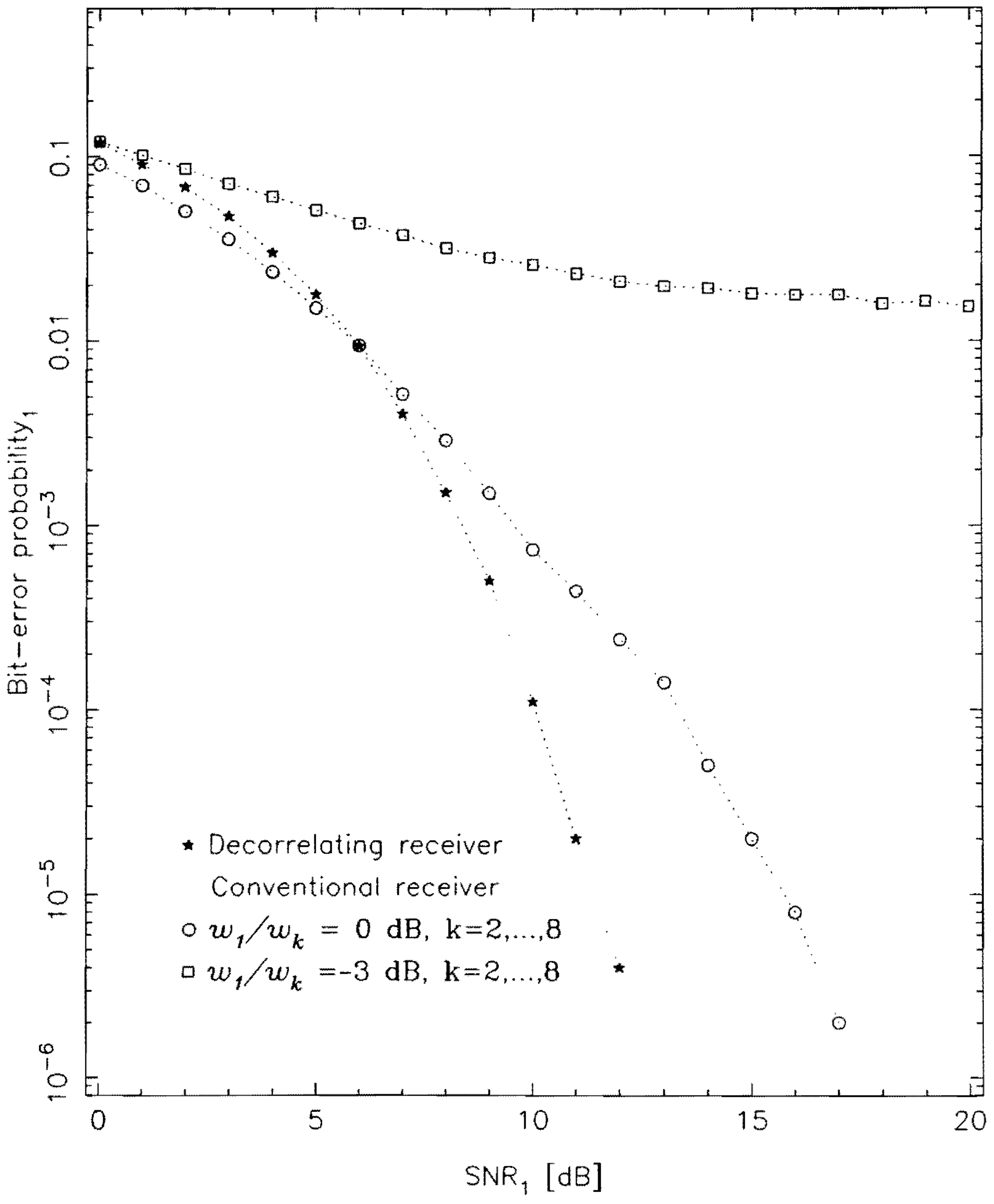


Fig. 6. Probability of error of user 1 in a CDMA system using 127-length Gold sequences, with eight active users having energies $w_k, k = 1, \ldots 8$, for the decorrelating and conventional receiver.

CHAPTER II

SIMULATION PROTOCOL FOR SPREAD-SPECTRUM VEHICLE COMMUNICATION

A. Introduction

For the problem of communication between vehicles on a highway and road-side beacons using spread spectrum techniques, the number of vehicles capable of transmitting at the same time is very important and influences directly the performance of the system. In order to obtain practical results for this value, a program simulating vehicle flow on a highway has been created (Appendix ??) that determines for a specific traffic, the average number of vehicles able to communicate with a beacon.

Another important problem for spread spectrum communication is the assignment of spreading sequences. In the case of communications between vehicles and beacons, each vehicle must have a unique sequence when it communicates. To control this sequence assignment, a protocol to access and quit the beacon has been designed and implemented jointly with the highway simulation program in order to study its performance.

B. Highway Simulation

1. Introduction

The program simulating a highway is based on a moving flux of vehicles. Each vehicle is created with a certain identification and a certain speed. The generation frequency of those vehicles is directly related to the traffic density chosen for the simulation. Also, once a vehicle is added in the highway, it moves every Δt seconds to a new location according to its original speed. During the simulation, a special region of the

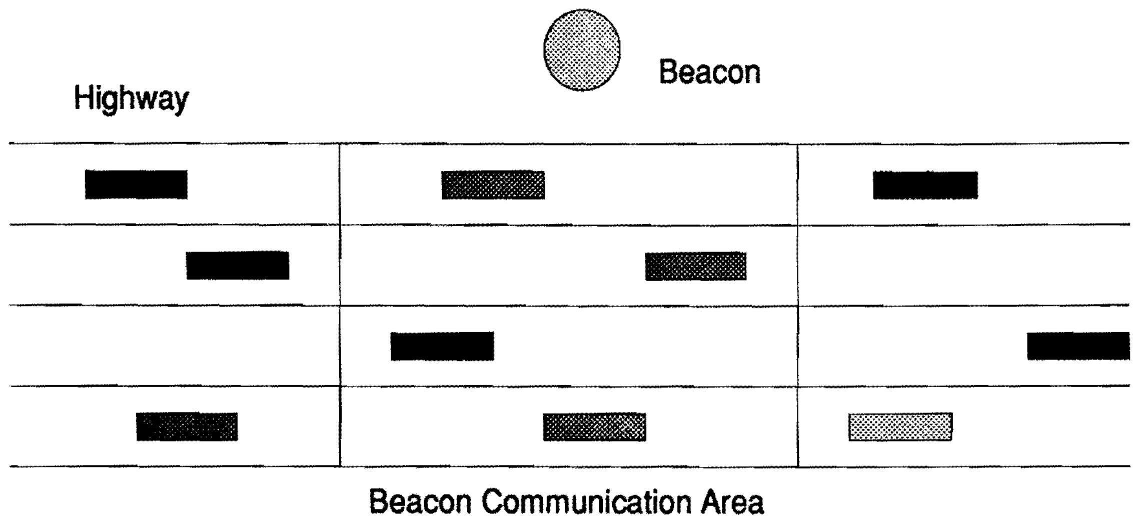


Fig. 7.

highway representing the beacon transmission area is continuously checked in order to compute the desired results.

2. Simulation results

In order to have realistic results some parameters are constant while others vary in a specific range. We considered in the simulation a four-lane highway, kept the length of the vehicles constant at 5 meters and the width of the beacon transmission area at 200 meters while the speed of the vehicles varied from 90 km/h to 120 km/h (resp. from 55 MPH to 75 MPH) and the traffic from 3000 vehicles/hour to 15000 vehicles/hour. Moreover, in order to compute accurate results, the simulation program simulated a total time of 10000 seconds with moving intervals of 0.1 seconds.

Fig. 8 shows the average number of vehicles in the beacon communication area and its standard deviation as a function of the traffic density. We notice that, as expected, the average number of vehicles linearly increases with the traffic from 5 vehicles at 3000 vehicles/hour to 27 vehicles at 15000 vehicles/hour. These results do

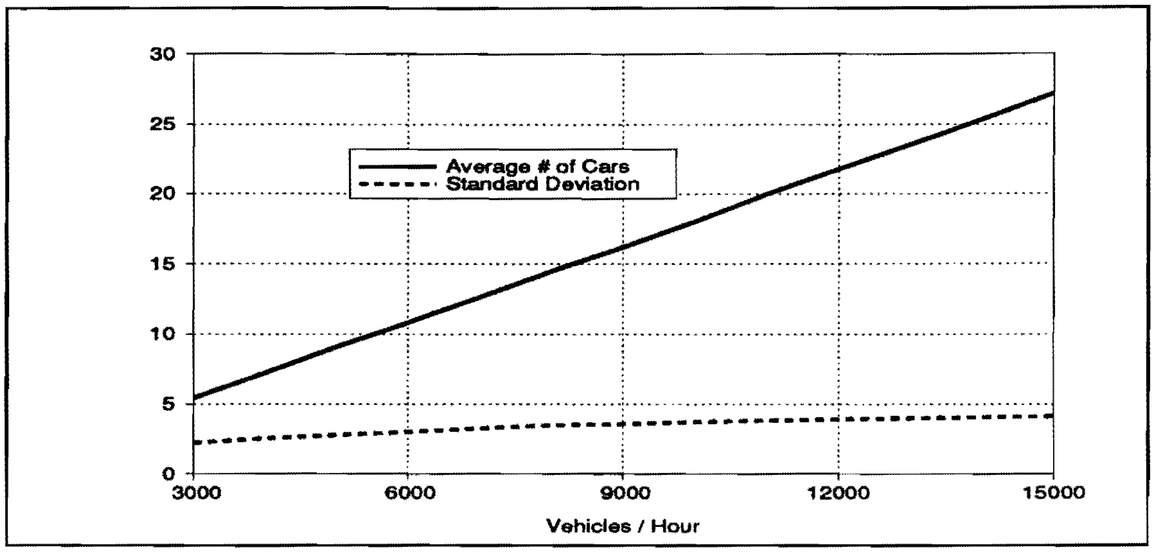


Fig. 8.

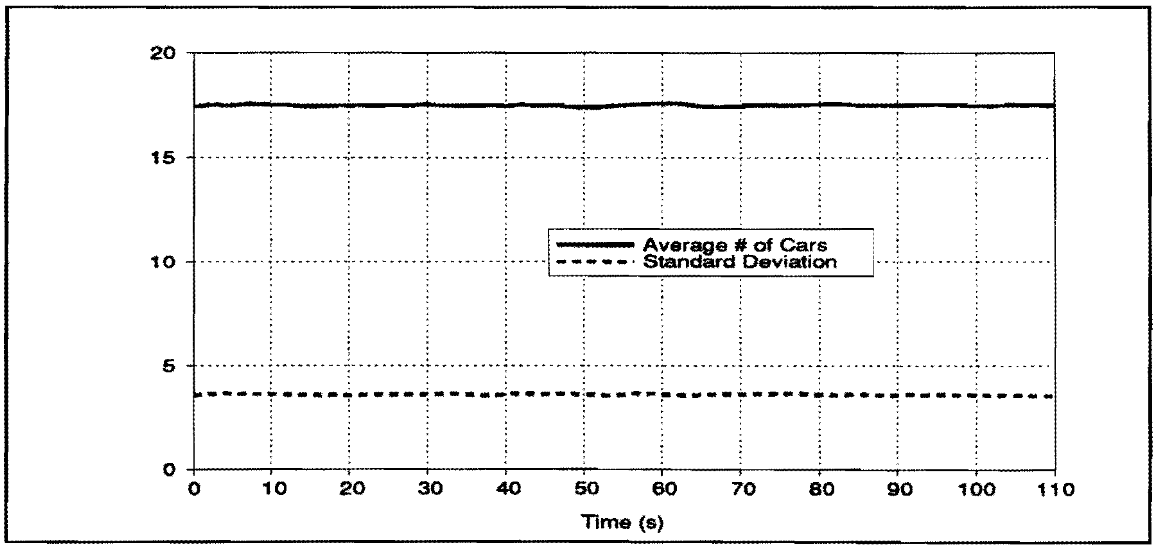


Fig. 9. Simulation for 9200 Vehicles / Hour

not vary as a function of time and stay constant for a specific traffic, as can be seen in Fig. 9.

3. Comparison with Queuing Theory

Since we have considered the highway as a continuous flux of vehicles and we are interested in the number of vehicles in the beacon area, queuing theory can be used in order to compute this result. Thus, if we consider the beacon area as a queue where the vehicles arrive at the rate of λ vehicles per second and where the service time is the time during which the vehicles clear the beacon area, using Little's formula

$$E[N] = \lambda E[T]$$

where

E[N] = Average number of vehicles in the beacon area,

λ = Arriving rate,

E[T] = Average service time,

we can obtain the average number of vehicles in the beacon area.

Table I shows the average number of vehicles in the beacon area by simulation and by use of Little's formula for different traffic densities. These results being very similar, we can consider the simulation program as a realistic model for further applications.

C. Protocol

The goal of this protocol is to allow the vehicles to communicate safely (with no interference) with the beacon. Since we are using spread spectrum techniques, each

Table I. Comparison between simulation and Little's formula

Traffic	Little's formula	Simulation
3000	5.4285	5.4358
4000	7.2380	7.2022
5000	9.0476	9.0406
6000	10.8571	10.8012
7000	12.6666	12.6370
8000	14.4761	14.4848
9000	16.2857	16.2116
10000	18.0952	18.0182
11000	19.9047	19.9855
12000	21.7142	21.7655
13000	23.5238	23.5248
14000	25.3333	25.3368
15000	27.1428	27.1642

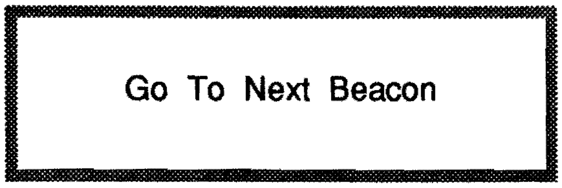
vehicle must use a different sequence to communicate. Thus, the protocol must be able to assign different sequences to users with the smallest delay and with the maximum reliability.

Status :

1 - The beacon decodes only a certain number among all possible sequences.

2 - The protocol assumes that the beacon and the vehicles are perfectly synchronized

3 - Two specific sequences are reserved for the protocol. They will be called seq0 and seq1. seq0 is used to communicate from the vehicle to the beacon while seq1 is used by the beacon to communicate with moving vehicles. Also, when the beacon is saturated (when the maximum number of users it achieved), a special frame "Go to next beacon" is sent with seq1 that indicates to the vehicles they cannot access the beacon.



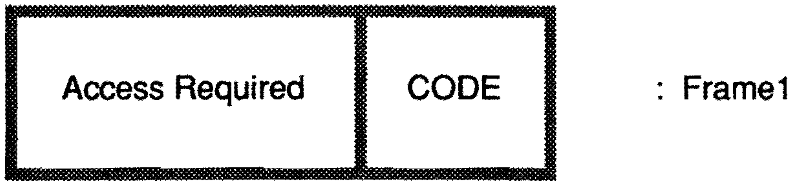
Initialization of the Transmission :

Vehicle :

Step 1 : Check seq1 to verify if the beacon can accept another user

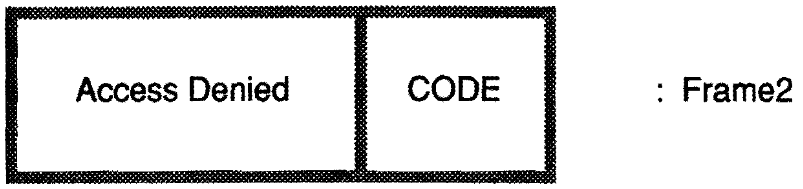
Step 2 : If the received frame is the frame "Go to next beacon" then the vehicle cannot access this beacon and must wait until the next beacon. If not, then the vehicle assumes that the beacon is not saturated

Step 3 : Send Frame1. This frame is composed of a command : "Access Acquired" and a recognition code. This code is a personal code and is unique to each vehicle. It is used to identify a vehicle during the sequence assignment.



Step 4 : The vehicle must wait for the response from the beacon by decoding seq1. At the same time a delay clock is initialized in order to restart the access in case of no response.

Step 5 : If the response is Frame2 then a request collision occurred. In this case, we need to generate a random delay T1 before retrying to access the beacon from Step 1.



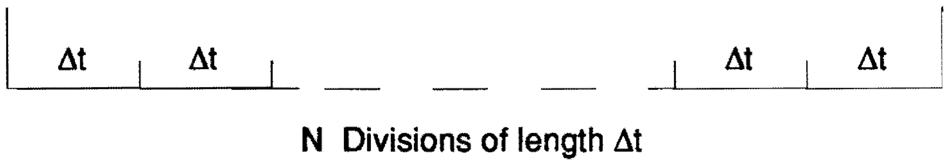
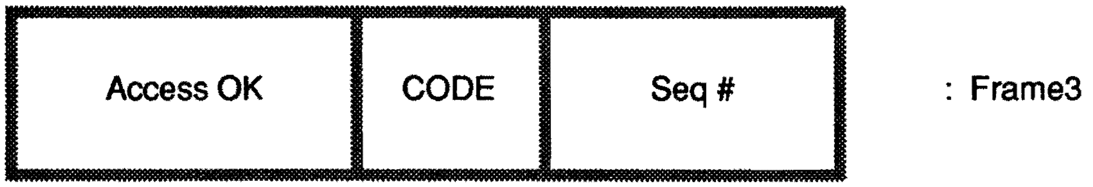


Fig. 10.

Step 6 : If the response is Frame3 with the correct identification code then we can start the communication by using the sequence indicated in the frame. The assignment of this sequence is controlled by the beacon which knows which sequences are already used and which are available for new arrivals.



Step 7 : If nothing happens up to a certain delay T2 then generate a random delay T1 and retry to access the beacon from Step 1. This delay T2 is checked with the delay clock initialized in Step 4.

The delay T1 is generated after a request collision. This delay is computed by generating a random integer in the interval [0,N] and by multiplying this value by Δt which corresponds to the maximum time needed for an initialization frame (Frame1) to go from a vehicle to the beacon (Fig. 10).

The delay T2 corresponding to the time of no reception of information can be computed by using the maximum time needed by an initialization frame to go from the vehicle to the beacon, be analyzed by the beacon, and then return to the vehicle.

Beacon :

Step 1 : If the maximum number of users is present, send the frame "Go to next beacon" with seq1. This frame informs the vehicles that they cannot access the beacon. It is mainly used when the beacon has reached its maximum number of users (vehicles), but can also be used when the beacon is momentanily out of order and cannot decode the different sequences.

If we assume the beacon is not saturated, then :

Step 2 : The beacon waits for Frame1 ("Acess Required") on seq0 which is the unique sequence to access the beacon.

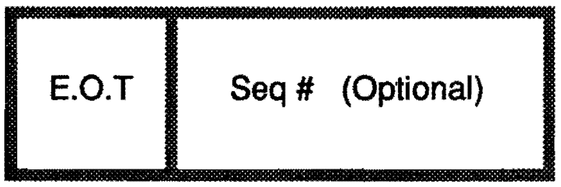
Step 3 : If the received frame is different than Frame1, then we know that a request collision occurred and the beacon sends Frame2 to signal this problem to the vehicles which tried to access the beacon.

Step 4 : If the received frame is Frame1 then the beacon sends Frame3 with an available sequence. This frame has a command field "Access OK", a sequence field which contains the sequence assigned by the beacon and a code field which contains the same code than in frame1 sent by the vehicle. This code being unique allows the beacon to give a specific sequence to a specific vehicle.

End of Transmission :

Vehicle :

1 - Send Frame4



: Frame4

E.O.T : End Of Transmission

The sequence number in frame4 is optional because the beacon already knows which sequence is used to decode this frame.

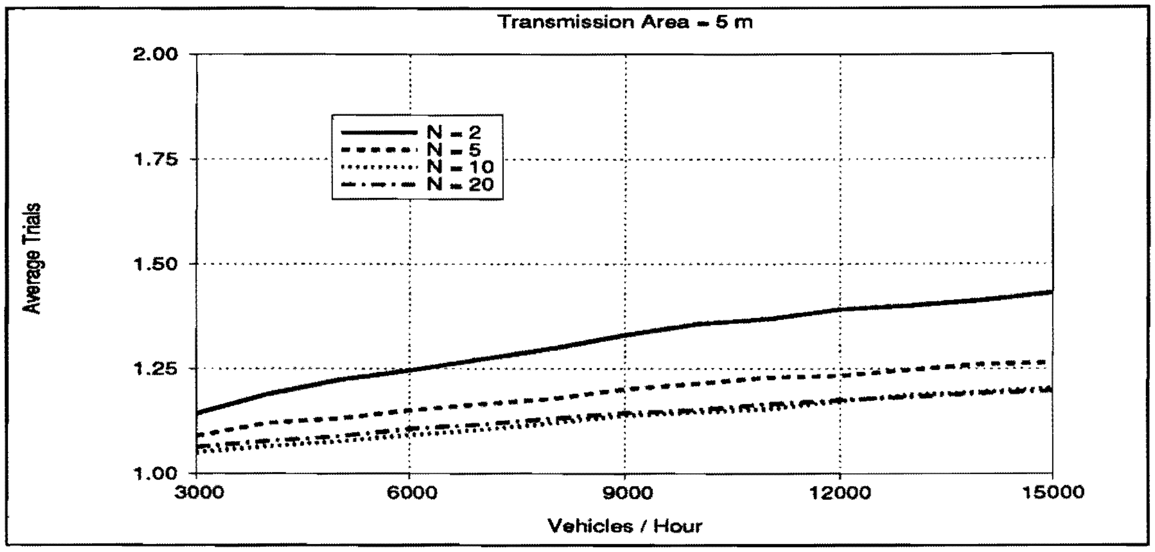
Beacon :

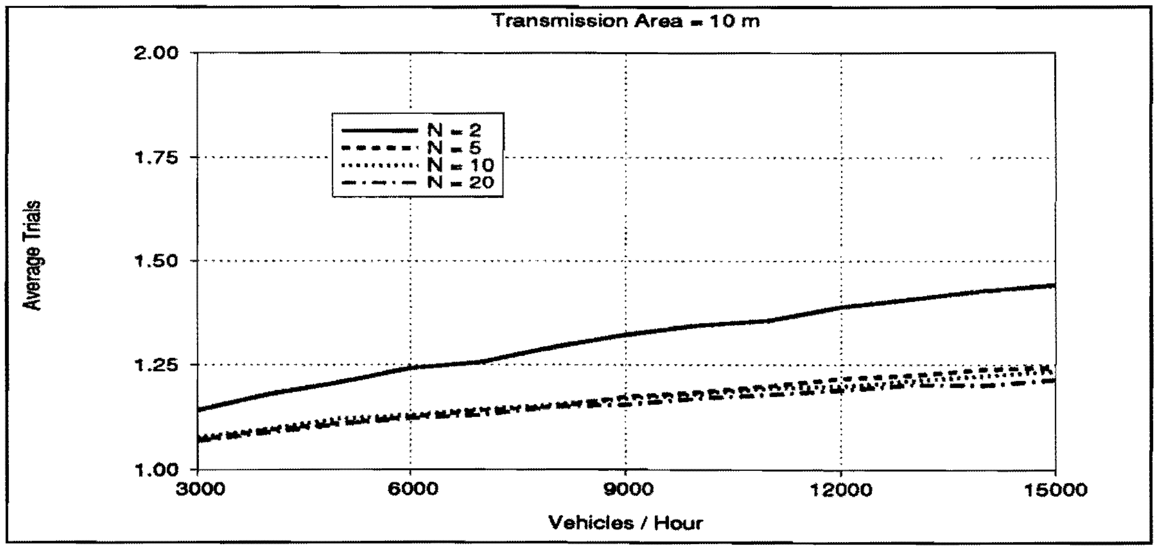
Step 1 : Wait for Frame4

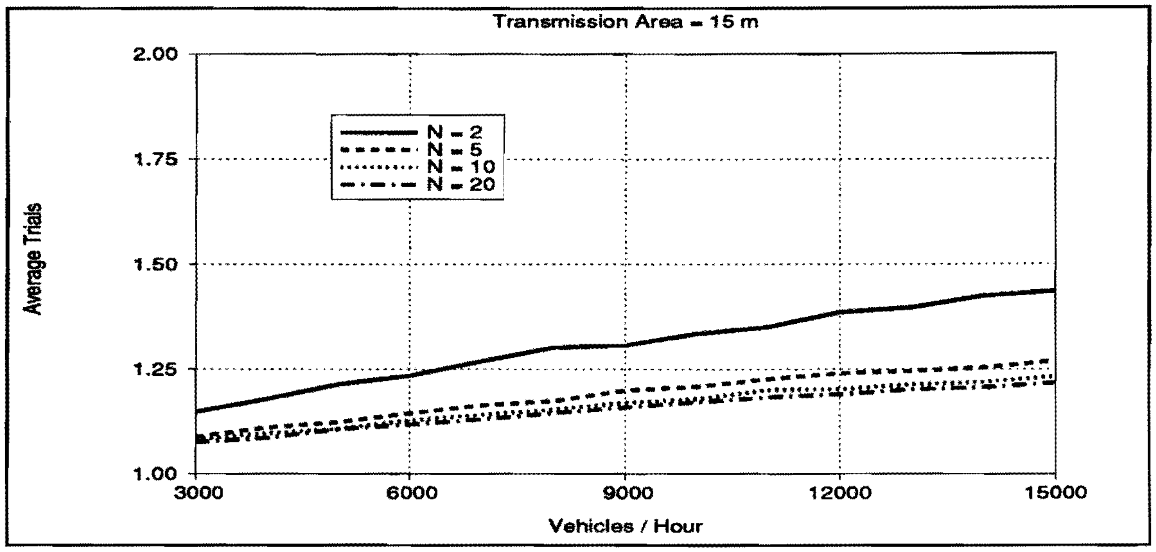
Step 2 : If Frame4 is received or if there is no transmission during a delay T3 then the beacon considers the vehicle is not transmitting any more and its assigned sequence is given to another user. The delay T3 can be chosen to specify the longest delay during which no transmission can exist.

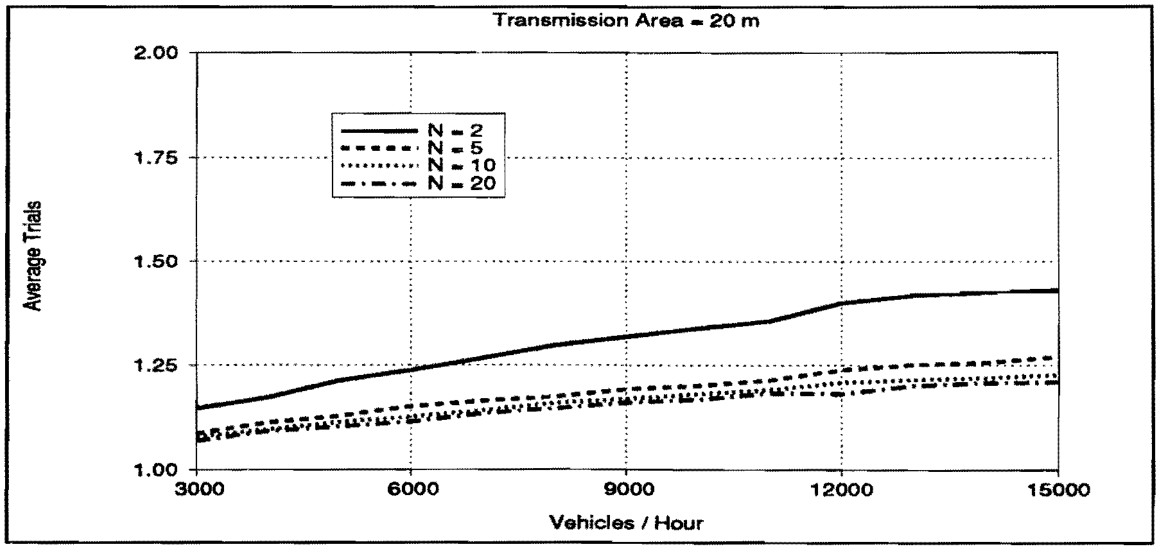
D. Protocol simulation

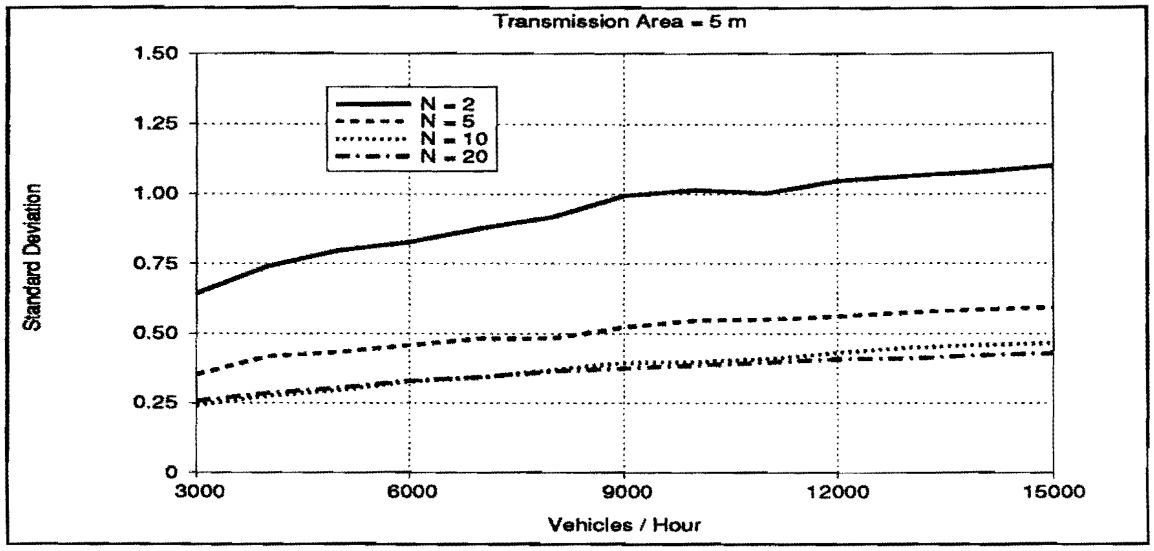
By using the highway simulation program with the same parameters (number of lanes, speed of the vehicles, width of the beacon area, etc...) as before, we can simulate the action of the protocol previously defined. The protocol simulation uses two principal parameters which are the transmission area and the number of slots for the retransmission. The transmission area is the area at the begining of the beacon area where the initialization procedure take place before actual data is communicated. We consider that all vehicles have established communication when they exit this area. The following results show the average number of trials needed to establish communication between the vehicle and the beacon, the standard deviation and the distribution of the trials. The average number of trials corresponds to the number of times the vehicles need to go back to step 1 in the protocol. The distribution of trials shows how many times (on average) k trial s were done to establish communication. We can easily see from these results that the average number of trials decreases when the transmission area is smaller or when the number of retransmission slots increases. In all cases, the average number of trials is less than two which means that in about two trials, a vehicle has access to the beacon.


Transmission Area = 5 m
Average Trials
2.00
1.75
1.50
1.25
1.00
3000
6000
9000
12000
15000
Vehicles / Hour
N = 2
N = 5
N = 10
N = 20

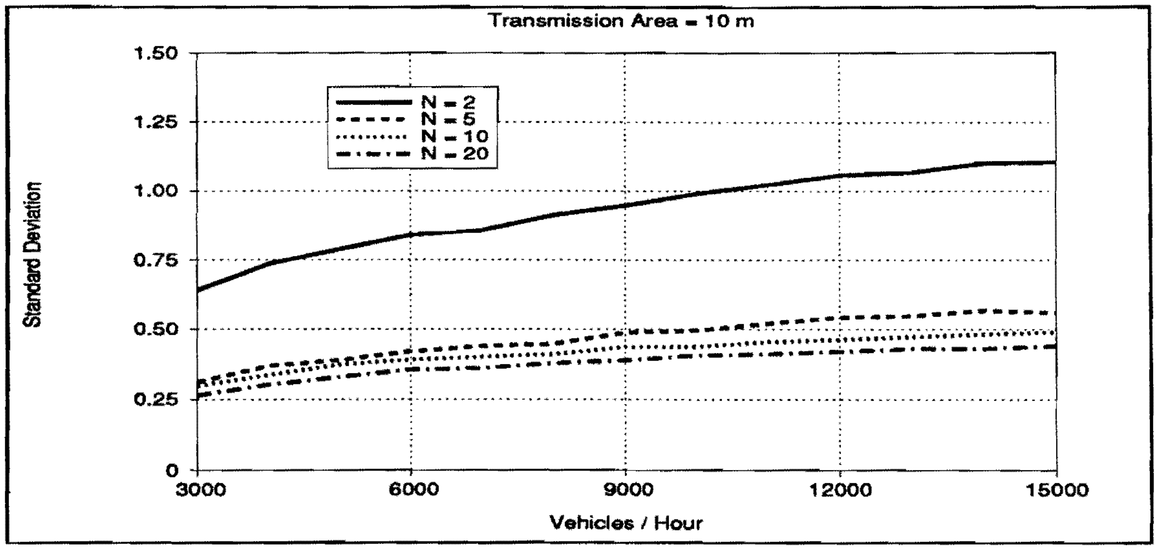

Transmission Area = 10 m
Average Trials
2.00
1.75
1.50
1.25
1.00
3000
6000
9000
12000
15000
Vehicles / Hour
N = 2
N = 5
N = 10
N = 20


Transmission Area = 15 m
2.00
1.75
1.50
1.25
1.00
Average Trials
N = 2
N = 5
N = 10
N = 20
3000
6000
9000
12000
15000
Vehicles / Hour

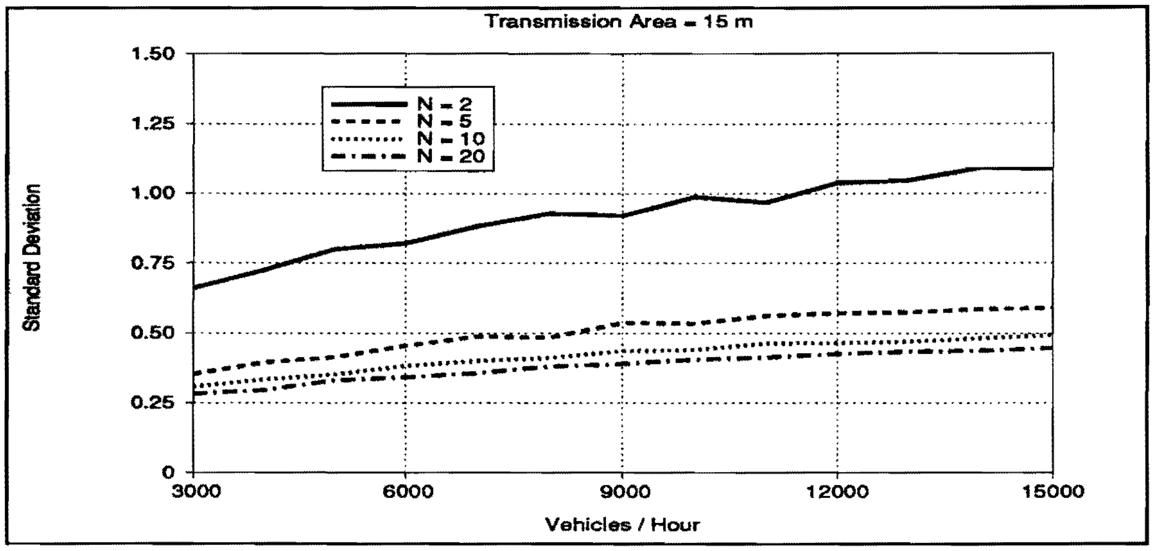

Transmission Area = 20 m
2.00
1.75
1.50
1.25
1.00
Average Trials
N = 2
N = 5
N = 10
N = 20
3000
6000
9000
12000
15000
Vehicles / Hour

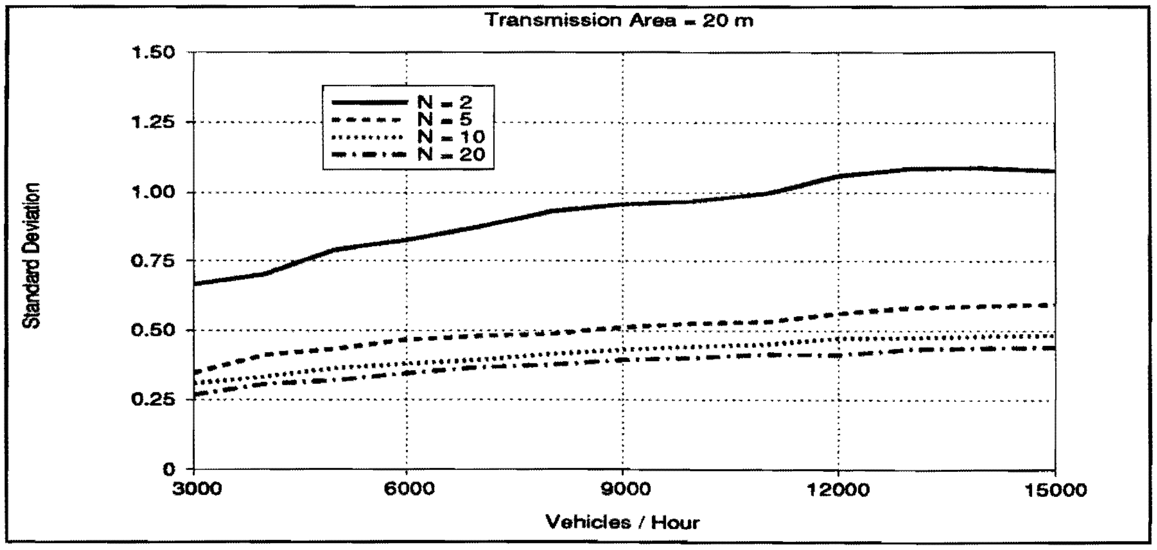


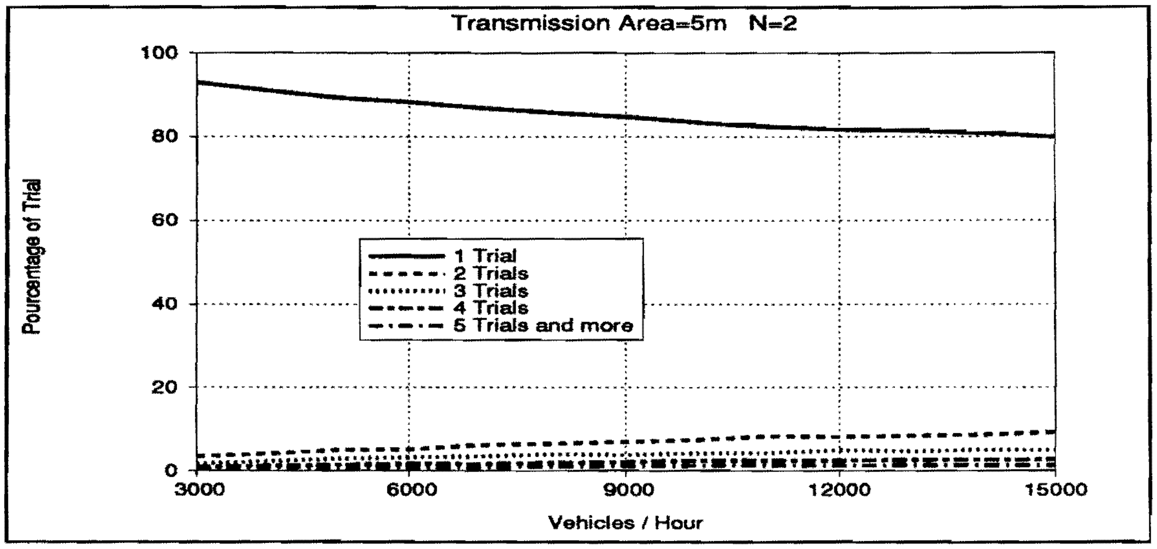
Transmission Area = 5 m
N = 2
N = 5
N = 10
N = 20
Standard Deviation
1.50
1.25
1.00
0.75
0.50
0.25
0
3000
6000
9000
12000
15000
Vehicles / Hour

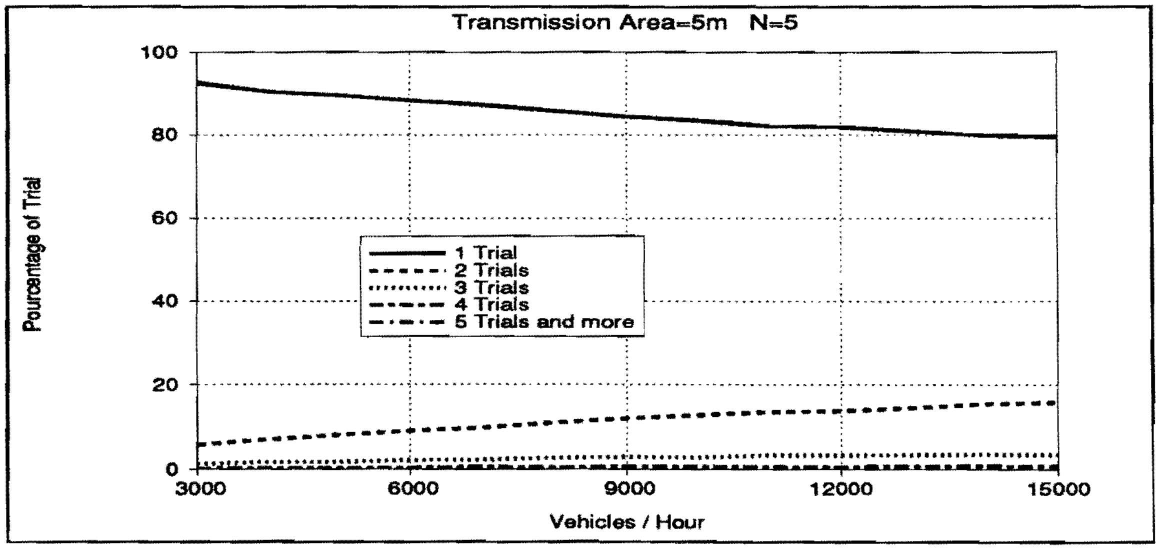


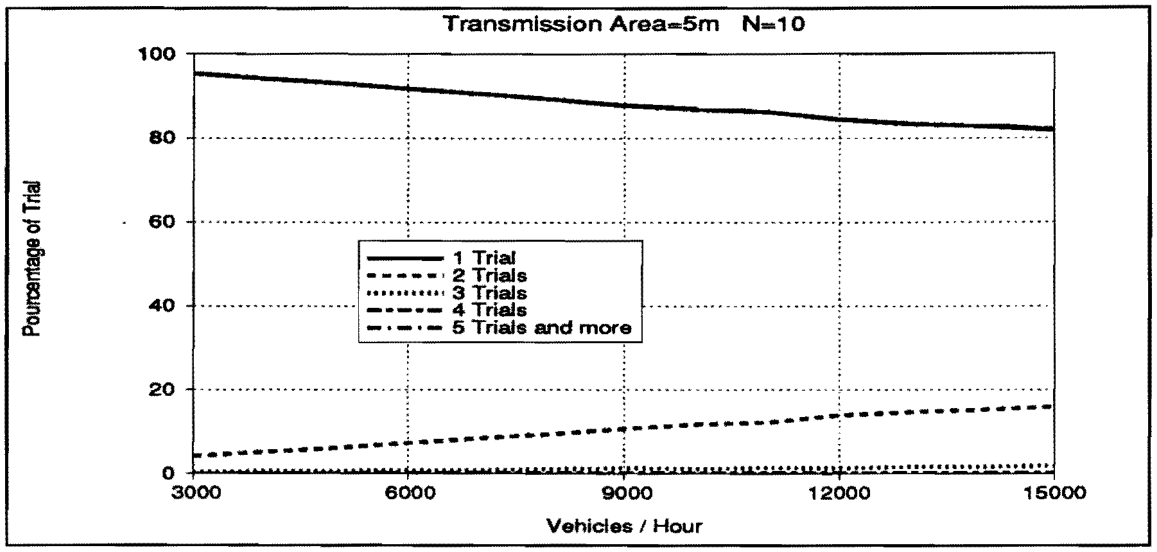
Transmission Area = 10 m
N = 2
N = 5
N = 10
N = 20
Standard Deviation
1.50
1.25
1.00
0.75
0.50
0.25
0
3000
6000
9000
12000
15000
Vehicles / Hour

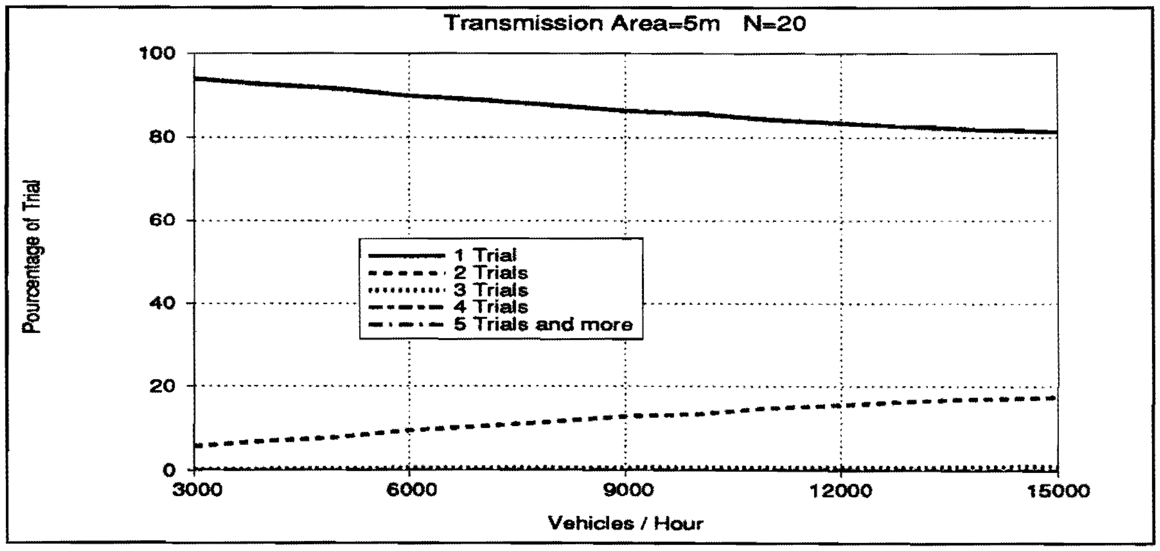

Transmission Area = 15 m
1.50
1.25
1.00
0.75
0.50
0.25
0
3000
6000
9000
12000
15000
Vehicles / Hour
Standard Deviation
N = 2
N = 5
N = 10
N = 20

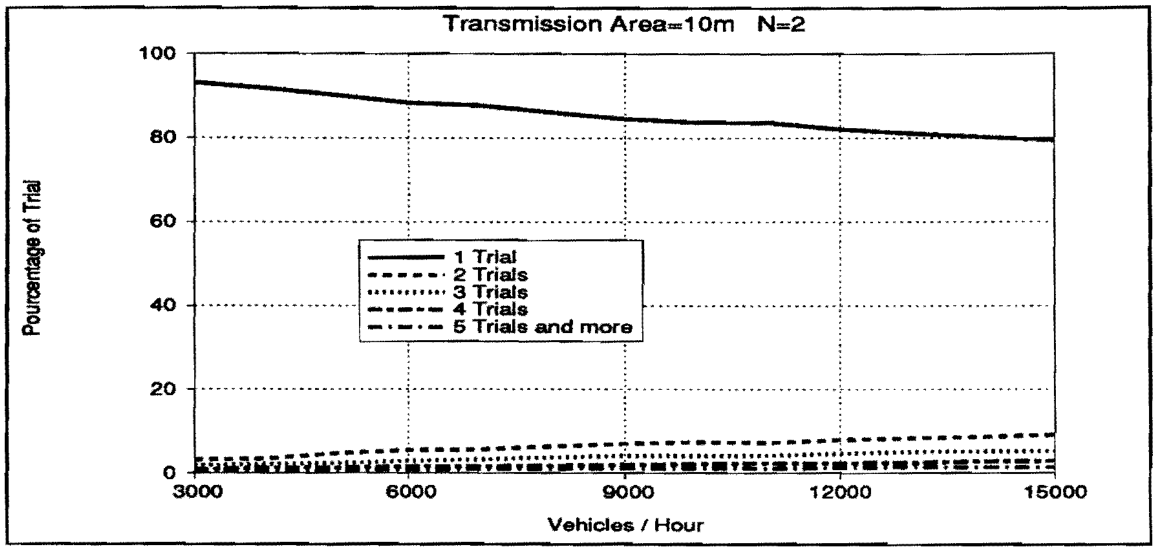

Transmission Area = 20 m
1.50
1.25
1.00
0.75
0.50
0.25
0
3000
6000
9000
12000
15000
Vehicles / Hour
Standard Deviation
N = 2
N = 5
N = 10
N = 20

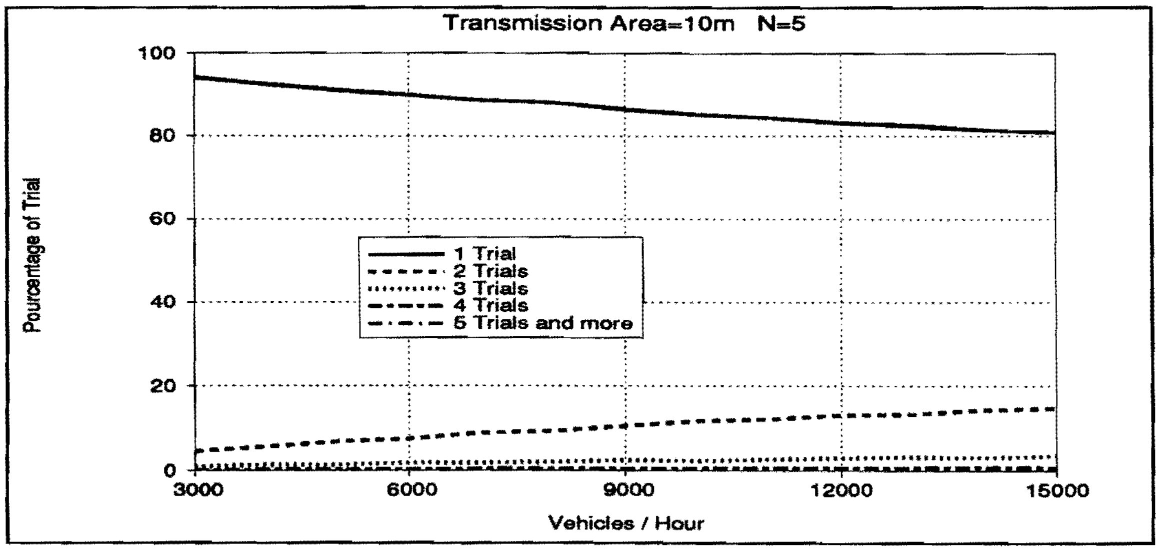

Transmission Area=5m N=2
Pourcentage of Trial
Vehicles / Hour
100
80
60
40
20
0
3000
6000
9000
12000
15000
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more

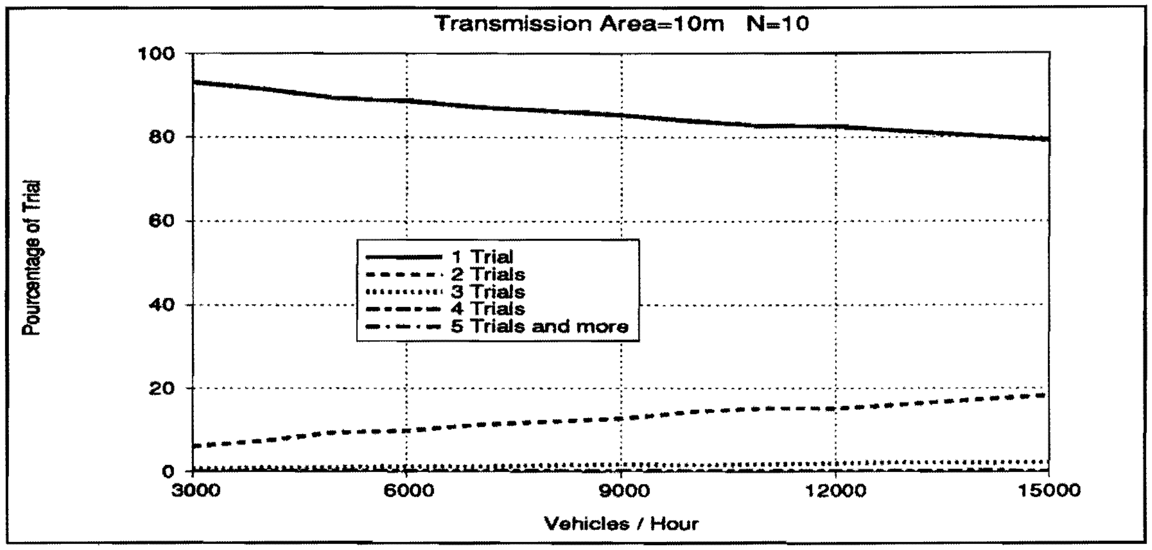

Transmission Area=5m N=5
Pourcentage of Trial
Vehicles / Hour
100
80
60
40
20
0
3000
6000
9000
12000
15000
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more

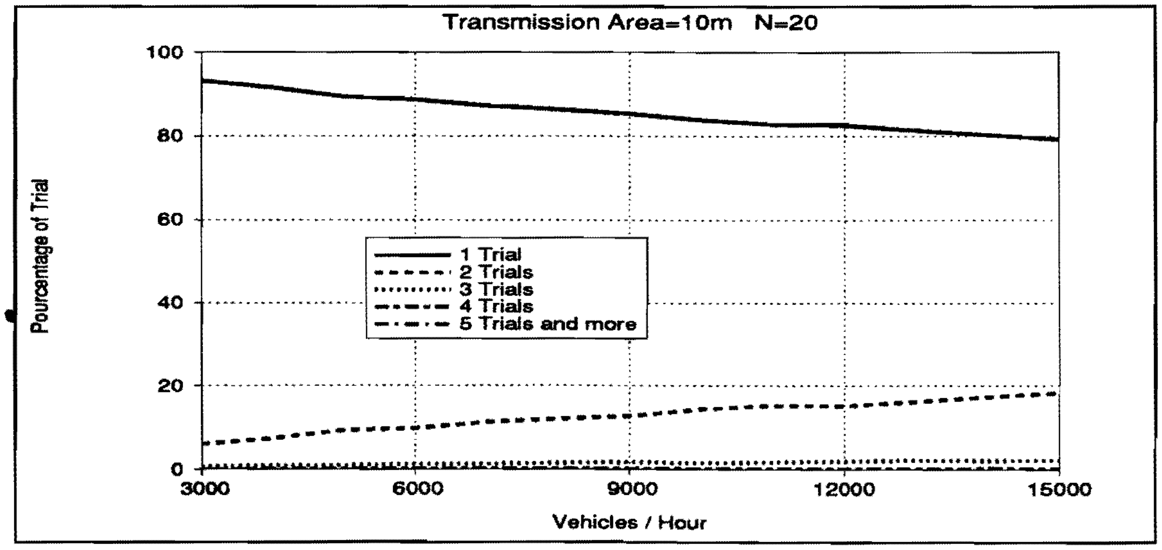

Transmission Area=5m N=10
Pourcentage of Trial
Vehicles / Hour
100
80
60
40
20
0
3000
6000
9000
12000
15000
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more

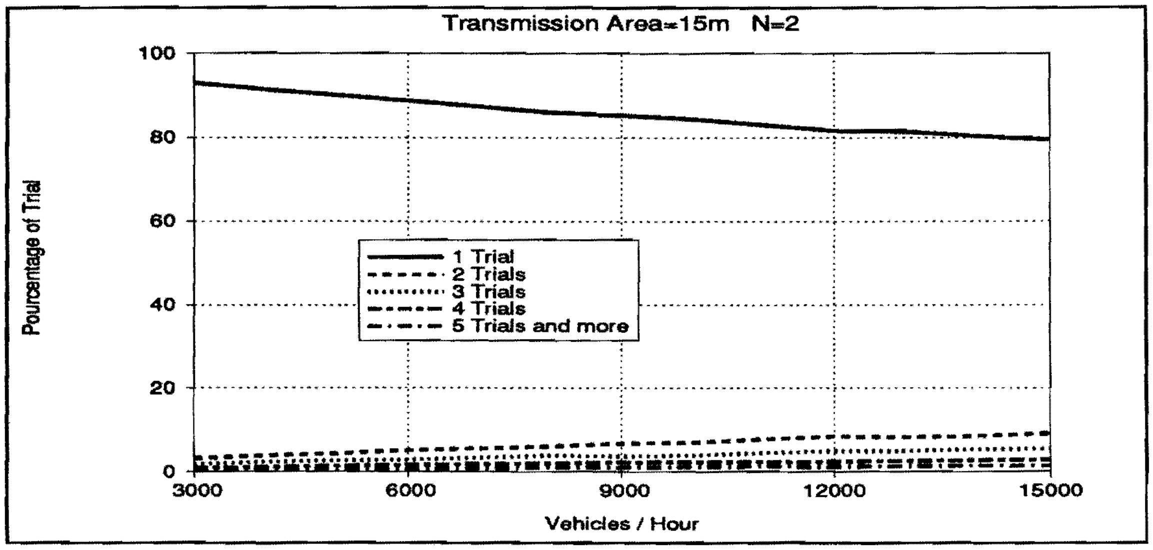

Transmission Area=5m N=20
Pourcentage of Trial
Vehicles / Hour
100
80
60
40
20
0
3000
6000
9000
12000
15000
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more

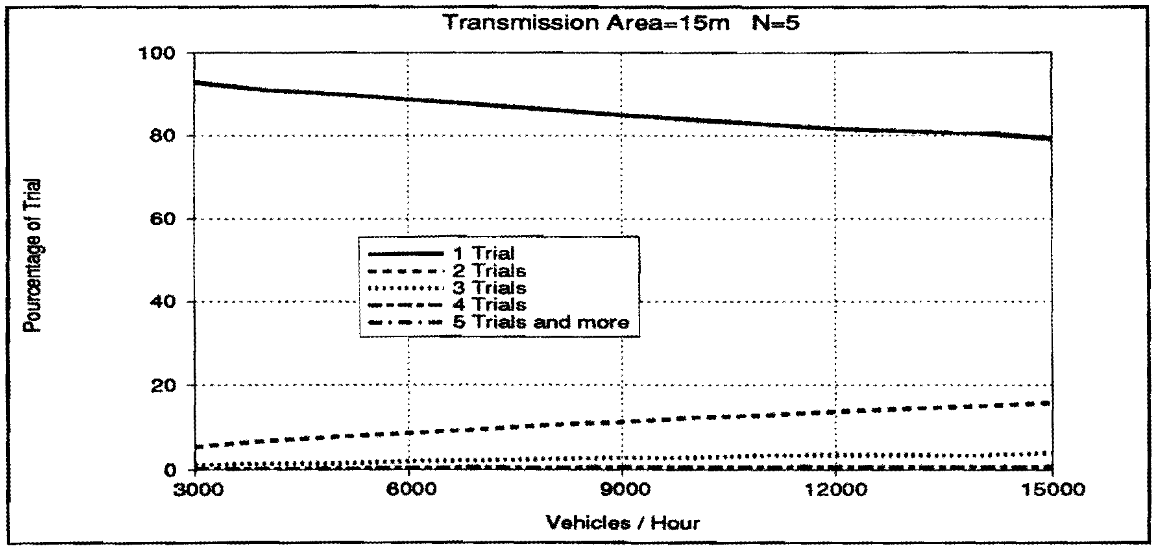

Transmission Area=10m N=2
Pourcentage of Trial
Vehicles / Hour
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more
100
80
60
40
20
0
3000
6000
9000
12000
15000

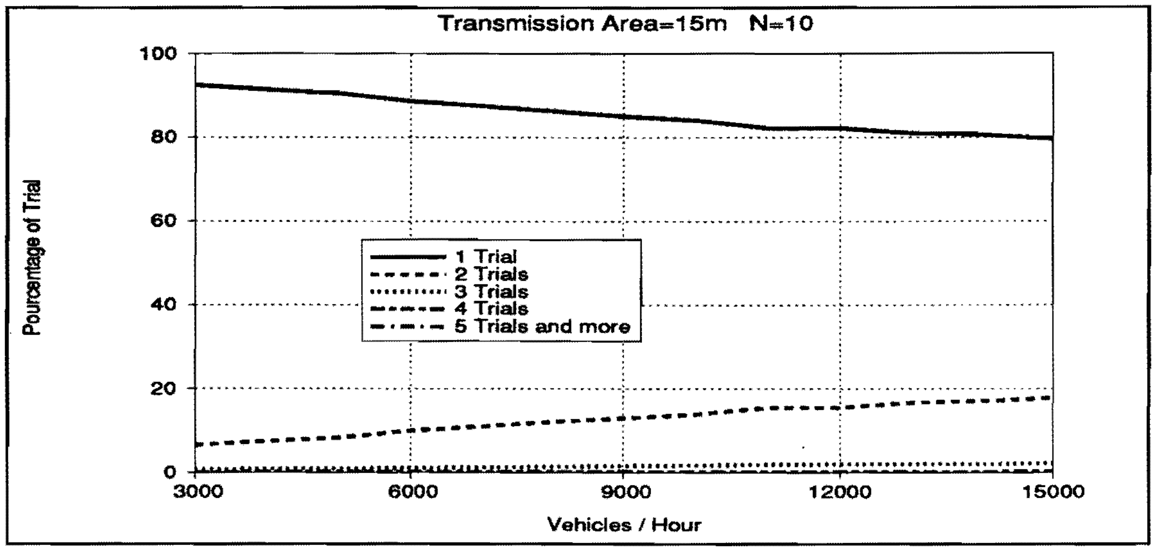

Transmission Area=10m N=5
Pourcentage of Trial
Vehicles / Hour
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more
100
80
60
40
20
0
3000
6000
9000
12000
15000

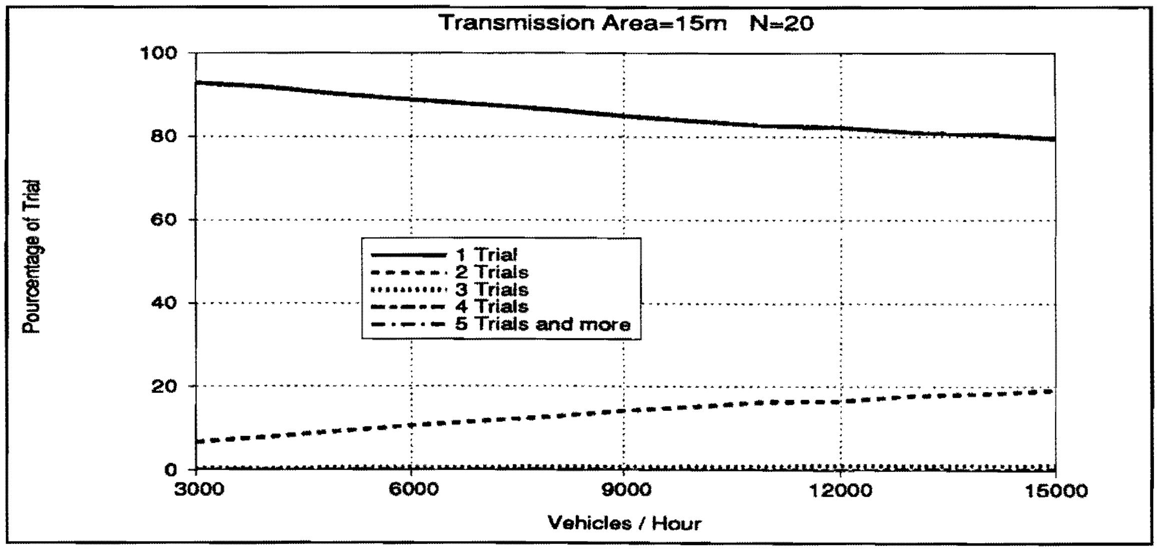

Transmission Area=10m N=10
Pourcentage of Trial
Vehicles / Hour
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more
100
80
60
40
20
0
3000
6000
9000
12000
15000

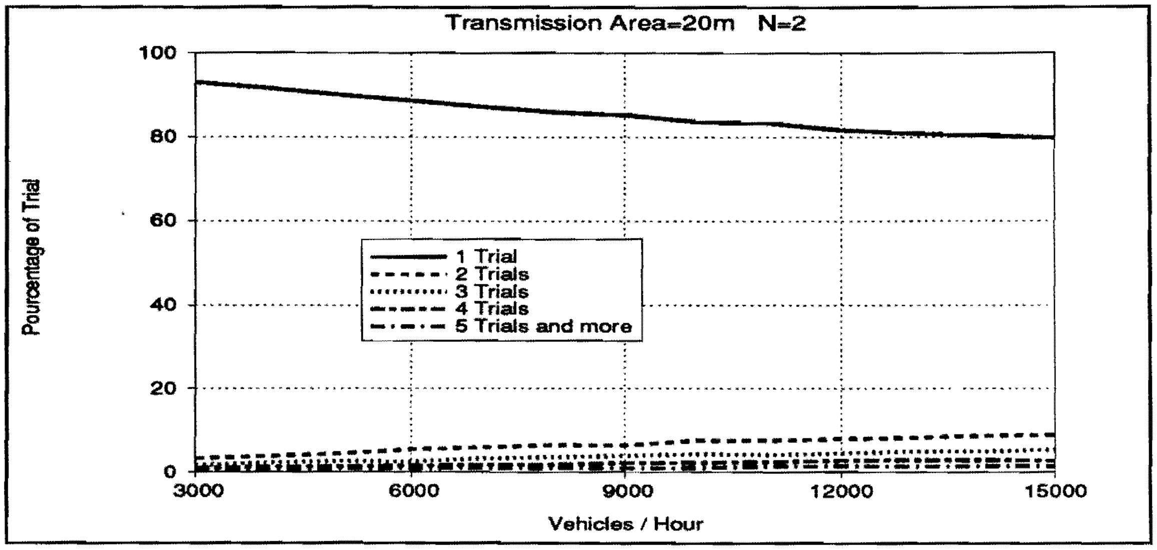

Transmission Area=10m N=20
Pourcentage of Trial
Vehicles / Hour
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more
100
80
60
40
20
0
3000
6000
9000
12000
15000

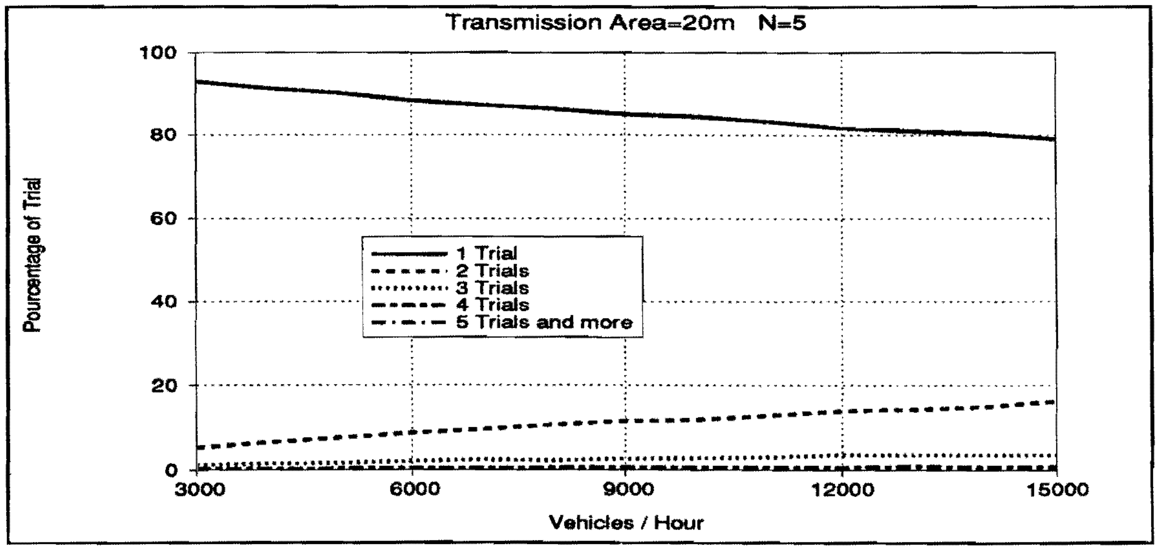

Transmission Area=15m N=2
Pourcentage of Trial
Vehicles / Hour
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more
100
80
60
40
20
0
3000
6000
9000
12000
15000

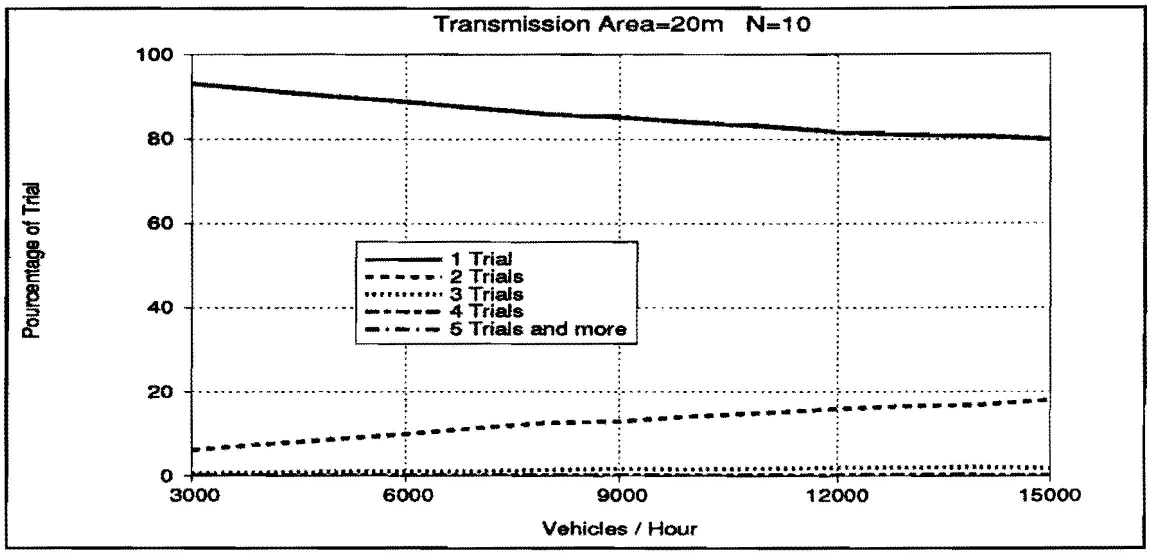

Transmission Area=15m N=5
Pourcentage of Trial
Vehicles / Hour
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more
100
80
60
40
20
0
3000
6000
9000
12000
15000

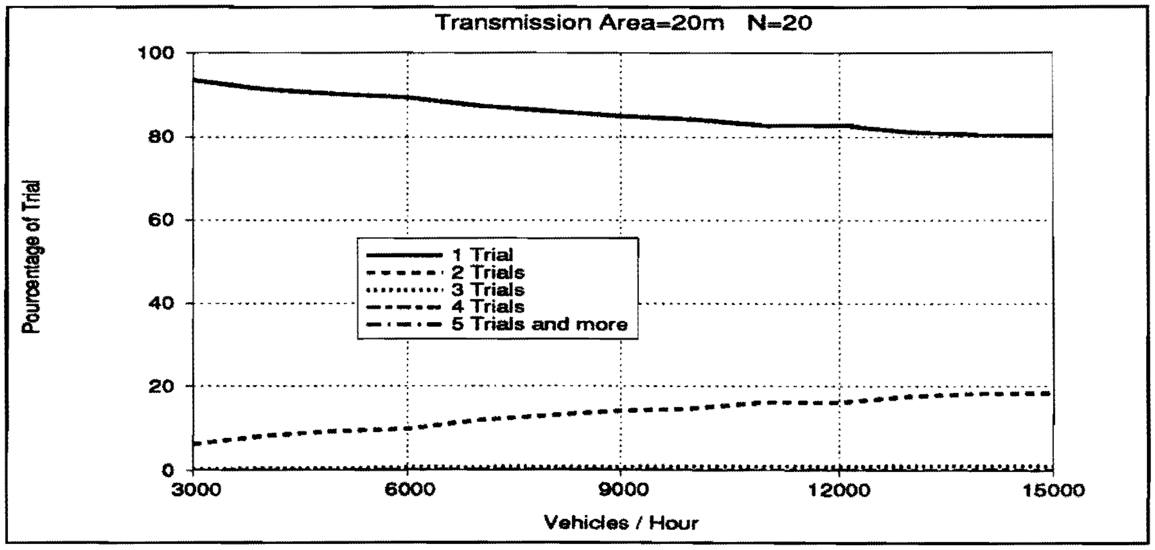

Transmission Area=15m N=10
100
80
60
40
20
0
3000
6000
9000
12000
15000
Vehicles / Hour
Pourcentage of Trial
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more


Transmission Area=15m N=20
100
80
60
40
20
0
3000
6000
9000
12000
15000
Vehicles / Hour
Pourcentage of Trial
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more


Transmission Area=20m N=2
100
80
60
40
20
0
3000
6000
9000
12000
15000
Vehicles / Hour
Pourcentage of Trial
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more


Transmission Area=20m N=5
100
80
60
40
20
0
3000
6000
9000
12000
15000
Vehicles / Hour
Pourcentage of Trial
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more


Transmission Area=20m N=10
100
80
60
40
20
0
3000
6000
9000
12000
15000
Vehicles / Hour
Pourcentage of Trial
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more


Transmission Area=20m N=20
100
80
60
40
20
0
3000
6000
9000
12000
15000
Vehicles / Hour
Pourcentage of Trial
1 Trial
2 Trials
3 Trials
4 Trials
5 Trials and more

E. Study of a Practical Case

By looking at the previous results, we can consider N=5 as the case which gives the better compromise with respect to the average number of trials, standard deviation and waiting time between two trials (worst case is equal to NΔt). Therefore, by taking N=5 and assuming that the maximum length of the initialization frame is 100 bits, the transmission rate is 200 kbits/s, the maximum vehicle speed is 120 km/h (75 MPH) and the computational time at the beacon is at worst equal to Δt (transmitting frame time), the moving distance of a vehicle during its two trials is equal to 0.17m. Also, by taking this value as the initialization area, we obtain an average number of trials uniformly equal to one regardless of the chosen traffic (Fig. 11). This result indicates that each vehicle needs only one trial to initialize the communication with the beacon which indicates that there can be no request collision during the initiali zation process. Also, in this case, the parameter N (number of retransmission slots) is useless and can be set equal to one, which reduces again the transmission area.

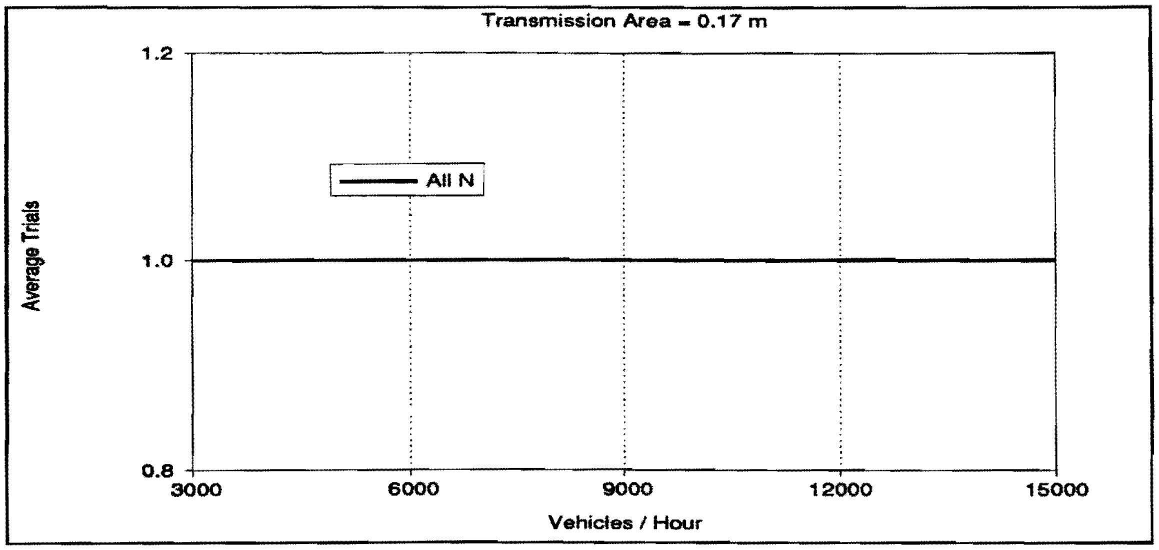


Fig. 11.

F. Conclusion

These simulations showed that the protocol is very efficient to initialize the communication between a vehicle and the beacon by reaching the minimum possible of one trial. However, these simulations were realized without considering any interferences except the request collisions. Also, future studies could be done on analyzing more realistic models by considering the probability of error in the transmission and problems such as fading and "near-far".

A study of the second part of the protocol (leaving the beacon) can be realized. This simulation will give us the average time taken by the beacon to recognize an end of transmission either by reception of frame4 or by checking the non-transmission-time. By adding the two parts of the protocol together, a simulation of the complete system can generate more realistic results and also the ratio of vehicles rejected by the beacon because of saturation.

Another interesting realization would be a simulation by varying the traffic density. The present simulation considers only fixed traffic density. However, we know that the number of vehicles per hour on a highway varies during a day. So, by varying this quantity an analysis of the evolution of the results in time can be performed in order to optimize the system.

Finally, a development of the protocol can also be realized in order to consider some high level applications in the context of highway communication.

APPENDIX B

SIMULATION PROGRAM

```
/*****************************************************************

    Simulation of a Highwayand protocol of communication

******************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define  boolean  int
#define  true  1
#define  false  0
#define  then
#define  and  &&
#define  or  ||
#define  not  !
#define  step 0.001
#define max_trial 8

struct car
       {
         int nber;
         int speed;
         double move;
       };

struct stat
       {
         int sum;
         int sum_sq;
       };


typedef struct stat tstat;
typedef struct car tcar;

typedef struct prot *pt_prot;
struct prot
       {
         int nber;
         int status;
         double thres;
         int ITZ;
         int nexttrans;
         pt_prot next;
          pt_prot prev;
       };

boolean square_free(zone,L,C,i,j)
tcar *zone;
int L;
int C;
int i;
int j;
{ tcar vehicle;
  vehicle = *(zone+i*C+j);
  return (vehicle.nber == 0);
}

/* x1 <= Dble < x2   ===> return x1 */
int integer_part(Dble)
double Dble;
{
  double rest=Dble;
  int result=0;

  if (Dble == 0)
  then return 0;
  else {
    while (rest >= 1)
       {
         rest = rest - 1.0;
         result = result + 1;
       }
     return (result);
       }
}


long int power2(n)
int n;
{
  long int res=1;
  int i;

  for (i=0;i<n;i++)
    res = res * 2;

  return res;
}


/* return a int value in [0,x]  x included */
```

```
int random_int(x)
int x;
{
  double aux;
  int ra;

  ra = random();
  aux = (ra*1.0 / (power2(31)-1) * (x+1));
  return integer_part(aux);
}


/* return a value in [0,x]  x included */
double random_double(x)
int x;
{
  double aux;
  int ra;

  ra = random();
  aux = (ra*1.0 / (power2(31)-1) * x);
  return aux;
}


void visu(zone,L,C2,C)
tcar *zone;
int L;
int C2;
int C;
{
  int i,j;
  char ch;
  tcar vehicle;

  for (i=0;i<L;i++)
   {
     for (j=C2-C;j<C2;j++)
       {
         vehicle = *(zone+i*C2+j);
         printf("(%6d %3d %1.31f)",vehicle.nber,vehicle.speed,vehicle.move);
       };
     printf("\n");
   }
scanf("%c",&ch);
}



void visualization(filename)
char *filename;
```

```
{
  FILE *input_file;
  int L,C;
  int i,j;
  tcar *zone;
  double time;
  double delta_t;
  tcar *pt_vehicle;

  input_file = fopen(filename,"r");
  fscanf(input_file,"%d",&L);
  fscanf(input_file,"%d",&C);
  fscanf(input_file,"%lf",&time);
  fscanf(input_file,"%lf",&delta_t);

  zone = (tcar *) calloc(L*C,sizeof(tcar));
  time = 0;
  while (not(feof(input_file)))
    { printf("T = %5.2f\n",time);
      for (i=0;i<L;i++)
for (j=0;j<C;j++)
  { pt_vehicle = zone+i*C+j;
    fscanf(input_file,"%d ",&((*pt_vehicle).nber));
    fscanf(input_file,"%d ",&((*pt_vehicle).speed));
    fscanf(input_file,"%lf ",&((*pt_vehicle).move));
  }
      visu(zone,L,C,C);
      time = time + delta_t;
    }
  fclose(input_file);
  free(zone);
}


int change_lane(zone,L,C,i,j,res)
tcar *zone;
int L;
int C;
int i;
int j;
int *res;
{
  int l1;
  int l2;
  int free_l1;
  int free_l2;

  if (i==0) { l2=1;
           free_l2 = square_free(zone,L,C,l2,j);
        if (free_l2) {*res=l2;return 1;} else return 0;
          }
  else if (i==L-1) {l1=L-2;
                   free_l1 = square_free(zone,L,C,l1,j);
          if (free_l1) {*res=l1; return 1;} else return 0;
         }
     else {l1=i-1;l2=i+1;
            free_l1 = square_free(zone,L,C,l1,j);
        free_l2 = square_free(zone,L,C,l2,j);
        if (free_l1) {*res=l1; return 1;}
        else if (free_l2) {*res=l2;return 1;}
             else return 0;
           };
}


void next_position(zone,L,C,lane,beg,end,lane_end,pos_end)
tcar *zone;
int L;
int C;
int lane;
int beg;
int end;
int *lane_end;
int *pos_end;
{
  boolean exit;
  int i;

  *lane_end = lane;
  if (beg == end)
  *pos_end = end;
  else if (square_free(zone,L,C,lane,end))
          { *pos_end = end;}
       else if (change_lane(zone,L,C,lane,end,lane_end))
            { *pos_end = end;}
        else {
           exit = false;
           for (i=end-1;(i>beg) and (not(exit));i--)
                     { if (square_free(zone,L,C,lane,i))
                       then exit=true;
                     }
               if (exit)
               *pos_end = i+1;
               else *pos_end = beg;
         }

}



move_next(zone,L,C,i,j,delta_t,LC,lane,next)
```

```
tcar *zone;
int L;
int C;
int i;
int j;
double delta_t;
int LC;
int *lane;
int *next;
{
  int speed;
  int position;
  double move;
  tcar *pt_vehicle;
  int next_pos;

  pt_vehicle = zone+i*C+j;
  speed = (*pt_vehicle).speed;
  move = (delta_t*speed*1000)/(3600*LC) + (*pt_vehicle).move;

  position = integer_part(move);

  if (j+position >= C)
  then *next = (j+position);
  else {
         next_position(zone,L,C,i,j,j+position,lane,next);

     if ((*lane==i) && (*next != j+position))
     { (*pt_vehicle).move = (*(zone+*lane*C+*next+1)).move;}
     else if ((*lane != i) && (*next == j+position-1))
             { (*pt_vehicle).move = (*(zone+*lane*C+*next+1)).move;
         }
         else {(*pt_vehicle).move = move - position;}
       }
}



move_cars(zone,L,C,C2,delta_t,LC,car_in,car_out)
tcar *zone;
int L;
int C;
int C2;
double delta_t;
int LC;
int *car_in;
int *car_out;
{
  int i,j;
  int next;
  int lane;
  tcar *pt_vehicle;
  tcar *pt_vehicle2;

 *car_in = 0;
  *car_out = 0;
  for (j=C2-1;j>=0;j--)
    for (i=0;i<L;i++)
      {
        pt_vehicle = (tcar *) zone+i*C2+j;
    if ((*pt_vehicle).nber != 0)
    {
      move_next(zone,L,C2,i,j,delta_t,LC,&lane,&next);
          pt_vehicle2 = (tcar *) zone+lane*C2+next;

          if ((j<C2-C) && (next>=C2-C))
      {*car_in = *car_in + 1;}

      if (next < C2)
      { if ((next != j) || (lane != i))
         {
           *(pt_vehicle2) = *(pt_vehicle);
           pt_vehicle -> nber = 0;
           pt_vehicle -> speed = 0;
           pt_vehicle -> move = 0;
         };
          }
      else { *car_out = *car_out + 1;
             (*pt_vehicle).nber = 0;
             (*pt_vehicle).speed = 0;
             (*pt_vehicle).move = 0;
           }
         }
      }

}


int law_speed(sp_min,sp_max)
int sp_min;
int sp_max;
{
 return (sp_min + random_int(sp_max-sp_min));
}


boolean law_arriving(CH,NL,delta_t)
int CH;
int NL;
double delta_t;
```

```
{
  double pstep;
  double lambda;
  double threshold;

  lambda = CH/(NL*3600.0);
  threshold = lambda * delta_t;

  if (random_double(1) < threshold)
  then return true;
  else return false;
}



int fill_arriving(zone,L,C,C2,car_nber,sp_min,sp_max,
                  delta_t,CH,NL,car_in)
tcar *zone;
int L;
int C;
int C2;
int car_nber;
int sp_min;
int sp_max;
double delta_t;
int CH;
int NL;
int *car_in;
{
  int i;
  int new_car = car_nber;
  tcar *pt_vehicle;

 *car_in = 0;
  for (i=0;i<L;i++)
    { if (square_free(zone,L,C2,i,0))
      then { if (law_arriving(CH,NL,delta_t))
         then { pt_vehicle = zone+i*C2;
                (*pt_vehicle).nber = new_car;
            (*pt_vehicle).speed = law_speed(sp_min,sp_max);
            (*pt_vehicle).move = 0;
                new_car = new_car + 1;
            if (C2==C)
            *car_in = *car_in + 1;
          }
      }
    }
  return (new_car - car_nber);
}
```

```
void init_circulation(zone,L,C)
tcar *zone;
int L;
int C;
{
  int i,j;
  tcar *pt_vehicle;

  for (i=0;i<L;i++)
    for (j=0;j<C;j++)
      { pt_vehicle = zone+i*C+j;
        (*pt_vehicle).nber = 0;
    (*pt_vehicle).speed = 0;
    (*pt_vehicle).move = 0;
      }
}



void save_data(zone,L,C,C2,filename,first,tmax,delta_t)
tcar *zone;
int L;
int C;
int C2;
char *filename;
boolean first;
double tmax;
double delta_t;
{
  FILE *output_file;
  int i,j;
  tcar vehicle;

  if (first)
     then { output_file = fopen(filename,"w");
        fprintf(output_file,"%d\n",L);
        fprintf(output_file,"%d\n",C);
        fprintf(output_file,"%lf\n",tmax);
        fprintf(output_file,"%lf\n",delta_t);

      }
     else { output_file = fopen(filename,"a"); }

  fprintf(output_file,"\n");

  for (i=0;i<L;i++)
    {
      for (j=C2-C;j<C2;j++)
    { vehicle = *(zone+i*C2+j);
      fprintf(output_file,"%d ",vehicle.nber);
      fprintf(output_file,"%d ",vehicle.speed);
      fprintf(output_file,"%lf ",vehicle.move);
    }
      fprintf(output_file,"\n");
    }
  fclose(output_file);
}



void simul_highway(LC,NL,WA,delta_t,t_max,speed_min,speed_max,CH,
                   filename,choice,prt)
int LC;
int NL;
int WA;
double delta_t;
double t_max;
int speed_min;
int speed_max;
int CH;
char *filename;
int choice;
int prt;
{
  int car_nber = 1;
  int car_out=0;
  int car_in=0;
  int nber_car=0;
  tcar *zone_beacon;
  int line;
  int column;
  int column_total;
  int Nb_mvt;
  int cpter;
  int i,t;
  int loop_max;
  int window;
  int car_fill_in;
  int present;
  double var_time;
  int cpter_max;


  line = NL;
  column = WA / LC;
  Nb_mvt = integer_part(t_max/delta_t);
  if (t_max/delta_t-Nb_mvt > 0)
  Nb_mvt++;
  t = (((speed_min+speed_max)/2)*delta_t*1000)/(3600*LC);
  if ( t > 1)
  window = integer_part(t) + 1;
```

```
else window = 5;
column_total = window * column;
zone_beacon = (tcar *) calloc(column_total*line,sizeof(tcar));

init_circulation(zone_beacon,line,column_total);
if (choice == 1)
save_data(zone_beacon,line,column,column_total,filename,true,t_max,delta_t);

car_nber = car_nber + fill_arriving(zone_beacon,line,column,
                                    column_total,car_nber,speed_min,
                         speed_max,delta_t,CH,NL,
                  &car_fill_in);
present = car_fill_in;


 if (choice==2)
 {var_time = 1;}
 else {var_time = delta_t;};
 loop_max = 10000;

for (t=0;    ((choice==2) && (t<loop_max))
          || ((choice==1) && (present==0));t++)
  {
    if ((choice==2) && (prt))
       printf("Stabilisation : %d / %d\n",t+1,loop_max);

    move_cars(zone_beacon,line,column,column_total,
              delta_t,LC,&car_in,&car_out);
    car_nber = car_nber + fill_arriving(zone_beacon,line,column,
                                       column_total,car_nber,speed_min,
                                       speed_max,var_time,CH,NL,
                                   &car_fill_in);
    present = present + car_fill_in + car_in - car_out;
  }

if (choice == 1)
  {
    cpter_max = Nb_mvt-1;
    save_data(zone_beacon,line,column,column_total,filename,false,0,0);
  }
  else {
         cpter_max = Nb_mvt;
     save_data(zone_beacon,line,column,column_total,filename,
               true,t_max,delta_t);
       };

cpter = 0;
do
  {
    move_cars(zone_beacon,line,column,column_total,
          delta_t,LC,&car_in,&car_out);
```

```
      car_nber = car_nber + fill_arriving(zone_beacon,line,column,
                                         column_total,car_nber,speed_min,
                             speed_max,delta_t,CH,NL,
                         &car_fill_in);
     save_data(zone_beacon,line,column,column_total,filename,false,0,0);
      present = present + car_fill_in + car_in - car_out;
      cpter++;
    }
  while (cpter < cpter_max);
  free(zone_beacon);
}



tstat *simul_MTime(max,LC,NL,WA,delta_t,t_max,
                   speed_min,speed_max,CH,filename,prt,choice)
int max;
int LC;
int NL;
int WA;
double delta_t;
double t_max;
int speed_min;
int speed_max;
int CH;
char *filename;
int prt;
int choice;
{
  int i,t;
  FILE *output;
  int column_stat;
  int car_nber = 1;
  int car_out=0;
  int car_in=0;
  tcar *zone_beacon;
  tstat *NCPT;    /* Nber of Car Per Time  */
  int line;
  int column;
  int column_total;
  int window;
  int car_fill_in;
  int present;
  int stab;
  double mean;
  double var;
  double var_time;

  output = fopen(filename,"w");
  line = NL;
```

```
column = WA / LC;
column_stat = integer_part(t_max/delta_t);
if (t_max/delta_t-column_stat > 0)
column_stat++;
mean = (((speed_min+speed_max)/2)*delta_t*1000)/(3600*LC);
if ( mean > 1)
window = integer_part(mean) + 1;
else window = 5;
column_total = window * column;
zone_beacon = (tcar *) calloc(column_total*line,sizeof(tcar));
NCPT = (tstat *) calloc(column_stat,sizeof(tstat));
init_circulation(zone_beacon,line,column_total);

stab = 0;
for (i=0;i<max;i++)
  {
   if ((choice ==1) || (stab==0))
   {
     init_circulation(zone_beacon,line,column_total);
     car_nber = car_nber + fill_arriving(zone_beacon,line,column,
                                    column_total,car_nber,speed_min,
                                      speed_max,delta_t,CH,NL,
                                    &car_fill_in);
     present = car_fill_in;
   }

   if ((stab==0) || (choice==1))
   {
     if (choice==2)
        {var_time = 1;}
    else {var_time = delta_t;};

     for (t=0;    ((choice==2) && (t<7200))
              || ((choice==1) && (present==0));t++)
       {
         if ((choice==2) && (prt))
            printf("Stabilisation : %d / %d\n",t+1,7200);
         move_cars(zone_beacon,line,column,column_total,
                   delta_t,LC,&car_in,&car_out);
         car_nber = car_nber + fill_arriving(zone_beacon,line,column,
                                        column_total,car_nber,speed_min,
                                        speed_max,var_time,CH,NL,
                                         &car_fill_in);


       present = present + car_fill_in + car_in - car_out;
       }
     stab = 1;
   };

   if (prt) printf("Simulation # %d / %d\n",i+1,max);
```

```
      for (t=0;t<column_stat;t++)
        {
          move_cars(zone_beacon,line,column,column_total,
                    delta_t,LC,&car_in,&car_out);
          car_nber = car_nber + fill_arriving(zone_beacon,line,column,
                                            column_total,car_nber,speed_min,
                                            speed_max,delta_t,CH,NL,
                                            &car_fill_in);
      present = present + car_fill_in + car_in - car_out;
      (*(NCPT+t)).sum = (*(NCPT+t)).sum + present;
      (*(NCPT+t)).sum_sq = (*(NCPT+t)).sum_sq + (present * present);
        }
    }

  if (choice==1)
  fprintf(output,"%lf %lf %lf\n",0.0,0.0,0.0);

  for (t=0;t<column_stat;t++)
    {
      mean = (*(NCPT+t)).sum*1.0/max;
      var = (*(NCPT+t)).sum_sq*1.0/max - (mean*mean);
      if (choice==1)
      fprintf(output,"%lf %lf %lf\n",(t+1)*delta_t,mean,sqrt(var));
      else fprintf(output,"%lf %lf %lf\n",t*delta_t,mean,sqrt(var));
    }

  fclose(output);
  free(zone_beacon);
  return NCPT;
}



void simul_MTraffic(max,LC,NL,WA,delta_t,t_max,speed_min,
                    speed_max,filename)
int max;
int LC;
int NL;
int WA;
double delta_t;
double t_max;
int speed_min;
int speed_max;
char *filename;
{
  int traffic_min;
  int traffic_max;
  int inter;
  tstat *res;
  tstat *sim;
  int i;
  int last_col;
  FILE *output;
  double mean;
  double var;
  int j;
  int nb_col;


  printf("Traffic Min   (Vehicles/hour) : ");scanf("%d",&traffic_min);
  printf("Traffic Max   (Vehicles/hour) : ");scanf("%d",&traffic_max);
  printf("Interval                      : ");scanf("%d",&inter);

  output = fopen(filename,"w");
  nb_col = (traffic_max-traffic_min)/inter + 1;

  sim = (tstat *) calloc(nb_col,sizeof(tstat));
  last_col = integer_part(t_max/delta_t);

  for (i=0;i<nb_col;i++)
    {
      printf("Simulation For %d Vehicles/Hour\n",traffic_min + (i*inter));

      res = simul_MTime(max,LC,NL,WA,delta_t,t_max,speed_min,
                        speed_max,(i*inter)+traffic_min,"stat.inter",0,2);

      for (j=0;j<last_col;j++)
        {
          (*(sim+i)).sum = (*(sim+i)).sum + (*(res+last_col-1-j)).sum;
          (*(sim+i)).sum_sq = (*(sim+i)).sum_sq
                        + (*(res+last_col-1-j)).sum_sq;
        };

       free(res);
    }

  for (i=0;i<nb_col;i++)
    {
      mean = (*(sim+i)).sum*1.0/(last_col*max);
      var = (*(sim+i)).sum_sq*1.0/(last_col*max) - (mean*mean);
      fprintf(output,"%d %lf %lf\n",(i*inter)+traffic_min,mean,sqrt(var));
    }

  fclose(output);
  free(sim);
}


pt_prot ITZ_zero(pt)
pt_prot pt;
{
  pt_prot pt_aux;
  pt_aux = pt;

  if (pt_aux == (pt_prot)NULL)
  return pt_aux;
  else {
         (*pt_aux).ITZ = 0;
         pt_aux = (*pt_aux).next;
         while (pt_aux != pt)
           {
             (*pt_aux).ITZ = 0;
             pt_aux = (*pt_aux).next;
           }

         return pt;
       }
}


pt_prot new_car_prot()
{
  pt_prot pt;

  pt = malloc(sizeof(struct prot));
  (*pt).nber = 0;
  (*pt).status = 0;
  (*pt).thres = 0;
  (*pt).ITZ = 0;
  (*pt).nexttrans = 0;
  (*pt).next = (pt_prot)NULL;
  (*pt).prev = (pt_prot)NULL;
  return pt;
}



pt_prot exist(pt,nb,ok)
pt_prot pt;
int nb;
int *ok;
{
  pt_prot pt_aux;
  pt_aux = pt;

  *ok = 0;
  if ((*pt_aux).nber >= nb)
    {
      if ((*pt_aux).nber == nb)
      *ok=1;

      return pt_aux;
    }
  else pt_aux = (*pt_aux).next;

  while (((*pt_aux).nber < nb) && (pt_aux != pt))
    pt_aux = (*pt_aux).next;

  if ((*pt_aux).nber == nb)
  *ok = 1;
  else if (pt_aux == pt)
           pt_aux = (pt_prot)NULL;

  return pt_aux;
}




pt_prot add_car_prot(nb,thres,pt)
int nb;
double thres;
pt_prot pt;
{
  pt_prot pt_aux;
  pt_prot pt_aux2;
  int found;

  if (pt == (pt_prot)NULL)
    {
      pt_aux = new_car_prot();
      (*pt_aux).nber = nb;
      (*pt_aux).thres = thres;
      (*pt_aux).ITZ = 1;
      (*pt_aux).prev = pt_aux;
      (*pt_aux).next = pt_aux;
      return pt_aux;
    }
  else {
         pt_aux = exist(pt,nb,&found);
     if (found == 1)
           {
             (*pt_aux).ITZ = 1;
             (*pt_aux).thres = thres;
             return pt;
           }
         else {
                pt_aux2 = new_car_prot();
                (*pt_aux2).nber = nb;
                (*pt_aux2).thres = thres;
                (*pt_aux2).ITZ = 1;

                if (pt_aux == (pt_prot)NULL)
```

```
                    {
                      (*pt_aux2).prev = (*pt).prev;
                      (*pt_aux2).next = pt;
                      (*((*pt).prev)).next = pt_aux2;
                      (*pt).prev = pt_aux2;
                      return pt;
                    }
                  else if (pt_aux == pt)
                         {
                           (*pt_aux2).prev = (*pt).prev;
                           (*pt_aux2).next = pt;
                           (*((*pt).prev)).next = pt_aux2;
                           (*pt).prev = pt_aux2;
                           return pt_aux2;
                         }
                      else {
                            (*pt_aux2).prev = (*pt_aux).prev;
                            (*((*pt_aux).prev)).next = pt_aux2;
                            (*pt_aux).prev = pt_aux2;
                            (*pt_aux2).next = pt_aux;
                            return pt;
                           }
          }
       }

}


pt_prot delete_car_prot(pt,pt_aux,del)
pt_prot pt;
pt_prot pt_aux;
int del;
{
  pt_prot pt_aux2;

  pt_aux2 = (*pt_aux).next;
  if (pt_aux == pt)
    {
      if (pt_aux2 == pt)
        {
      if (del) free(pt_aux);
      return (pt_prot)NULL;
    }
      else {
             (*pt_aux2).prev = (*pt_aux).prev;
             (*((*pt_aux).prev)).next = pt_aux2;
             if (del) free(pt_aux);
             return pt_aux2;
           }
    }
  else {
          (*pt_aux2).prev = (*pt_aux).prev;
          (*((*pt_aux).prev)).next = pt_aux2;
          if (del) free(pt_aux);
          return pt;
        };

}


pt_prot selection_TZ_prot(pt)
pt_prot pt;
{
  pt_prot pt_aux;
  pt_prot pt_aux2;

  pt_aux = pt;
  pt_aux2 = pt;


  while ( (pt_aux != (pt_prot)NULL) && ((*pt_aux).ITZ == 0) )
    {
      pt_aux = delete_car_prot(pt_aux2,pt_aux,1);
      pt_aux2 = pt_aux;
    }

  if (pt_aux != (pt_prot)NULL)
  pt_aux = (*pt_aux).next;

  while ((pt_aux != (pt_prot)NULL) && (pt_aux != pt_aux2))
    if ((*pt_aux).ITZ == 0)
      if ((*pt_aux).status == 2)
        pt_aux = delete_car_prot(pt_aux2,pt_aux,1);
      else {
             (*pt_aux).status = 1;
             pt_aux = (*pt_aux).next;
           }
    else pt_aux = (*pt_aux).next;

  return pt_aux2;
}



pt_prot trans(pt)
pt_prot pt;
{
 pt_prot pt_aux=pt;
 double rand;

 if (pt != (pt_prot)NULL)
```

```
    {
      rand = random_double(1);
      if ( (rand <= (*pt_aux).thres) && ((*pt_aux).status == 0))
      (*pt_aux).status = 1;
      pt_aux = (*pt_aux).next;

      while (pt_aux != pt)
        {
          rand = random_double(1);
          if ( (rand <= (*pt_aux).thres) && ((*pt_aux).status == 0))
             (*pt_aux).status = 1;
          pt_aux = (*pt_aux).next;
        }
    }
  return pt;
}



void random_time(pt,pt_aux,N)
pt_prot pt;
pt_prot pt_aux;
int N;
{
  pt_prot pt_aux2=pt_aux;

  (*pt).nexttrans = random_int(N-1);
  (*pt_aux2).nexttrans = random_int(N-1);
  pt_aux2 = (*pt_aux2).next;
  while (pt_aux2 != pt_aux)
    {
      (*pt_aux2).nexttrans = random_int(N-1);
      pt_aux2 = (*pt_aux2).next;
    }
}



pt_prot compare_times(pt,pt_aux,same)
pt_prot pt;
pt_prot pt_aux;
int *same;
{
  int pt_time=(*pt).nexttrans;
  pt_prot pt_aux2=pt_aux;
  pt_prot pt_aux3=pt_aux;

  *same=0;
  while (     (pt_aux2 != (pt_prot)NULL)
          && ((*pt_aux2).nexttrans != pt_time) )
    {
      pt_aux2 = delete_car_prot(pt_aux3,pt_aux2,1);
      pt_aux3 = pt_aux2;
    }

  if (pt_aux2 != (pt_prot)NULL)
    {
      *same = 1;
      pt_aux2 = (*pt_aux2).next;
    }

  while ( (pt_aux2 != (pt_prot)NULL) && (pt_aux2 != pt_aux3))
    {
      if ((*pt_aux2).nexttrans == pt_time)
        {
          *same=1;
          pt_aux2 = (*pt_aux2).next;
        }
      else pt_aux2 = delete_car_prot(pt_aux3,pt_aux2,1);
    }

  return pt_aux3;
}



void delete_all_prot(pt,pt_aux)
pt_prot pt;
pt_prot pt_aux;
{
  free(pt);
  while (pt_aux != (pt_prot)NULL)
    pt_aux = delete_car_prot(pt_aux,pt_aux,1);
}



int count(pt)
pt_prot pt;
{
  pt_prot pt_aux=pt;
  int res;

  if (pt == (pt_prot)NULL)
  return 0;
  else {
         pt_aux = (*pt_aux).next;
     res = 1;
     while (pt_aux != pt)
       {
        res = res + 1;
        pt_aux = (*pt_aux).next;
      }

    return res;
      }
}




int analyze_single_car(pt,interval,trial_all)
pt_prot pt;
int interval;
int *trial_all;
{
  pt_prot pt_aux=pt;
  pt_prot pt_car=pt;
  int same_time=0;
  int res=0;
  int nb_car;
  int nb_coll;

  if (pt == (pt_prot)NULL)
    {
      *trial_all = 0;
      return res;
    }
  else res = 1;

  pt_aux = delete_car_prot(pt,pt,0);
  nb_car = count(pt_aux);
  nb_coll = nb_car + 1;

  if (pt_aux != (pt_prot)NULL)
    {
      do
        {
          random_time(pt,pt_aux,interval);
          nb_car = count(pt_aux);
          pt_aux = compare_times(pt,pt_aux,&same_time);
      res = res + 1;
          nb_coll = nb_coll + nb_car + 1;
        }
      while (same_time);
    }

  delete_all_prot(pt,pt_aux);
  *trial_all = nb_coll;
  return res;
```

```
}



int analyze_trans(pt,interval,trial_all)
pt_prot pt;
int interval;
int *trial_all;
{
  pt_prot pt_aux=pt;
  pt_prot pt_analyze=(pt_prot)NULL;
  int res;

  if (pt != (pt_prot)NULL)
    {
      if ((*pt_aux).status == 1)
        pt_analyze = add_car_prot((*pt_aux).nber,0.0,pt_analyze);
      pt_aux = (*pt_aux).next;

      while (pt_aux != pt)
        {
          if ((*pt_aux).status == 1)
            pt_analyze = add_car_prot((*pt_aux).nber,0.0,pt_analyze);
          pt_aux = (*pt_aux).next;
        }
      res = analyze_single_car(pt_analyze,interval,trial_all);
      return res;
    }
  else return 0;
}



pt_prot update_trans(pt)
pt_prot pt;
{
  pt_prot pt_aux=pt;

  if (pt != (pt_prot)NULL)
    {
      if ((*pt_aux).status == 1)
        (*pt_aux).status = 2;
      pt_aux = (*pt_aux).next;

      while (pt_aux != pt)
        {
          if ((*pt_aux).status == 1)
            (*pt_aux).status = 2;
          pt_aux = (*pt_aux).next;
        }
    }

  return pt;
}


pt_prot init_trans(pt)
pt_prot pt;
{
  pt_prot pt_aux=pt;
  double rand;

  if (pt != (pt_prot)NULL)
    {
      rand = random_double(1);
      if (rand <= (*pt_aux).thres)
        (*pt_aux).status = 2;
      pt_aux = (*pt_aux).next;

      while (pt_aux != pt)
        {
      rand = random_double(1);
          if (rand <= (*pt_aux).thres)
            (*pt_aux).status = 2;
          pt_aux = (*pt_aux).next;
    }
    }

  return pt;
}




void trait_protocol(max,file_in,col_lim,res1,res2,distance,interval,cars)
int max;
char *file_in;
int col_lim;
int *res1;
int *res2;
double distance;
int interval;
int *cars;
{
  FILE *input;
  pt_prot pt=(pt_prot)NULL;
  double aux;
  int i,j;
  int L;
  int C;
```

```
tcar *zone;
tcar *pt_vehicle;
int trial=0;
int trial_all=0;
int k;
int nb_cars;

input = fopen(file_in,"r");
fscanf(input,"%d",&L);
fscanf(input,"%d",&C);
fscanf(input,"%lf",&aux);
fscanf(input,"%lf",&aux);

zone = (tcar *) calloc(L*C,sizeof(tcar));
k = 0;
nb_cars = 0;
while ((!feof(input)) && (nb_cars < max))
  {
    for (i=0;(i<L);i++)
  for (j=0;(j<C);j++)
    { pt_vehicle = zone+i*C+j;
      fscanf(input,"%d ",&((*pt_vehicle).nber));
      fscanf(input,"%d ",&((*pt_vehicle).speed));
      fscanf(input,"%lf ",&((*pt_vehicle).move));
    }

    pt = ITZ_zero(pt);
    for (i=0;i<L;i++)
    for (j=0;j<col_lim;j++)
      {
    pt_vehicle = zone+i*C+j;
    if ((*pt_vehicle).nber != 0)
       pt = add_car_prot((*pt_vehicle).nber,
                         (j+(*pt_vehicle).move)/distance,pt);
  }

    if (k == 0) pt = init_trans(pt);
    pt = selection_TZ_prot(pt);
    pt = trans(pt);
    trial = analyze_trans(pt,interval,&trial_all);
    pt = update_trans(pt);
    if (trial != 0)
      nb_cars++;

    /* Mean of trials */
    *(res1+2*k)=trial;
    *(res1+2*k+1) = trial * trial;

    /* Repartition of trials */
    if (trial != 0)
       if (trial < max_trial)
```

```
          {
        (*(res2+(2*(trial-1))+1))++;
        (*(res2+(2*(trial-1))))++;
          }
        else {
               (*(res2+(2*(max_trial-1))))++;
           (*(res2+(2*(max_trial-1))+1))++;
             };

     k++;
   }

  fclose(input);
  free(zone);
  *cars = nb_cars;
}



int simul_PTime(max,LC,NL,WA,delta_t,t_max,speed_min,speed_max,CH,
                 trans_area,interval,result1,result2,file_hway)
int max;
int LC;
int NL;
int WA;
double delta_t;
double t_max;
int speed_min;
int speed_max;
int CH;
double trans_area;
int interval;
int *result1;
int *result2;
char *file_hway;
{
  int *res1;
  int *res2;
  int step_time;
  int i,j;
  int total;
  int col_lim;
  int cars;
  int cars_total;
  double distance;

  FILE *aux;

  step_time = integer_part(t_max/delta_t)+1;
  col_lim = integer_part(trans_area/LC);
  if (col_lim < trans_area/LC)
```

```
    col_lim++;
  distance = trans_area/LC;

  /* Mean of Trials */
  res1 = (int *) calloc(2*step_time,sizeof(int));

  /* Repartition of trials */
  res2 = (int *) calloc(2*max_trial,sizeof(int));

  cars_total = 0;
  while(cars_total < max)
    {
      aux = fopen("trials","a");
      fprintf(aux,"Interval=%d Total trial = %d\n",interval,cars_total);
      fclose(aux);

      simul_highway(LC,NL,WA,delta_t,t_max,speed_min,
                    speed_max,CH,file_hway,2,0);
      trait_protocol(max,file_hway,col_lim,res1,res2,distance,
                           interval,&cars);

      cars_total = cars_total + cars;

      /* Mean of Trials */
      for (j=0;j<2*step_time;j++)
        {
          *(result1+j) = *(result1+j) + *(res1+j);
    }

      /* Repartition of trials */
      for (j=0;j<2*max_trial;j++)
        {
      *(result2+j) = *(result2+j) + *(res2+j);
      *(res2+j) = 0;
    }

    }

  free(res1);
  free(res2);
  return cars_total;
}


void simul_PTraffic(max,LC,NL,WA,delta_t,t_max,speed_min,speed_max,CH,
                    trans_area,interval,file_out1,file_out2,file_hway)
int max;
int LC;
int NL;
int WA;
```

```
double delta_t;
double t_max;
int speed_min;
int speed_max;
int CH;
double trans_area;
int interval;
char *file_out1;
char *file_out2;
char *file_hway;

{
  int *tab1;
  int *tab2;
  int nb_col;
  int total;
  int total_sq;
  int res1;
  int i,j;
  int traffic_min;
  int traffic_max;
  int inter;
  int nb_loop;
  FILE *output1;
  FILE *output2;
  double mean;
  double var;

  printf("Traffic Min   (Vehicles/hour)  : ");scanf("%d",&traffic_min);
  printf("Traffic Max   (Vehicles/hour)  : ");scanf("%d",&traffic_max);
  printf("Interval                       : ");scanf("%d",&inter);

  output1 = fopen(file_out1,"w");
  fclose(output1);
  output2 = fopen(file_out2,"w");
  fclose(output2);

  nb_loop = (traffic_max-traffic_min)/inter + 1;
  nb_col = integer_part(t_max/delta_t)+1;


  for (i=0;i<nb_loop;i++)
    {

      tab1 = (int *) calloc(2*nb_col,sizeof(int));  /* Mean of Trials */
      /* Repartition of the Trials */
         tab2 = (int *) calloc(2*max_trial,sizeof(int));

      /*
      printf("Simulation For %d Vehicles/Hour\n",traffic_min + (i*inter));
      */
      res1 = simul_PTime(max,LC,NL,WA,delta_t,t_max,speed_min,speed_max,
                         (i*inter)+traffic_min,trans_area,interval,tab1,tab2,
             file_hway);

      /* Mean of Trials */
      total = 0;
      total_sq = 0;
      for (j=0;j<nb_col;j++)
        {
          total = total + *(tab1+2*j);
      total_sq = total_sq + *(tab1+2*j+1);
    };
      mean = total*1.0/res1;
      var = total_sq*1.0/res1 - (mean*mean);
      printf("Mean = %lf -- Variance = %lf\n",mean,sqrt(var));
      output1 = fopen(file_out1,"a");
      fprintf(output1,"%d %lf %lf\n",(i*inter)+traffic_min,mean,sqrt(var));
      fclose(output1);

      /* Repartition of Trials */
      output2 = fopen(file_out2,"a");
      fprintf(output2,"Traffic=%d ",(i*inter)+traffic_min);
      for (j=0;j<max_trial;j++)
        {
          mean = *(tab2+2*j)*1.0/res1;
          var = *(tab2+2*j+1)*1.0/res1 - (mean*mean);
          fprintf(output2,"Mean=%e ",mean*100);
        }
      fprintf(output2,"\n");
      fclose(output2);
      free(tab1);
      free(tab2);
    }
}




main(argc,argv)
int argc;
char *argv[];
{
  int LC = 5;                  /* in m */
  int WA = 40;                 /* in m */
  int NL = 4;
  double delta_t = 1;          /* in second */
  double t_max = 60;           /* in second */
```

```
int speed_min = 90;         /* in km/h   */
int speed_max = 120;        /* in km/h   */
int car_hour = 9200;        /* # of cars per hour  */
double trans_area = 15;     /* Width of the transmission aera */
int interval = 20;          /* # of slots for retransmission  */
char file_simul1[20];
char file_simul2[20];
char file_hway[20];
char file_trans[20];
int choice;
int max;

pt_prot pt=(pt_prot)NULL;
pt_prot pt_aux=(pt_prot)NULL;
int found;
double ave_trial;


printf("Length of the car         (m) : ");scanf("%d",&LC);
printf("Width of the Beacon Aera  (m) : ");scanf("%d",&WA);
printf("Number of Lane                : ");scanf("%d",&NL);
printf("Time Max                  (s) : ");scanf("%lf",&t_max);
printf("Interval of Time          (s) : ");scanf("%lf",&delta_t);
delta_t = integer_part(delta_t/step) * step;
printf("Speed Min              (km/h) : ");scanf("%d",&speed_min);
printf("Speed Max              (km/h) : ");scanf("%d",&speed_max);

printf("[1] : Simulation Highway\n");
printf("[2} : Statistics (Mean/Time)\n");
printf("[3] : Statistics (Mean/Traffic)\n");
printf("[4] : Protocol (Average Trials/Traffic)\n");
printf("[5] : Quit\n");
printf("Choice : ");scanf("%d",&choice);

if (choice == 1)
   {
     printf("Traffic        (Vehicles/hour) : ");scanf("%d",&car_hour);
     printf("Store Simul in ....            : ");scanf("%s",&file_simul1);
     printf("[1] Simulation From t=0\n");
     printf("[2] Simulation After Stabilisation\n");
     printf("Choice : ");scanf("%d",&choice);

     simul_highway(LC,NL,WA,delta_t,t_max,speed_min,
              speed_max,car_hour,&file_simul1,choice,1);
     visualization(&file_simul1);
   }
else if (choice == 2)
   {
     printf("Traffic        (Vehicles/hour) : ");scanf("%d",&car_hour);
     printf("Store Stat  in ....            : ");scanf("%s",&file_simul1);
     printf("Number of Iteration            : ");scanf("%d",&max);
```

```
        printf("[1] Simulation From t=0\n");
        printf("[2] Simulation After Stabilisation\n");
        printf("Choice : ");scanf("%d",&choice);

        simul_MTime(max,LC,NL,WA,delta_t,t_max,speed_min,
                       speed_max,car_hour,&file_simul1,1,choice);
      }
    else if (choice == 3)
           {
             printf("Store Stat  in ....          : ");scanf("%s",&file_simul1);
             printf("Number of Iteration          : ");scanf("%d",&max);
             simul_MTraffic(max,LC,NL,WA,delta_t,t_max,speed_min,
                              speed_max,&file_simul1);
           }
     else if (choice == 4)
            {
              printf("Store Stat Mean in ....              : ");
              scanf("%s",&file_simul1);
          printf("Store Stat Repartition in ....     : ");
              scanf("%s",&file_simul2);
          printf("Store Hway  in ....                : ");
              scanf("%s",&file_hway);
          printf("Maximum Simulation Trials          : ");
          scanf("%d",&max);
          printf("Width of the transmission aera     : ");
          scanf("%lf",&trans_area);
          printf("Interval of Retransmission         : ");
          scanf("%d",&interval);
          simul_PTraffic(max,LC,NL,WA,delta_t,t_max,speed_min,
                         speed_max,car_hour,trans_area,interval,
                         file_simul1,file_simul2,file_hway);
               };
}
```

CHAPTER III

A POSITION ESTIMATION ALGORITHM FOR VEHICLE FOLLOWING

A. Introduction

We look at the problem of vehicle following where an autonomous vehicle tracks the trajectory of a leading vehicle. Several approaches have been proposed to guide autonomous vehicles, such as computer vision [1] and vehicle lateral control systems [2]. The problems of computer vision are that the camera is expensive and the processing time is very slow. For the vehicle lateral control system, the installation of magnetic markers is time consuming and rather cumbersome. This proposed distance estimation system does not require very expensive equipment but simple hardware, and the processing time is short.

The idea is to install two transceivers at the back of the preceding vehicle and two transceivers at the front of the following vehicle. The two different transmitters at the front end of the following vehicle continually cycle through two different data sequences. After the receivers at the back of the preceding vehicle receive these two different data sequences they send them back to the receivers at the front end of the following vehicle. Once the transceivers at the front end of the following vehicle receive the sequences, the cross-correlation of the sending sequences with the received sequences is calculated. When the cross-correlator output reaches a peak value, the time difference between these sequences is the time delay for the sequence to travel back and forth between the two vehicles. Estimating the traveling time of these data sequences between transceivers, four different distances between the preceding vehicle and the following vehicle can be measured. From these four estimated distances, the turn angle of the preceding vehicle can be calculated using basic trigonometry. Taking the estimated distance and turn angle into account the following vehicles can adjust their speed and change their direction to follow the preceding vehicle. Since data sequences with good autocorrelation characteristic are needed, binary maximal-length sequences (m-sequences) have been used.

B. The minimum variance of time delay estimation

Since the time delay used to calculate the distance is an estimated value, from estimation theory there exists a minimum variance of this time delay estimate, given by the Cramer-Rao bound. Let $m(t)$ be the transmitted signal which is an m-sequence (here we use m=3). Then [3]

$$m(t) = \sum_{i=0}^{6}(2x_i - 1)\psi(t - iT_c), \tag{B.1}$$

where x_i is the binary data sequence, and $\psi(t-iT_c)$ is a pulse of duration T_c normalized such that

$$\frac{1}{T_c}\int_0^{T_c} \psi^2(t)\, dx = 1. \tag{B.2}$$

The Cramer-Rao bound for an estimate of arrival time is [4]

$$\sigma_{\hat{\tau}}^2 \geq \left\{ \frac{2}{N_0} \int_{-T/2}^{T/2} \left[\frac{\partial m(t-\tau)}{\partial \tau} \right]^2 dt \right\}^{-1}, \tag{B.3}$$

where τ is the time difference between the transmitting sequences and the received sequence, and T is the period of $m(t)$. Since $\tau \approx 0$, we may replace (B.3) by

$$\sigma_{\hat{\tau}}^2 \geq \left\{ \frac{2}{N_0} \int_{-T/2}^{T/2} \left[\frac{\partial m(t)}{\partial t} \right]^2 dt \right\}^{-1}. \tag{B.4}$$

Now assume $m(t)$ has a Fourier transform denoted by $M(j\omega)$; then $dm(t)/dt$ has a Fourier transform given by $j\omega M(j\omega)$. By Parseval's theorem, it follows that

$$\int_{-T/2}^{T/2} \left[\frac{\partial m(t)}{\partial t}\right]^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} \omega^2 \mid M(j\omega) \mid^2 d\omega, \tag{B.5}$$

and the minimum variance in (B.4) may be expressed as

$$\sigma_{\hat{\tau}}^2 \geq \left[\frac{1}{N_0 \pi} \int_{-\infty}^{\infty} \omega^2 \mid M(j\omega) \mid^2 d\omega\right]^{-1}. \tag{B.6}$$

When the sending sequence is 0010111, we have

$$m(t) = A[-\psi(t)-\psi(t-T_c)+\psi(t-2T_c)-\psi(t-3T_c)+\psi(t-4T_c)+\psi(t-5T_c)+\psi(t-6T_c)], \tag{B.7}$$

where A is the amplitude of single pulse. From $m(t)$ we have

$$\begin{aligned} M(j\omega) &= A[-\int_0^{T_c} e^{-j\omega t}dt - \int_{T_c}^{2T_c} e^{-j\omega t}dt + \int_{2T_c}^{3T_c} e^{-j\omega t}dt - \int_{3T_c}^{4T_c} e^{-j\omega t}dt \\ &\quad + \int_{4T_c}^{5T_c} e^{-j\omega t}dt + \int_{5T_c}^{6T_c} e^{-j\omega t}dt + \int_{6T_c}^{7T_c} e^{-j\omega t}dt] \qquad \text{(B.8)} \\ &= \frac{A}{-j\omega}[1 - 2\cos 2\omega T_c + 2\cos 3\omega T_c - 2\cos 4\omega T_c + \cos 7\omega T_c \\ &\quad + j(2\sin 2\omega T_c - 2\sin 3\omega T_c + 2\sin 4\omega T_c - \sin 7\omega T_c)], \qquad \text{(B.9)} \end{aligned}$$

$$\mid M(j\omega) \mid^2 = \frac{A^2}{\omega^2}(14 - 16\cos\omega T_c + 4\cos 2\omega T_c - 4\cos 5\omega T_c + 2\cos 7\omega T_c), \tag{B.10}$$

$$\int_{-\infty}^{\infty} \omega^2 \mid M(j\omega) \mid^2 d\omega = \int_{-\infty}^{\infty} A^2(14-16\cos\omega T_c+4\cos 2\omega T_c-4\cos 5\omega T_c+2\cos 7\omega T_c)d\omega. \tag{B.11}$$

Substituting (B.11) into (B.6) we obtain the minimum variance of estimated arrival time as

$$\sigma_{\hat{\tau}}^2 \geq \left[\frac{A^2}{N_0\pi}\int_{-\infty}^{\infty}(14 - 16\cos\omega T_c + 4\cos 2\omega T_c - 4\cos 5\omega T_c + 2\cos 7\omega T_c)d\omega\right]^{-1}. \tag{B.12}$$

If we truncate ω that outside $-\frac{2\pi}{T_c} \sim \frac{2\pi}{T_c}$, the Cramer-Rao bound becomes

$$\sigma_{\hat{\tau}}^2 \geq \left[(20.79)\frac{7A^2}{N_0\pi T_c}\right]^{-1}; \tag{B.13}$$

here the signal power P_s is

$$P_s = \frac{1}{T}\int_0^T m^2(t)dt = A^2. \tag{B.14}$$

C. Minimum variance of estimated distance

1. No loss of the received power

In this section a system without loss at the receiver end has been considered. The frequency band of this system is 850 $\sim$ 900MHz, data rate is 100 kbps, and the transmitted power is 1 mW. From [5] when the frequency band is 850 $\sim$ 900MHz the noise factor $F_a \approx 17$dB. Where F_a is [5]

$$F_a = 10\log(\frac{p_n}{kT_0 b}); \tag{B.15}$$

p_n is the average noise power, k is Boltzman's constant equaling $1.38 \times 10^{-23} J/^oK$, T_0 is the reference temperature (290^{o}K), and b is the receiver noise bandwidth equaling $\frac{1.2}{T_c}$. Using (B.15)

$$17 = 10\log\frac{p_n}{kT_0 b}$$

$$\begin{aligned} p_n &= 2.0057 \times 10^{-19} \times b \\ &= \frac{N_0}{2} \times b \\ N_0 &= 4.0114 \times 10^{-19}, \end{aligned} \tag{B.16}$$

where $N_0/2$ is the power spectral density of the noise. Substitute $T_c = \frac{1}{10^5}$, $A^2 = 10^{-3}$, and the result from (B.16) into (B.13) we can obtain

$$\begin{aligned} \operatorname{Var}[\hat{\tau} - \tau \mid \tau] &\geq \left[\frac{7 \times 10^{-3}(20.79)}{4.0114 \times 10^{-19} \times \pi \times T_c}\right]^{-1} \\ &= 8.66 \times 10^{-23} \end{aligned} \tag{B.17}$$

$$\hat{\sigma} \geq 9.31 \times 10^{-12}. \tag{B.18}$$

The calculated standard deviation for the estimated distance is 0.279 cm.

2. Consider loss of the received power

In this section free space loss, feeder loss, adaptor and connector losses will be taken into account to calculate the received power [6]. Free space loss for line-of-sight point-to-point radio link is [7]

$$L_{\text{dB}} = 32.44 + 20 \log D + 20 \log f, \tag{B.19}$$

where D is the path length in km, f is the frequency of the carrier in MHz. For our system assume the distance between two vehicles to be 5 meters and the carrier frequency to be 850 MHz, then the calculated free space loss is 45.01 dB. Assume the feeder loss is 0.23 dB, and the adaptor and connector loss is 1 dB; then the total loss at the receiver end is 46.24 dB. Given a transmitted power is 4 mW and a data rate of 800 kbits/sec, the calculated received power is 9.507×10^{-5} mW. So the Cramer-Rao bound becomes

$$\begin{aligned}\mathrm{Var}[\hat{\tau} - \tau \mid \tau] &\geq [\frac{7 \times 9.507 \times 10^{-8} \times 20.79}{4.0114 \times 10^{-19} \times \pi \times \frac{1}{8\times 10^5}}]^{-1} \\ &= 1.139 \times 10^{-19}. \qquad \text{(B.20)}\end{aligned}$$

So the standard deviation of the estimated distance is 5.061 cm.

D. Improved minimum variance of estimated distance

Since the minimum variance of the estimated distance we obtained in last section was too large, we need to reduce this minimum variance to an acceptable value. According to the results we got in Section B, in order to reduce the minimum variance we can either increase the transmitted power or energy per signal, or increase the data rate (decrease T_c). Since the transmitted power is very crucial for our system, in this section we try to increase the data rate and energy per signal (that is to increase the number of bits in the data sequence) in order to reduce the minimum variance of the estimated distance. Suppose we are using m-sequences of m=4; then the transmitted signal is

$$m(t) = \sum_{i=0}^{14} (2x_i - 1)\psi(t - iT_c), \qquad \text{(B.21)}$$

where $\psi(t - iT_c)$ is defined in (B.2). Then the Fourier transform of $m(t)$ is

$$M(j\omega) = \int_{-\infty}^{\infty} \sum_{k=0}^{14} (2x_k - 1)\psi(t - kT_c)e^{-j\omega t}dt. \qquad \text{(B.22)}$$

Let $X_k = 2x_k - 1$, $\Psi(j\omega)$ is the Fourier transform of $\psi(t)$, then

$$\mid M(j\omega) \mid^2 = \sum_{k=0}^{14} X_k e^{-j\omega kT_c} \sum_{l=0}^{14} X_l e^{j\omega lT_c} \mid \Psi(j\omega) \mid^2 . \qquad \text{(B.23)}$$

When the sequence is 000100110101111,

$$\begin{aligned} m(t) &= A[-\psi(t) - \psi(t-T_c) - \psi(t-2T_c) + \psi(t-3T_c) - \psi(t-4T_c) \\ &\quad -\psi(t-5T_c) + \psi(t-6T_c) + \psi(t-7T_c) - \psi(t-8T_c) \\ &\quad +\psi(t-9T_c) - \psi(t-10T_c) + \psi(t-11T_c) + \psi(t-12T_c) \\ &\quad +\psi(t-13T_c) + \psi(t-14T_c)]. \end{aligned} \tag{B.24}$$

From (B.23) we can get

$$\begin{aligned} \mid M(j\omega) \mid^2 &= (15 + 2\cos 2\omega T_c + 4\cos 3\omega T_c + 4\cos 4\omega T_c + 2\cos 6\omega T_c \\ &\quad -4\cos 7\omega T_c + 2\cos 8\omega T_c - 4\cos 9\omega T_c - 2\cos 10\omega T_c \\ &\quad -4\cos 11\omega T_c - 6\cos 12\omega T_c - 4\cos 13\omega T_c - 2\cos 14\omega T_c) \\ &\quad \times \mid \Psi(j\omega) \mid^2, \end{aligned} \tag{B.25}$$

where

$$\begin{aligned} \Psi(j\omega) &= \int_{-\frac{T_c}{2}}^{\frac{T_c}{2}} e^{-j\omega t} dt \\ &= \frac{2A \sin \omega \frac{T_c}{2}}{\omega}. \end{aligned} \tag{B.26}$$

Substituting (B.25) and (B.26) into (B.6), then caculate the integration from $-\frac{2\pi}{T_c} \sim \frac{2\pi}{T_c}$ we can get the Cramer-Rao bound of the time delay as

$$\sigma_{\hat{\tau}}^2 \geq \left[\frac{120A^2}{N_0 T_c}\right]^{-1}. \tag{B.27}$$

From Section C we have $A^2 = 9.507 \times 10^{-8}$ and $N_0 = 4.0114 \times 10^{-19}$; also data rate is $f_c = \frac{1}{T_c}$, substituting these into (B.27) we obtain the minimum standard deviation of time delay as

$$\sigma_{\hat{\tau}} \geq \frac{1}{\sqrt{2.844 \times 10^{13} \times f_c}}, \tag{B.28}$$

and the minimum standard deviation of the estimated distance as

$$\sigma_{\hat{d}} \geq \frac{1.5 \times 10^{10}}{\sqrt{2.844 \times 10^{13} \times f_c}}. \tag{B.29}$$

From (B.28) and (B.29) the minimum standard deviation of time delay and estimated distance of different data rates can be obtained and are shown in Table D. From Table

data rates	$\sigma_{\hat{\tau}} \geq$ (unit:sec)	$\sigma_{\hat{d}} \geq$ (unit:cm)
2M	1.326×10^{-10}	1.989
5M	8.386×10^{-11}	1.258
8M	6.230×10^{-11}	0.994
10M	5.930×10^{-11}	0.889
15M	4.842×10^{-11}	0.726
20M	4.193×10^{-11}	0.629

Table II. Standard deviation of time delay and distance versus data rate

D it is seen that when the data rate is above 8M bps a minimum standard deviation of estimated distance which is below 1 cm can be obtained.

E. Estimated angle

So far we only evaluated the performance of the estimated distance, in this section the estimated angle will be investigated.

1. Exact angle estimation-take one measurement

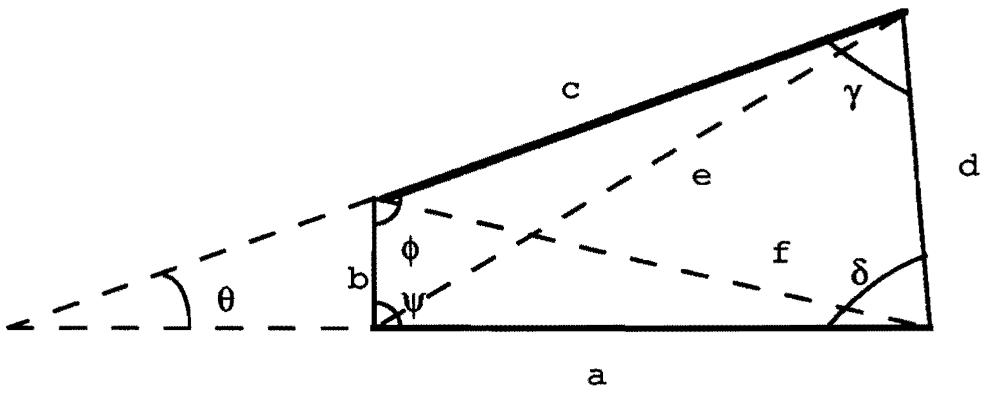


Fig. 12. Angle and distance relation between two vehicles. a is the distance between two antennas located on the back of preceding vehicle, b is the distance between the two antennas on the front of the following vehicle, and θ is the angle between the two vehicles.

In this subsection we will propose a model which can estimate the turn angle precisely. From Fig. 12 we can see that in order to estimate the angle between two cars exactly, we need not only know the length of a, b, c, d but also the length of e and f. The length of a and c is the distance between the two antennas which is under our control. The lengthes of b and d can be estimated from our proposed system. In order to estimate the distance of e and f one more correlator must be added to each receiver at the front end of the following vehicles. The purpose of this additional correlator is to correlate the data sequence transmitted from the diagonal transceiver in order to estimate the distance of e and f. From geometry

$$\begin{aligned} e^2 &= c^2 + b^2 - 2bc\cos\phi && \text{(B.30)} \\ f^2 &= a^2 + b^2 - 2ab\cos\psi && \text{(B.31)} \\ \theta &= 180^o - (180^o - \phi) - (180^o - \psi). && \text{(B.32)} \end{aligned}$$

From (B.30), (B.31), and (B.32) it's easy to get

$$\theta = \arccos\left(\frac{c^2 + b^2 - e^2}{2bc}\right) + \arccos\left(\frac{a^2 + b^2 - f^2}{2ab}\right) - 180^o. \qquad \text{(B.33)}$$

A computer simulation program has been developed to evalute the performance of this angular estimation problem. In this simulation we change ϕ from 90 degree to 130 degree and ψ from 90 degree to 140 degree respectively. Given the distances of a, b, and c we can caculate the value of e and f. Adding a Gaussian random variable to the value of b, e, and f we can abtain the simulating estimated distances. Substituting the estimated distances of b, e and f into (B.33) the estimated angle θ can be obtained. Square the difference of the exact angle and the estimated angle the variance of the estimated angle can be obtained. Given the variance of the estimated distance is 0.3, 0.4, 0.5, 0.6 the relationship between the variance of the estimated angle and the exact value of θ are plotted in Fig. 14. The computer program to calculate the variance of the angle is in Appendix C.

2. Exact angle estimation:multiple-measurements

Since the signal traveling time between cars is very short, we can afford to take more than one measurement then average all the measurements together in order to get more accurate angle and distance estimation. Since the vehicle driver control system response is slower then the communication system taking more measurements makes the control system response time more reasonable. Given the variance of the estimated distance the plots of variance of estimated angle versus exact angle for taking two, five, eight, ten measurements are in Fig. 15, Fig. 16, Fig. 17 and Fig. 18. The computer program to implement this simulation is in Appendix D. From the figures we can see that taking more measurements reduces the variance of the estimated angle.

3. Improved angle estimation:Utilize four estimated distances

So far we only utilize three distances (b, e, f) to get the estimated value of θ, but we have four estimated distances available. Taking the fourth estimated distance d into account, some kind of improvement can be expected. The relationship between θ and d, e, f is:

$$\theta = 180^{o} - \arccos\left(\frac{a^2+d^2-e^2}{2ad}\right) + \arccos\left(\frac{c^2+d^2-f^2}{2cd}\right). \tag{B.34}$$

In order to utilize (B.33) and (B.34) to calculate the values of θ, the exact values of b, d, e, and f should be known and this involves solving the following transcendental equations

$$f^2 = c^2+d^2-2cd\cos\gamma, \tag{B.35}$$

$$e^2 = a^2+d^2-2ad\cos\delta, \tag{B.36}$$

$$\gamma+\delta = 360^{o} - \phi - \psi, \tag{B.37}$$

where d, γ, and δ are unknown. Two numerical algorithms called *Müller* method and Secant method [8] have been used to solve the above transcendental equations. Since these two algorithms are very sensitive to the initial values, a separate program has been used to narrow down the initial value. It turns out the *Müller* method converges faster than the Secant method. Finally the *Müller* method has been chosen to solve the transcendental equations above to get the exact value of b, d, e, and f. After all the distances are known, (B.33) and (B.34) can be used to calculate two separate values of θ; averaging these two angles can improve the accuracy. The computer simulation to solve this problem is in Appendix E. This improvement is significant compared to using three distances, as shown in Fig. 19, Fig. 20, Fig. 21, Fig. 22.

4. Approximation model for angle estimation

Since the processing time of each estimation is so quick, the following vehicle can track the leading vehicle very closely, so the estimated angle is very small. Under this condition we propose a linear approximation model in Fig. 13. In Fig. 13 since θ is small, dx is approximately parallel to dy. Assume L is the distance between the two antennas; then

$$\sin\theta \approx \frac{dy - dx}{L} \approx \theta. \tag{B.38}$$

From probability theory, the variance of the estimated angle is equal to the sum of the variance of dx and dy divides by L^2. From the previous section when 15 bits/sequence and 20M bps was used, the standard deviation of the estimated distance is 0.629 cm, so the variance of the estimated angle is 1.978×10^{-5} and the standard deviation of the estimated angle is 4.448×10^{-3} radian. The accuracy of this approximation has been evaluated, and the result is in Appendix F. From the results in Appendix F we can see when the estimated angle θ is 5 degrees, the approximation error is only 1.2367×10^{-3}, which is very small.

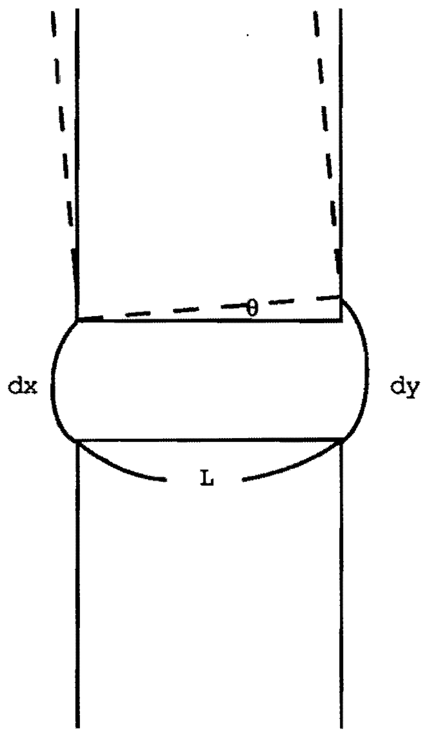


Fig. 13. Approximate model of angle and distance relation between preceding vehicle and following vehicle. Where the dash line represents the preceding vehicle after turning θ angle.

REFERENCES

[1] N. Kehtarnavaz, N. C. Griswold and J. K. Eem, "Real- time visual control for an intelligent vehicle - The convoy problem," *Proceedings of the SPIE Symposium on Image Processing and Artificial Intelligence*, Orlando, April 1990.

[2] Huei Peng and Masayoshi Tomizuka, "Preview Control for Vehicle Lateral Guidance in Highway Automation," *Proceedings of the American Control Conference*, Boston, June 1991.

[3] Dilip V. Sarwate & Michael B. Pursley, "Crosscorrelation Properties of Pseudo-random ande Related Sequences," *Proc. IEEE*, vol. 68, p. 593, 1980.

[4] Anthony D. Whalen, *Detection of Signals in Noise*, 1971.

[5] Edward N. Skomal, *Man-Made Radio Noise*, New York:Van Nostrand Reinhold, 1978.

[6] A. A. R. Townsend, *Digital Line-Of-Sight Radio Links*, Englewood Cliffs,NJ:Prentice Hall, 1988.

[7] Roger L. Freeman, *Telecommunication Transmission Handbook*, John Wiely & Sons, New York, 1975.

[8] Royce Beckett & James Hurt, *Numerical Calculations and Algorithms*, New York:McGraw-Hill, 1967.

APPENDIX C

THIS PROGRAM UTILIZE 3 ESTIMATED DISTANCES TO GET THE ESTIMATED ANGLE-TAKE ONE MEASUREMENT

```
C     THE MAIN PURPOSE OF THIS PROGRAM IS TO SIMULATE THE NONLINEAR
C     EQUATION IN THE TTI PROJECT
      INTEGER I1,I2,I3,I4,I6,II,NR,ISEED,COUNT(91),PSI,PHI,THETADE
      DATA PI/3.14159265/,AL/200.0/,CL/200.0/,M/1000/
      REAL*8 PSIRAD,PHIRAD,VAR1,VAR2,VAR3,VAR4,EL,FL,THETA,THETAR
      REAL*8 ELR,FLR,BLR,VARFNL(91),VARMID(91)
      REAL BL,RAD1

      ISEED=12345795
      OPEN(UNIT=10,FILE='TTISIM3N3.DAT',STATUS='NEW')
      PSI=89
      PSIRAD=89.0*PI/180
      RAD1=PI/180
      VAR4=0.0

      DO 10 II=1,90
        VARMID(II)=0.0
        COUNT(II)=0
10    CONTINUE

      DO 200 I1=1,41
        PSI=PSI+1
        PSIRAD=PSIRAD+RAD1
        PHI=90
        PHIRAD=PI/2
        VAR3=0

        DO 160 I2=1,50
          PHI=PHI+1
          PHIRAD=PHIRAD+RAD1

          THETADE=PHI+PSI-180

          COUNT(THETADE)=COUNT(THETADE)+1

          THETA=PHIRAD+PSIRAD-PI
          BL=300
          VAR2=0.0

          DO 120 I3=1,3
```

```
            BL=BL+30
            EL=DSQRT(CL**2+BL**2-2*BL*CL*DCOS(PHIRAD))
            FL=DSQRT(AL**2+BL**2-2*AL*BL*DCOS(PSIRAD))
            VAR1=0.0

            DO 80 I4=1,M
              ELR=EL+GAUSSD1(ISEED,0.5477)
              FLR=FL+GAUSSD1(ISEED,0.5477)
              BLR=BL+GAUSSD1(ISEED,0.5477)
              THETAR=DACOS((CL**2+BLR**2-ELR**2)/(2*BLR*CL))
     1          +DACOS((AL**2+BLR**2-FLR**2)/(2*AL*BLR))-PI
              VAR1=VAR1+(THETA-THETAR)**2/M
80          CONTINUE

            VAR2=VAR2+VAR1/3.0
120       CONTINUE

          VARMID(THETADE)=VARMID(THETADE)+VAR2

160     CONTINUE

200   CONTINUE

      DO 205 I6=1,90
        VARFNL(I6)=VARMID(I6)/COUNT(I6)
        WRITE (10,203) I6,VARFNL(I6)
203     FORMAT (1X,I4,2X,F10.7)
205   CONTINUE

      STOP
      END


      FUNCTION    GAUSSD1(ISEED,SD)
C~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~C
C   The function, given the mean and standard deviation of a disribution, C
C   returns a random number distributed according to a gaussian           C
C   distribution.                                                         C
C                                                                         C
C~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~C
      INTEGER*4 ISEED

      IF(ISET.EQ.0.)THEN
14      V1=2.0*RAN(ISEED)-1.0
        V2=2.0*RAN(ISEED)-1.0
        R=V1*V1+V2*V2

        IF((R.GE.1.0).OR.(R.EQ.0.0))GOTO 14

        FAC=SQRT(-2.0*LOG(R)/R)
        GSET=V1*FAC
```

```
  ISET=1
  GAUSSD1=V2*FAC*SD
  RETURN
ELSE
  ISET=0
  GAUSSD1=GSET*SD
  RETURN
ENDIF

END
```

APPENDIX D

THIS PROGRAM CALCULATE THE TURN ANGLE USING THREE DISTANCES AND TAKING MULTIPLE-MEASUREMENTS

```
      INTEGER I1,I2,I3,I4,I5,I6,II,NR,ISEED,COUNT(91),PSI,PHI,THETADE
      DATA PI/3.14159265/,AL/200.0/,CL/200.0/,M/1000/
      REAL*8 PSIRAD,PHIRAD,VAR1,VAR2,EL,FL,THETA,THETAR
      REAL*8 ELR,FLR,BLR,VARFNL(91),VARMID(91),TEM1,TEM2,TEM3
      REAL BL,RAD1

      ISEED=12345795
      OPEN(UNIT=10,FILE='TTISIM_15.DAT',STATUS='NEW')
      PSI=89
      PSIRAD=89.0*PI/180
      RAD1=PI/180

      DO 10 II=1,90
        VARMID(II)=0.0
        COUNT(II)=0
10    CONTINUE

      DO 200 I1=1,41
        PSI=PSI+1
        PSIRAD=PSIRAD+RAD1
        PHI=90
        PHIRAD=PI/2

        DO 160 I2=1,50
          PHI=PHI+1
          PHIRAD=PHIRAD+RAD1

          THETADE=PHI+PSI-180

          COUNT(THETADE)=COUNT(THETADE)+1

          THETA=PHIRAD+PSIRAD-PI
          BL=300
          VAR2=0.0

          DO 120 I3=1,3
            BL=BL+30
            EL=DSQRT(CL**2+BL**2-2*BL*CL*DCOS(PHIRAD))
            FL=DSQRT(AL**2+BL**2-2*AL*BL*DCOS(PSIRAD))
            VAR1=0.0

            DO 80 I4=1,M
```

```
              TEM1=0.0
              TEM2=0.0
              TEM3=0.0
              AVG=1
              DO 70 I5=1,AVG
                TEM1=TEM1+EL+GAUSSD1(ISEED,0.7071)
                TEM2=TEM2+FL+GAUSSD1(ISEED,0.7071)
                TEM3=TEM3+BL+GAUSSD1(ISEED,0.7071)
70            CONTINUE

              ELR=TEM1/AVG
              FLR=TEM2/AVG
              BLR=TEM3/AVG
              THETAR=DACOS((CL**2+BLR**2-ELR**2)/(2*BLR*CL))
     1          +DACOS((AL**2+BLR**2-FLR**2)/(2*AL*BLR))-PI
              VAR1=VAR1+(THETA-THETAR)**2
80          CONTINUE

            VAR2=VAR2+VAR1*1.094268783
120       CONTINUE

          VARMID(THETADE)=VARMID(THETADE)+VAR2


160     CONTINUE

200   CONTINUE

      DO 205 I6=1,90
        VARFNL(I6)=VARMID(I6)/COUNT(I6)
        WRITE (10,203) I6,VARFNL(I6)
203     FORMAT (1X,I4,2X,F10.7)
205   CONTINUE


      STOP
      END


      FUNCTION    GAUSSD1(ISEED,SD)
C-----------------------------------------------------------------------C
C   The function, given the mean and standard deviation of a disribution, C
C   returns a random number distributed according to a gaussian          C
C   distribution.                                                        C
C                                                                        C
C-----------------------------------------------------------------------C
      INTEGER*4 ISEED

      IF(ISET.EQ.0.)THEN
14      V1=2.0*RAN(ISEED)-1.0
        V2=2.0*RAN(ISEED)-1.0
```

```
          R=V1*V1+V2*V2

          IF((R.GE.1.0).OR.(R.EQ.0.0))GOTO 14

          FAC=SQRT(-2.0*LOG(R)/R)
          GSET=V1*FAC
          ISET=1
          GAUSSD1=V2*FAC*SD
          RETURN
        ELSE
          ISET=0
          GAUSSD1=GSET*SD
          RETURN
        ENDIF

        END
}
```

APPENDIX E

THIS PROGRAM CALCULATE THE TURN ANGLE USING FOUR DISTANCES AND TAKING MULTIPLE-MEASUREMENTS

```
C*****************************************************************
C   THE MAIN PURPOSE OF THIS PRCOGRAM IS GIVEN PSI, PHI, A, C     *
C   AND B THEN CALCULATE D.  THIS WILL INVOLVE SOLVING THE        *
C   TRANSCENDENTAL EQUATION INCLUDE THE VARIABLE ANGLE1, ANGLE2,  *
C   AND DISTANCE D.  THE METHOD USING HERE IS THE MULLER METHOD.  *
C                                                                 *
C   Comment: 1. The convergence of this method depends on the     *
C               initial value.  So one procedure that select      *
C               the approxiate initial has been used. Without     *
C               This procedure this method never converged.       *
C               Originally angle 1 changing by 1 degree has       *
C               been used to select the initial value, but        *
C               when the angle difference of PHI and PSI is       *
C               1 degree or when PHI and PSI are higher it        *
C               cannot converge.  After decrease the angle        *
C               diference to 0.125 then it can converge for       *
C               all value of PHI and PSI.                         *
C*****************************************************************

      INTEGER I1,I2,I3,I4,II,PSI,PHI,THETADE,API
      DATA PI/3.14159265/,AL/200.0/,CL/200.0/,LIMIT/1000/
      REAL*8 PSIRAD,PHIRAD,DELTA,Z1,Z2,Z3,THETAR,APIR,AN2,ZNEW
      REAL*8 FZ1,FZ2,FZ3,LA1,DE1,RDLT,EL2,FL2,G1,C1,LA,ZNEWD
      REAL*8 EPSO,EPSN,RAD1,RAD2,AN1R,AN2R,DT,FZN,FZO,ERR,B

      OPEN(UNIT=10,FILE='TTISIM_DNON.DAT',STATUS='NEW')
      PSI=89
      ERR=0.000001
      PSIRAD=89.0*PI/180
      RAD1=PI/180.0
      RAD2=PI/360.0
      RAD8=PI/1440.0
      RAD16=PI/2880.0

      DO 200 I1=1,41
        PSI=PSI+1
        PSIRAD=PSIRAD+RAD1
        PHI=90
        PHIRAD=PI/2

        DO 160 I2=1,50
          PHI=PHI+1
          PHIRAD=PHIRAD+RAD1

          API=360-PHI-PSI
          APIR=2*PI-PHIRAD-PSIRAD

          EL2=40000.0+90000.0-120000.0*DCOS(PHIRAD)
          FL2=40000.0+90000.0-120000.0*DCOS(PSIRAD)

          IF(PSI.EQ.PHI) THEN
            ZNEW=PI-PHIRAD
            ZNEWD=180-PHI
            AN2=API-ZNEWD
            B=200.0*DCOS(ZNEW)
            RDLT=B+DSQRT(B**2-40000.0+EL2)
            GOTO 115
          ENDIF

          AN1R=APIR
          AN1=API
          NUM=API*8-1
          FZO=1000000.0

          DO 60 I31=1,NUM
            AN1=AN1-0.125
            AN1R=AN1R-RAD8
            AN2R=APIR-AN1R
            DT=(FL2-EL2)/(400.0*(DCOS(AN2R)-DCOS(AN1R)))
            IF(DT.LT.0.0)GOTO 60
            FZN=DABS(FL2-40000.0-DT**2+400.0*DT*DCOS(AN1R))
            IF(FZN.LT.FZO) THEN
              FZO=FZN
              Z3=AN1R
            ENDIF
60        CONTINUE

              Z1=Z3-RAD16
              Z2=Z3+RAD16

          COUNT=0
          EPSO=1000000.0

80        DT1=(EL2-FL2)/(400*(DCOS(Z1)-DCOS(APIR-Z1)))
          DT2=(EL2-FL2)/(400*(DCOS(Z2)-DCOS(APIR-Z2)))
          DT3=(EL2-FL2)/(400*(DCOS(Z3)-DCOS(APIR-Z3)))
          FZ1=40000.0+DT1**2-FL2-400.0*DT1*DCOS(Z1)
          FZ2=40000.0+DT2**2-FL2-400.0*DT2*DCOS(Z2)
          FZ3=40000.0+DT3**2-FL2-400.0*DT3*DCOS(Z3)

          LA1=(Z3-Z2)/(Z2-Z1)
          DE1=(Z3-Z1)/(Z2-Z1)
          G1=(LA1**2)*FZ1-(DE1**2)*FZ2+(LA1+DE1)*FZ3
          C1=LA1*(LA1*FZ1-DE1*FZ2+FZ3)

          IF(G1.GE.0.0) THEN
            LA=0.0-(2*DE1*FZ3)/(G1+DSQRT(G1**2-4*DE1*C1*FZ3))
          ELSE
            LA=0.0-(2*DE1*FZ3)/(G1-DSQRT(G1**2-4*DE1*C1*FZ3))
          ENDIF
C      PRINT *,'FL2DIF=',FL2DIF

82        ZNEW=Z3+LA*(Z3-Z2)

          EPSN=DABS((ZNEW-Z3)/Z3)
          IF(EPSN.LT.ERR)GOTO 100

          IF(EPSN.GT.EPSO) THEN
            COUNT=COUNT+1
            IF(COUNT.GE.LIMIT) THEN
              WRITE(10,85) PSI,PHI
85            FORMAT(1X,'PSI=',4I,' PHI=',4I,'CANNOT CONVERGE')
              GO TO 100
            ENDIF
          ENDIF

          EPSO=EPSN
92        Z1=Z2
          Z2=Z3
          Z3=ZNEW

          GO TO 80

100       ZNEWD=ZNEW*180/PI
          AN2=API-ZNEWD
          RDLT=(EL2-FL2)/(400.0*(DCOS(ZNEW)-DCOS(APIR-ZNEW)))
115       WRITE(10,116) PSI,PHI,ZNEWD,AN2,RDLT
116   FORMAT(1X,'PSI=',I4,'PHI=',I4,'A1=',F12.8,'A2=',F12.8,'D=',F12.8)

160     CONTINUE

200   CONTINUE

      STOP
      END

}
```

APPENDIX F

THIS PROGRAM EVALUATE ANGLE ESTIMATION USING APPROXIMATION

```
/**********************************************************
  This program is used to caculate the angle approximation
 **********************************************************/

#include <stdio.h>
#include <math.h>
#define pi 3.14159265

main ()
{
 FILE *ffp;
 double theta1,theta2,theta11,theta12,a,b,c,d,e,thetaTrue,thetaApprox,tError;
 int i;

 theta1=pi/2;
 theta2=pi/2;
 a=200;
 b=500;

 ffp = fopen ("aapprox.d", "w");
 fprintf(ffp, "thetaTrue  thetaApprox  thetaError\n");

 for (i=1; i<=10; i++)
 {
   theta2 += pi/180;
   d=sqrt(pow(a,2)+pow(b,2)-2*a*b*cos(theta2));
   e=sqrt(pow(a,2)+pow(b,2)-2*a*b*cos(theta1));
   theta11=acos((pow(b,2)+pow(d,2)-pow(a,2))/(2*b*d));
   theta12=theta1-theta11;
   c=sqrt(pow(a,2)+pow(d,2)-2*a*d*cos(theta12));
   thetaTrue=theta2-theta1;
   thetaApprox=(c-b)/a;
   tError=fabs(thetaTrue-thetaApprox)/thetaTrue;
   fprintf(ffp, "%.8e %.8e %.8e\n",thetaTrue,thetaApprox,tError);
 }
}
```

```
 Accurate Angle  Accurate Angle  Approximate       Error
  (in degree)     (in radian)    Angle(radian)

     1           1.74532925e-02  1.74524074e-02  5.07104885e-05
 2          3.49065850e-02  3.48995663e-02  2.01071790e-04
 3          5.23598775e-02  5.23363205e-02  4.49905019e-04
 4          6.98131700e-02  6.97576246e-02  7.95628763e-04
 5          8.72664625e-02  8.71585376e-02  1.23672865e-03
 6          1.04719755e-01  1.04534221e-01  1.77172291e-03
 7          1.22173047e-01  1.21879935e-01  2.39915736e-03
 8          1.39626340e-01  1.39191040e-01  3.11760388e-03
 9          1.57079632e-01  1.56462991e-01  3.92565986e-03
10          1.74532925e-01  1.73691336e-01  4.82194764e-03
```

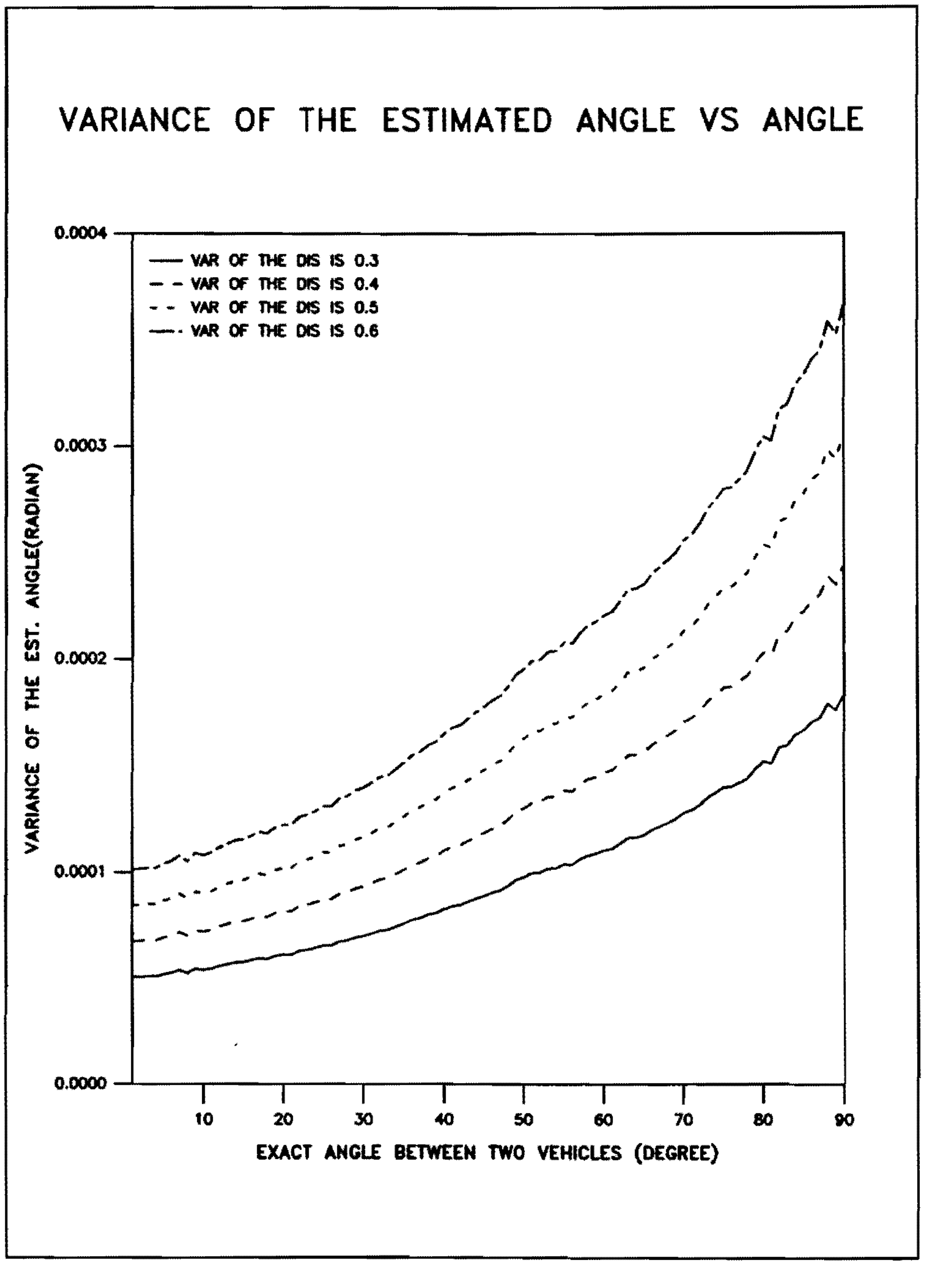


Fig. 14. The variance of the estimated angle for different variance of estimated distances.

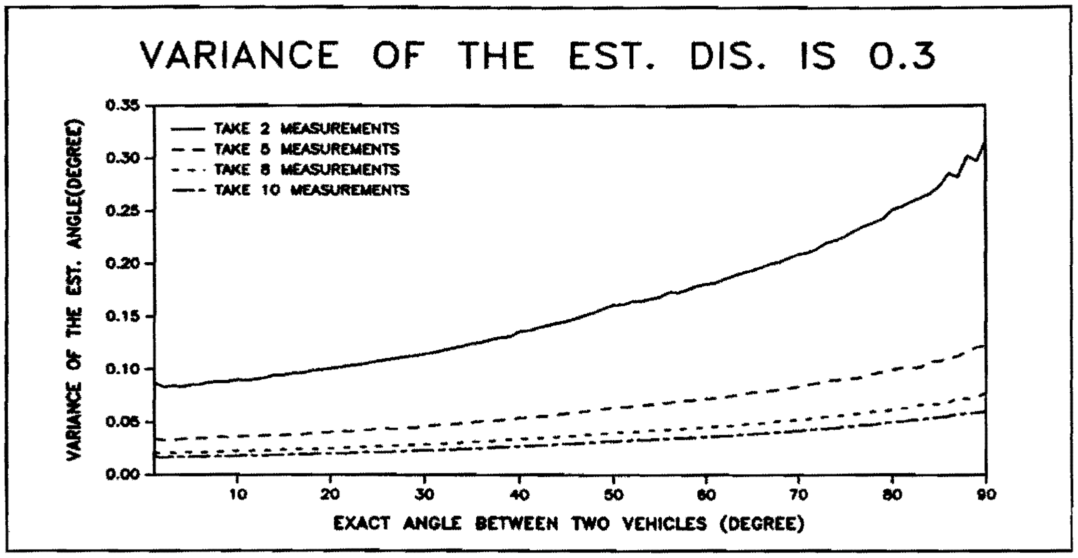


Fig. 15. Compare the variance of the estimated angle when taking more measurements, the variance of the estimated distance is 0.3.

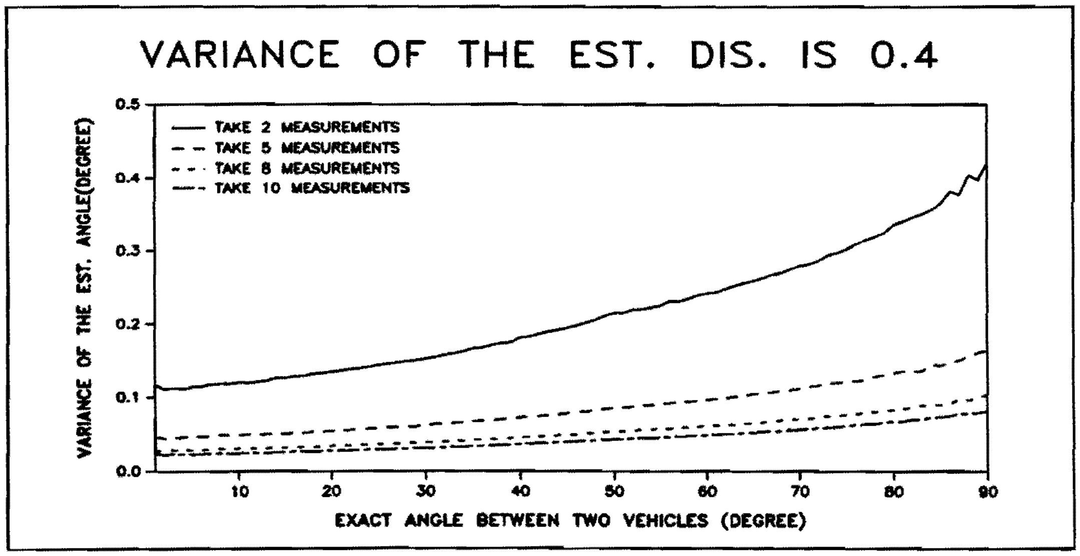


Fig. 16. Compare the variance of the estimated angle when taking more measurements, the variance of the estimated distance is 0.4.

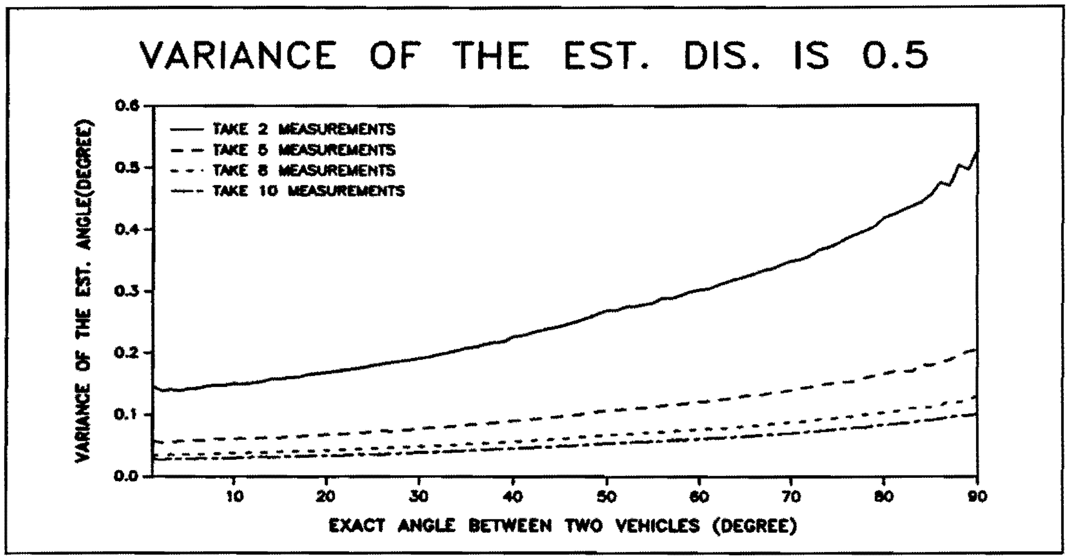


Fig. 17. Compare the variance of the estimated angle when taking more measurements, the variance of the estimated distance is 0.5.

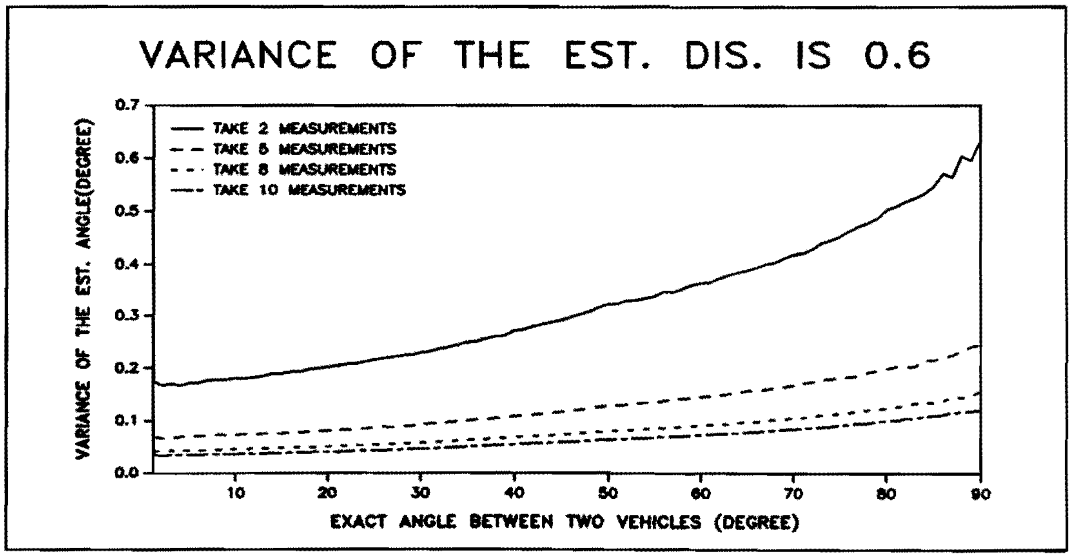


Fig. 18. Compare the variance of the estimated angle when taking more measurements, the variance of the estimated distance is 0.6.

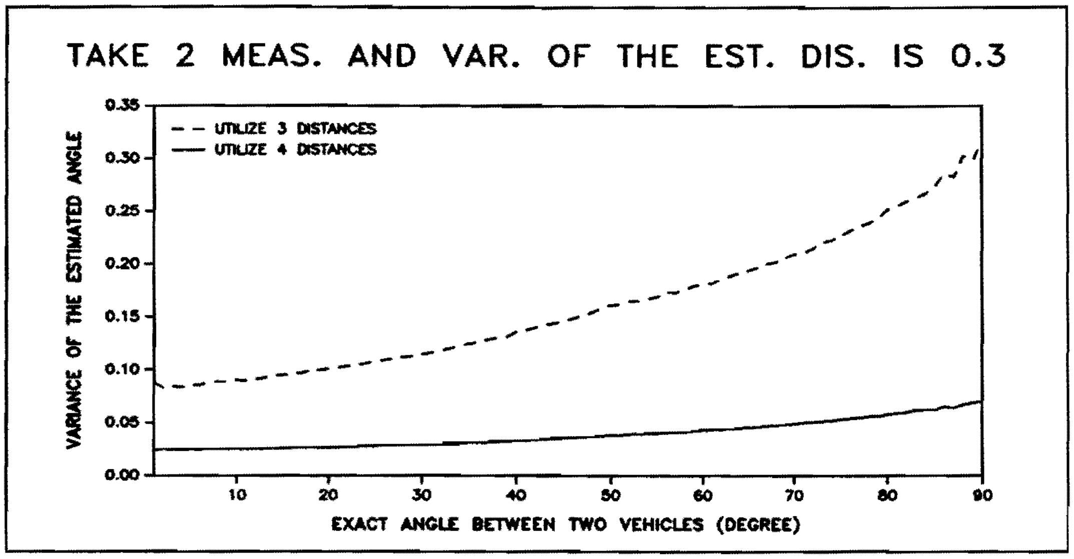


Fig. 19. Angle estimation using 3 distances compare with using 4 distances, the variance of the estimated distance is 0.3.

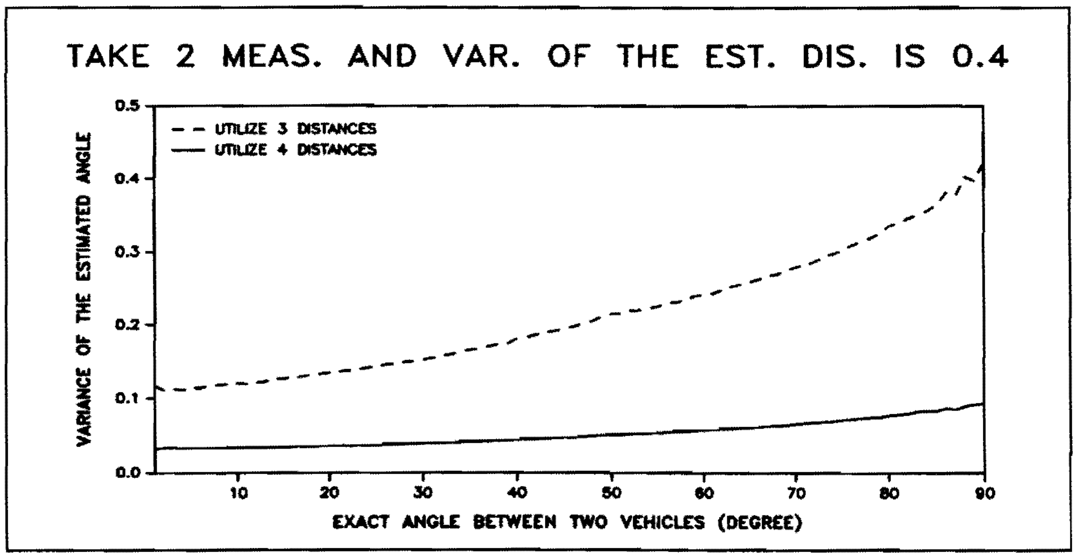


Fig. 20. Angle estimation using 3 distances compare with using 4 distances, the variance of the estimated distance is 0.4.

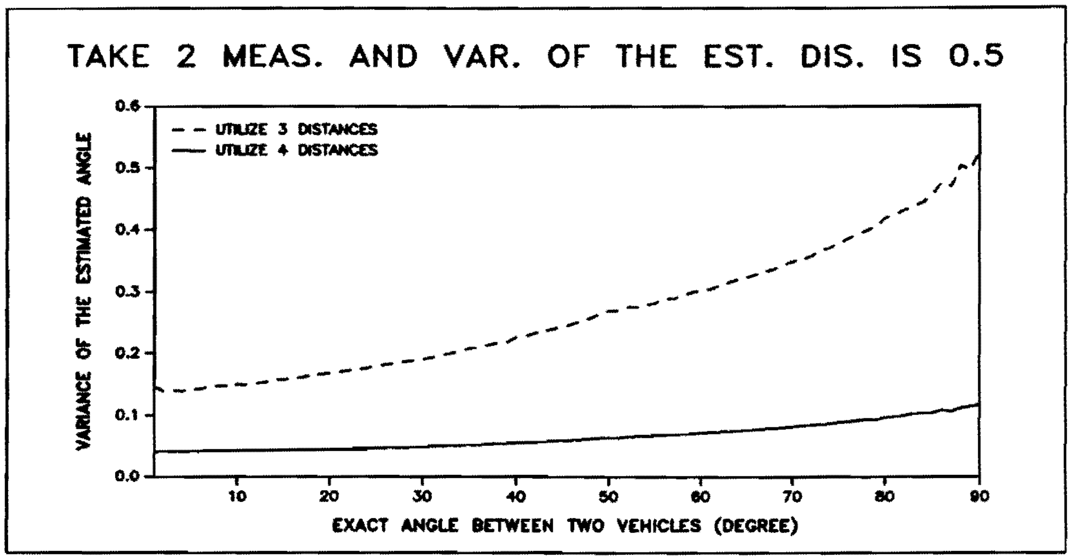


Fig. 21. Angle estimation using 3 distances compare with using 4 distances, the variance of the estimated distance is 0.5.

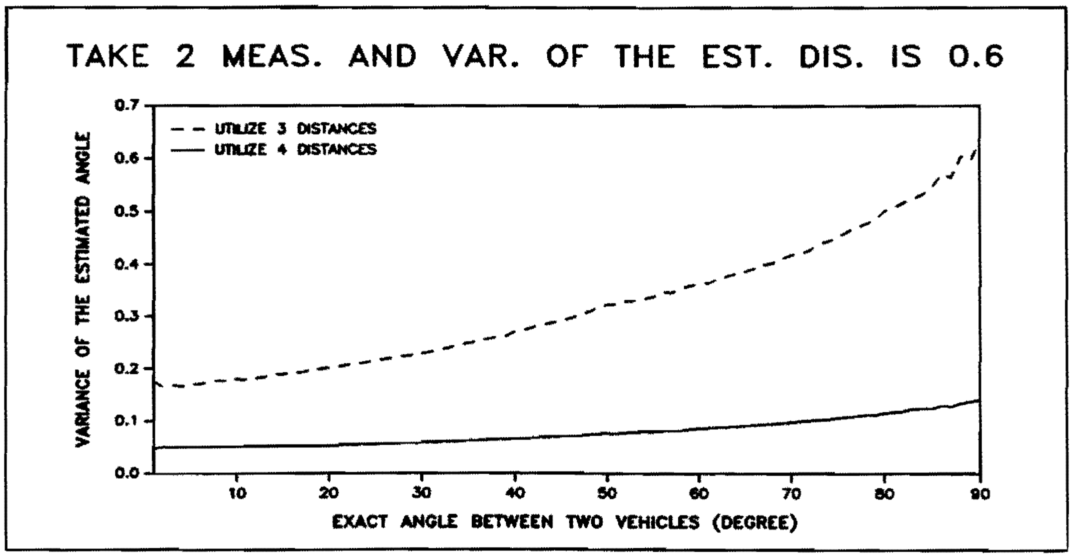


Fig. 22. Angle estimation using 3 distances compare with using 4 distances, the variance of the estimated distance is 0.6.

CHAPTER IV

TWO WAY MULTI-ACCESS COMMUNICATION USING INFRA-RED PRINCIPLES

A. Introduction

The need for a two-way (i.e., a scenario in which there is a to and fro communication between the moving vehicles and roadside beacons) communication has never been greater than today. With the increased traffic on the roads and the limited resources available, we feel the need to know the conditions (i.e., road and weather) for "good" travel on-line. With the large traffic, there is an acute necessity to have a communication link wherein more than one vehicle can have access to a given roadside beacon. This is termed as the "mutli-access facility" (i.e., a facility wherein more than one user on the highway, at any instant of time, can access a given roadside beacon, simultaneously).

It is now evident that regardless of any improvements in the utilization of the allotted radio frequency (RF) spectrum, there is little likelihood that enough RF spectrum will be available to meet the informational needs of future automated vehicles. The number of available channels is also limited.

There are many reasons why infrared signals are preferred [1].

- Simple possibility of lending required characteristics to infrared radiation lobes by lenses and reflectors
- Simple adjustability and alignment of optical transmitters and receivers
- Low susceptibility of influencing by neighbouring systems
- Low-cost production of transmitters and receivers
- No problems with frequency band allocation

Taking into account the present technology available to us, we feel the solution to increased demand, is the use of optical infrared principles.

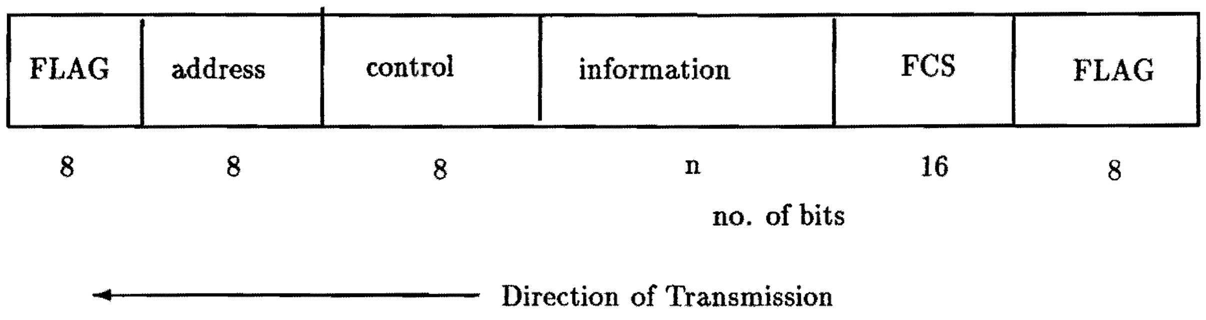
This report presents a preliminary investiagation of the above two-way, mutli-access infrared communication system. Our prime objective in this proposed work is to make a performance anaylsis of the above investigated communication link in terms of the maximum data rate that can be exchanged between the moving vehicle and the roadside beacon for a given bit error probability.

B. System Analysis

The proposed system here is based on the line-of-sight transmission principles. In this scheme optical radiation in the near infrared region is used as a signal carrier. Each vehicle is equipped with an LED (light Emitting Diode) and a photodiode for converting an electronic signal to an optical signal and vice versa. Reception of signals is nondirectional, i.e., the photodiodes receive the incoming radiation from a wide field of view (FOV).

Optical radiation from the beacon(downlink) reaches every vehicle uniformly, i.e., all the vehicles get the same information from the beacon. This forms a **Broadcast Channel**. Unique channels for the downlink and uplink are more practical to avoid interference problems. The radiation from the vehicles reaches the beacon as well as other vehicles. The uplink is thus a **Multi-Access Channel**. In any multi-access scheme or environment the problem of time sharing among all users arises. Access can be random or by polling. Polling process may be based on the SDLC (Synchronous Data Link Control) format. The SDLC is a bit oriented protocol developed by IBM. The bit oriented protocol frame format is given in the following page [6].

Bit Oriented Protocol Frame Format



FLAG : is the bit pattern 0111 1110, which marks the begining of the frame

address : contains the ID of sender or receiver of this frame

control : contains bits that describe the frame type and provide other information to control the flow of data

information : protocol data unit

FCS : frame check sequence, used for error detection

FLAG : is the bit pattern 0111 1110 which marks the end of frame

1. Power Analysis

Apart from the infrared signal, the receiver is also exposed to ambient light. A photon striking the photodiode produces with a given probability a hole-electron pair which is manifested as electrical current. Since this is a statistical process, the generation of the photocurrent during ambient light is associated with the noise which degrades the transmission quality. The analysis of the IR optical channel is based on the extensive work done Patarasen [5].

2. Choice of Receiver and Transmitter

The transmission rate for atmospheric propagation is largely limited by two factors:

- Ambient light fluctuations
- Response time of LED.

The ambient light constraints have been addressed extensively in [5]. We can have two types of diodes:

- Laser diodes
- LED.

The laser diodes can be used at higher modulation rates than the LED. But, the temperature dependence of laser thresholds complicate the driver circuitry of lasers relative to that of LED's. On the receiver side, two basic detector choices are:

- Avalanche photodiode
- PIN diode.

The PIN diode is favoured because of its tolerance to wide temperature fluctuations.

3. Data Transmission

Transmission can be realized by means of a high level data link control(HDLC) block format [1]. The signals issued by an HDLC protocol are as follows:- The HDLC protocol component's output retains its previous status when a log "1" is output. It changes its signal state only if a log "0" appears. A brief flash of light is emitted upon every signal transition from "0" to "1" and from "1" to "0".

4. Modulation Techniques

In this section the different modulation schemes which are applicable to the above mentioned communication link are investigated. In infra-red communication modulation could take place in two stages :

- Transmitted message modulates a carrier
- Modulated signal intensity (amplitude) modulates the emitted infra-red light.

Since the infra-red links suffer from extensive amplitude fluctuations, direct amplitude modulation by the message signal is not preferred. The effects of ambient light are reduced by modulating the transmitted infra-red signal.

5. Pulse Modulation Technique

In pulse position modulation (PPM) or pulse amplitude modulation (PAM) or pulse duration modulation (PDM) only time is expressed in discrete form, whereas the respective modulation parameters (namely, pulse amplitude, duration and position) are varied in a continuous manner in accordance with the message. In these modulation systems, information transmission is accomplished in analog form at discrete times.

6. Non Amplitude Modulation Systems

The most commonly used non-amplitude modulation systems are PSK and FSK systems. The modulated carrier method offers the possibility of having several separate optical channels for the uplink and downlink. In addition to this, receiver passband is shifted away from the baseband, thus allowing effective electrical filtering of ambient light fluctuations.

C. Multi-Access

As mentioned earlier, the uplink (which is the link from the vehicles to the roadside beacon) is a multi-access channel. In a multi-user environment, a multiple access method is needed to provide transmission capacity for individual users. The basic methods for multiple access are:

- FDMA (frequency division multiple access)
- TDMA (time division multiple access)
- CDMA (code division multiple access)

The first generation of multiple access communication systems has been dominated by FDMA systems. But, nowadays with the availability of precise clocks and high speed switching elements TDMA systems are favoured. In these schemes, the transmitted signals are orthogonal in frequency and time, respectively. FDMA is simple to implement but the channel bandwidth is not efficiently used except when all users are sending information simultaneously. In TDMA, all receivers are identical and monitor the same channel while the transmitters operate on different assigned slots of time.

CDMA is a method involving both time and frequency domains. In practice, CDMA offers user access flexibility but requires more complicated signal processing than TDMA or FDMA. As compared with TDMA, it is possible to implement CDMA without any sort of synchronization (due to the bursty nature of TDMA, synchronization among all users is required).

Since the broadcast channel is used by a single transmitter, no transmission conflicts will arise. We have to resolve conflicts which arise when simultaneous demands are placed on the channel. There are two obvious solutions to this problem:

- To form a queue of conflicting demands and serve them in the same order
- Lose any demands which are made while the channel is in use.

There are some controlling methods for channel multiple access in which access is not preassigned in time. There are two examples to the random access schemes which are suitable:

- ALOHA Systems [3]

In this system, *all* simultaneous demands made on the channel are lost.

- Carrier Sense Multiple Access with Collision Detection(CSMA -CD) [8]

Under the CSMA-CD protocol, every station wanting to transmit a packet must listen to the channel in order to find out whether any transmission is in progress. In spite of carrier sensing, packet collisions cannot be completely ruled out. Upon detection of a collision, transmission is aborted and the vehicle reschedules its message by selecting a random retransmission interval. To avoid accumulation of retransmissions, the retransmission interval can be dynamically adjusted to the actual traffic load.

In conclusion, the problem we are faced with is how to effect time-sharing of the multi-access channel among all users in a fashion which produces an acceptable level of performance.

REFERENCES

[1] Ronald Von Tomkewitsch, "Dynamic route guidance and interactive transport management with ALI-SCOUT," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 45-50, Feb. 1991.

[2] Fritz R.Gfeller, Urs Bapst, "Wireless in-house data communication via diffuse infrared radiation," *Proceedings of the IEEE*, vol. 67, pp. 1474-1486, Nov. 1979.

[3] Kleinrock L., "Packet switching in a multi-access broadcast channel: Performance evaluation," *IEEE Trans. Commun.*, vol. COM-23, pp. -, Apr. 1975.

[4] Kaveh Pahlavan, "Wireless Communications for Office Information Networks," *IEEE Communications Magazine*, vol. 23, No. 6, pp. 19-27, June 1985.

[5] Sittiporn Pataresan, "Infrared road-automobile communication," *Proceedings of the American Control Conference*, Boston, MA, June 1991.

[6] Scott A.Helmers, "*Data Communications*," Prentice Hall, 1989.

[7] H.T.Yura, "Physical model for strong optical-amplitude fluctuations in a turbulent medium," *J. Opt. Soc. Am.*, vol. 64,no. 1, pp. 59-67, Jan. 1974.

[8] Steven L.Beuerman and Edward J.Coyle, "The delay characteristics of CSMA-CD networks," *IEEE Trans. Commun.*, vol. 36, no. 5, pp. 553-563, May.1988.