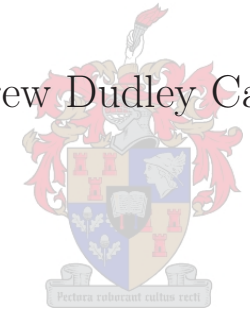
UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Adaptation, Optimisation and Simulation of the CSMA/CA Protocol for a Low Earth Orbit Satellite UHF Link

by

Andrew Dudley Cawood



Department of Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Dr R. Wolhuter

December 2006

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .....

A.D. Cawood

Date: .....

Abstract

A low earth orbit satellite is to provide the telecommunications link to facilitate e-mail services to rural areas, where the infrastructure necessary for e-mail is lacking (e.g. no telephone lines). Communication time with this satellite from any particular point on the ground is less than one hour per day. It is thus of utmost importance to maximise the data throughput rate for the system.

The contribution of this thesis is to improve the performance of CSMA/CA by adapting and optimising it for the above application. This improved protocol is used to regulate data flow through the system. Specific attention is given to the comparison of various random variable distributions for use as the back-off random variable.

Two pieces of software are further contributed. First, a set of MATLAB scripts which are used for comparing various back-off random variable distributions and optimising each of these distributions. Secondly, an extensive (more than 2500 lines of code) OMNeT++ simulation of the improved CSMA/CA protocol, complete with MATLAB scripts for setting up multiple simulation runs and plotting the results. Both pieces of software accept the system constraints as parameters, and are thus easily adaptable for a similar system which may use the same protocol, but has different parameters.

It is concluded that the set of MATLAB scripts are a fairly accurate tool for optimising throughput, as is confirmed by the OMNeT++ simulations, and that OMNeT++ has merit for simulating the given type of system and protocol.

Samevatting

'n Satelliet met 'n lae wentelbaan gaan gebruik word om 'n e-pos diens aan landelike streke te verskaf. Dié diens word op streke gemik waar daar 'n gebrek is aan die infrastruktuur wat vir e-pos benodig word (b.v. telefoonlyne). Daar kan daagliks, van enige gegewe punt op die aarde se oppervlak, vir 'n maksimum van een uur met die satelliet gekommunikeer word. Dit is dus hoogs belangrik dat die deurset van die stelsel so hoog moontlik is.

Die bydrae van hierdie tesis is om CSMA/CA aan te pas en te optimeer vir die bogenoemde toepassing, sodat sy werkverrigting verbeter word. Die aangepaste protokol word gebruik om die datavloei van die stelsel te beheer. Aandag word spesifiek aan die vergelyking van verskeie lukrake veranderlike verspreidings (wat vir die wag-tyd veranderlike gebruik word) gegee.

Daar word ook twee stukke sagteware bygedra. Die eerste is 'n stel MATLAB programme wat gebruik word om die verskillende wag-tyd verspreidings met mekaar te vergelyk, en om elkeen te optimeer. Die tweede is 'n omvattende OMNeT++ simulasie (meer as 2500 reëls kode) van die aangepaste CSMA/CA protokol. Dit sluit ook MATLAB programme in wat gebruik word om veelvoudige simulasies op te stel en die resultate in grafieke op te trek. Beide gedeeltes sagteware aanvaar die fisiese stelsel beperkings as parameters. Hulle is dus maklik aanpasbaar vir soortgelyke stelsels wat dieselfde protokol (maar ander parameters) gebruik.

Van die gevolgtrekkings wat gemaak word is dat:

a) Die MATLAB optimerings programme waardevol is om deurset te maksimeer. Dit word deur OMNeT++ simulasies bevestig.

b) Die OMNeT++ omgewing baie handig is vir die simulasie van die gegewe stelsel en protokol.

Acknowledgements

I would like to thank the following people for their inputs. Without them this thesis would not have been possible.

My study leader, Dr Riaan Wolhuter, for his guidance, advice, wisdom, time, enthusiasm and encouragement.

My gran, Dudu, for her thorough proof-reading of the bulk of this thesis. Any errors which may have crept in are most likely my later additions.

My family and the many friends who carried me in prayer, especially through the final six weeks of work.

My God, for His grace, and His creation - the mountains and sea which provided the fresh air and peace when my head needed clearing and my body could sit still no longer.

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	ACKnowledge
BER	Bit Error Rate
CRC	Cyclic Redundancy Check
CDMA	Code Division Multiple Access
CSMA	Carrier-Sense Multiple Access
CSMA/CA	Carrier-Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
CTS	Clear To Send
CTS-0	CTS with `dataLeft` field set to zero
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FES	Future Events Set
NAK	Not AcKnowledge
OMNeT++	Objective Modular Network Testbed in C++
PDF	Probability Density Function
RNG	Random Number Generator
RTS	Request To Send
RX	Receive
SNR	Signal to Noise Ratio

SSIS	SunSpace Information Systems
TDMA	Time Division Multiple Access
TX	Transmit
VCS	Virtual Carrier Sense

List of Symbols

Constants:

$k = \quad 1.38 \cdot 10^{-23} W.s.K^{-1}$

$c = \quad 299\,792\,458 m.s^{-1}$

$R_e = \quad 6378 km$

$T_o = \quad 290 K$

Chapter 1

Introduction

1.1 Background

On 23 February 1999 the Delta II rocket carrying the SunSat micro-satellite was launched from the Vandenberg Air Force base in California, USA. SunSat orbited the earth for two years relaying video and photographic images, before it ceased responding to communications on 19 January 2001. SunSat is a 60kg micro-satellite which was built and tested by twenty-two Masters students at the Electronic Systems Laboratory in the Department of Electrical and Electronic Engineering at Stellenbosch University. It was South Africa's first amateur radio and scientific satellite.

On 3 October 2005, during the launch of the South African chapter of the World Space Week, the Minister of Science and Technology, Mr Mosibudi Mangena, officially announced the ZASAT Pathfinder project. It is a three-year satellite development project involving collaboration between the Department of Science and Technology, Stellenbosch University, SunSpace & Information Systems (Pty) Ltd (SSIS) and the CSIR Satellite Application Centre (SAC), and has a budget of R26-million. The first outcome of this project is the development of the Sumbandila micro-satellite (previously known as ZASat Pathfinder 1, or ZA-002). *Sumbandila* is Venda for "the way ahead" or "pathfinder", and the name was chosen through a national competition for schools.

Sumbandila is an 80kg low earth orbit (LEO) satellite. Amongst the services it will provide are the following:

- Earth observation by means of on-board cameras
- A test platform for a Software Defined Radio (SDR) project of the University

of Stellenbosch, as well as other experiments from various universities

- E-mail access for rural communities by means of a UHF data link and an on-board mail server

E-mail Access

E-mail access is a service considered almost a basic necessity by millions of people worldwide, yet there are large areas of South Africa where people do not have access to such a luxury due to the lack of necessary infrastructure. The idea behind this leg of the satellite project is to make e-mail access available to many who are currently without it.

It is envisioned that user-level access will be provided through a central point in the rural community, equipped with a radio transceiver ground station and simple user terminals. E-mails will be relayed via a mail client on the Sumbandila micro-satellite, either to other similar stations or to a station which is connected to the Internet.

1.2 Objectives

The following are the objectives of this thesis:

- To make a study of existing protocols for multi-user single-channel environments
- To select the protocol best suited for this environment and, to a degree, the specific application of non-real-time rural communications
- To adjust and refine the protocol for specific application with a LEO satellite, such as Sumbandila
- To simulate the working protocol by means of a suitable simulation package, with the aim of estimating system characteristics, such as throughput
- To optimise the protocol for maximum throughput

1.3 Overview of Thesis

- Chapter 2 investigates the various physical constraints on the system which result from the medium in which communication takes place and the orbital

characteristics of the satellite. Where necessary these physical constraints are quantified.

- In Chapter 3 various existing multi-user single-channel protocols are investigated. The merits of each of the protocols are summarised, and the best-suited protocol is selected.
- Chapter 4 starts with the basic CSMA/CA protocol and details the adjustments and refinements made to the protocol, and the reasons for these improvements.
- Chapter 5 explores the concepts of error-detection and error-correction, and suggests their applicability for implementation with the defined protocol.
- The simulation of the protocol is presented in Chapter 6. It explains the structure of the code making up the simulation, and also explains how certain physical constraints were applied and implemented in software.
- In Chapter 7 the `vcs_added_backoff` parameter is singled out as being the most important for optimising throughput. MATLAB scripts for the optimisation of this parameter are presented, various random number distributions are considered, and the initial results are presented.
- Chapter 8 presents the results of the OMNeT++ simulation. It investigates the correlation between the two sets of results.
- Chapter 9 provides the conclusions that can be drawn from the thesis and suggests areas for further research.
- A list of references and several appendices are included. The latter contain flow-diagrams of the final protocol, data sheets of specific hardware and excerpts from code used for calculations.

There is a summary at the end of each chapter highlighting the findings.

1.4 Contributions

The following are the contributions of this thesis:

- A CSMA/CA-type protocol specifically adapted for LEO satellite applications, defined in flow-chart format

- A flag system for avoiding unnecessary packet transmissions
- An extensive working simulation (more than 2500 lines of code) of the above protocol, which allows the user to determine expected behaviour of the system and to estimate throughput and other statistical characteristics
- MATLAB scripts which automatically set up runs of the above simulation for a given range of values of certain parameters
- MATLAB code for the optimisation of the above protocol
- A Boolean-type random variable distribution for the back-off strategy
- A comparison of various random variable distributions for the back-off

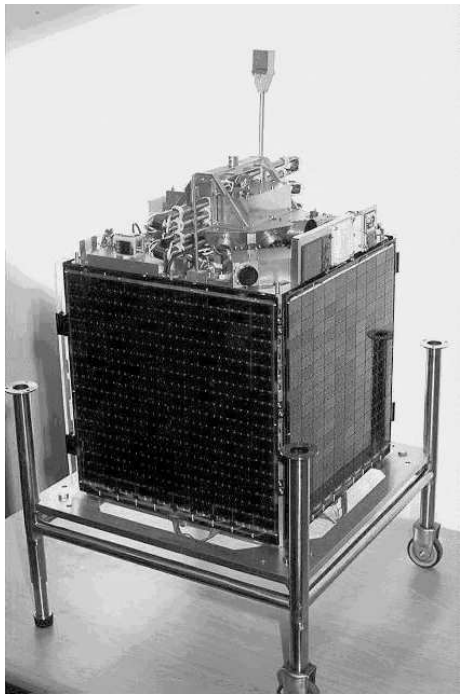


Figure 1.1: SunSat - the predecessor of Sumbandila (ZA-002)

Chapter 2

System Constraints

Before any decisions can be made, one first needs to establish what physical constraints and limitations there are on the system, and how these factors affect data transmission between ground stations and the satellite.

2.1 Given Information

SSIS indicated that the satellite's orbit will be circular, polar and sun-synchronous, and at an orbital altitude of approximately 600km.

2.2 Distance to Satellite

The distance between the satellite and any ground station with which it communicates is an important factor in calculating the signal strength loss over this path.

Communication in the UHF band takes place over line-of-sight links, and the signal is absorbed by mountains and the ground. The satellite can thus only communicate with ground stations that are in line-of-sight, and not with stations that are already over the horizon. It is important to know at what maximum distance from the centre of the satellite's footprint a ground station is still in sight of the satellite, and also what the direct, line-of-sight distance is.

The satellite will orbit at a height $h = 600km$ above the earth's surface. Because this value is only one order of magnitude smaller than the radius of the earth ($R_e = 6378km$), the curvature of the earth's surface must be taken into account. The problem can be solved making use of co-ordinate geometry.

Any circle can be described by the equation $(x-a)^2 + (y-b)^2 = r^2$, where (a, b)

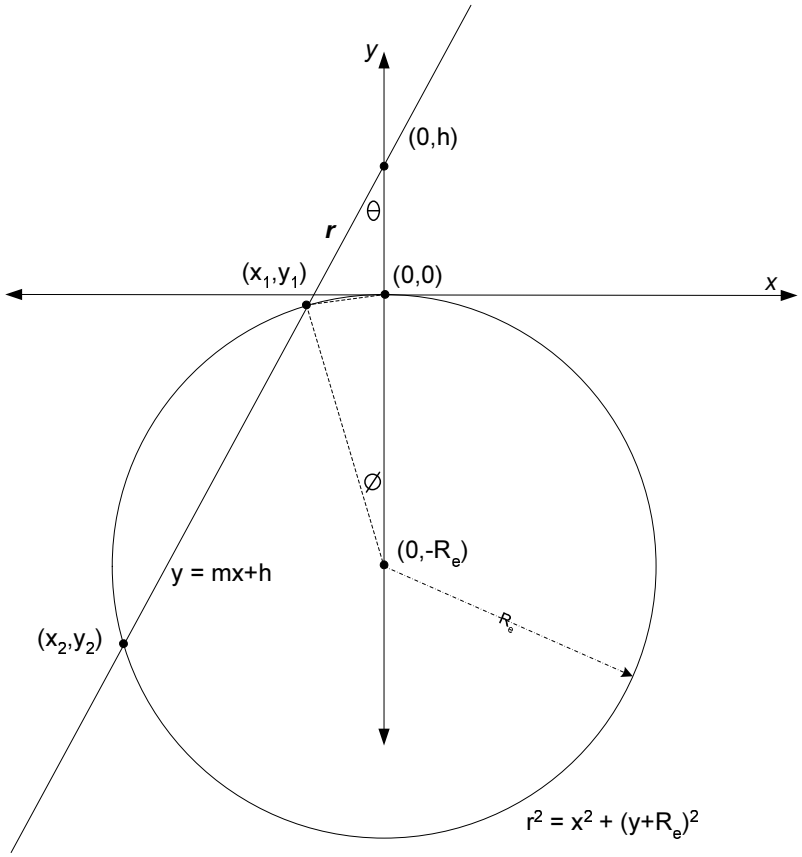


Figure 2.1: Co-ordinate geometry solution

is the centre of the circle and r is the radius. The earth will be described as

$$x^2 + (y + R_e)^2 = R_e^2 \tag{2.1}$$

and the satellite placed at position $(0, h)$. A line is drawn from the satellite through a ground station located at position (x_1, y_1) on the earth's surface. This line is described by

$$y = mx + h \tag{2.2}$$

θ and ϕ are defined as in Figure 2.1.

Substituting (2.2) into (2.1) we get

$$x^2 + (mx + h + R_e)^2 = R_e^2$$

This can be factorised and rewritten in the form of

$$ax^2 + bx + c = 0$$

where

$$a = 1 + m^2$$

$$b = 2m\left(h + R_e\right)$$

$$c = h^2 + 2hR_e$$

Using the General Formula

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.3}$$

we are able to solve for x in terms of m, and by substituting this back into (2.2) we are able to solve for y in terms of m. Finally, using Pythagoras, we can find that the distance r from the satellite to the ground station is

$$r = \sqrt{x_1^2 + \left(h - y_1\right)^2} \tag{2.4}$$

We know that $m = \frac{1}{tan\theta}$, which means that for any given value of θ we can calculate the distance r using the above formulae. We can sweep θ upwards from zero until the point where (2.3) no longer gives us a real result, and this will be at the furthest possible point for communication. All that now remains is to calculate the distance along the arc from $(0, 0)$to (x_1, y_1), which is the distance that the ground station is from the middle of the satellite's footprint.

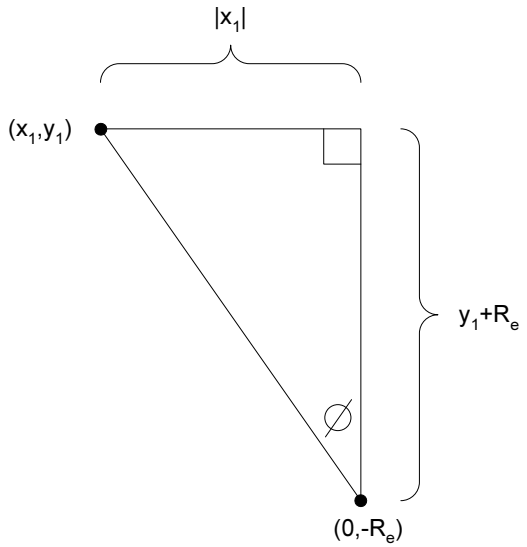


Figure 2.2: Calculation of phi

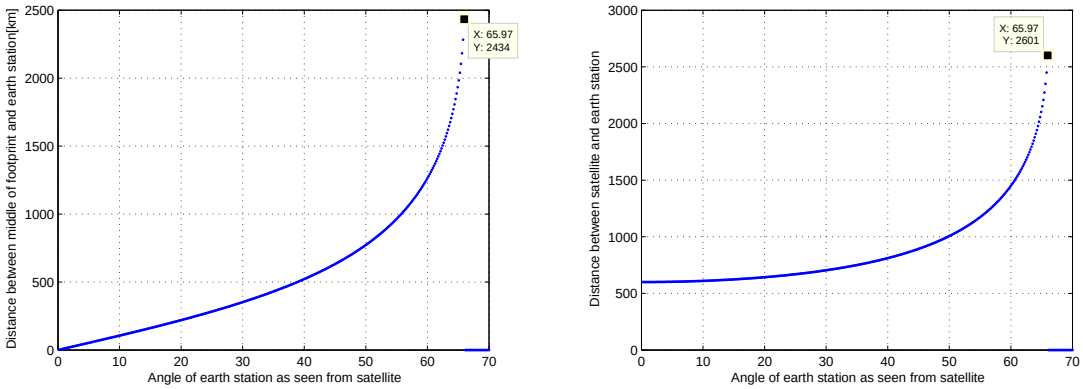
In Figure 2.2 it can be seen that we can calculate the value of ϕ from x_1 and y_1 by using

$$\phi = tan\left(\frac{y_1 + R_e}{|x_1|}\right)$$

Finally, d can be calculated as a fraction of the earth's circumference using

$$d = 2\pi R_e \times \frac{\phi}{360°}$$

A MATLAB script `pathlengths.m` was written to sweep the value of θ and plot the resulting values of d. The results are shown in Figure 2.3.



(a) Distance from centre of footprint

(b) Line-of-sight distance

Figure 2.3: Distance from satellite to ground station for given theta

The farthest point from the middle of the footprint at which communication may still be possible is at 2434km, which translates into a line-of-sight distance of 2601km. The signal-to-noise ratio (SNR) will determine whether communication is actually possible over this distance.

2.3 Propagation Time

Radio waves propagate at the speed of light. Propagation time for a station in the middle of the footprint is thus

$$t_p = \frac{600 \cdot 10^3}{c} = 2ms$$

and for a station at the edge of the footprint it is

$$t_p = \frac{2600 \cdot 10^3}{c} = 8.67ms$$

2.4 Orbital Calculations

The satellite's orbit is given as circular, polar and sun-synchronous.

Kepler and Newton's laws can be combined and a number of useful equations derived for calculations regarding the orbiting of (among other things) a satellite around the earth. A full explanation of orbiting principles can be found in [Campb-96], but, for now, we simply make use of the equations that they provide:

$$rev/day = 16.997 \left(\frac{R_e}{R_e + h} \right)^{\frac{3}{2}} \tag{2.5}$$

where rev/day is the number of times that the satellite orbits the earth per day, R_e is the radius of the earth, and h is the height of the satellite above the earth's surface.

$$t_{view} = \frac{2cos^{-1}\left(\frac{R_e}{R_e+h}\right)}{\frac{360}{88.49}\left(\frac{R_e}{R_e+h}\right)^{\frac{3}{2}} - \frac{260}{1436}} \tag{2.6}$$

where t_{view} is the maximum time in minutes which a satellite is visible from a fixed point on the earth during a single overhead pass, and the arc cosine is calculated in degrees.

Substituting the satellite's height into (2.5) gives:

$$rev/day = 16.997 \left(\frac{R_e}{R_e + 600} \right)^{\frac{3}{2}} = 14.85$$

or one revolution every 96.9 minutes. The circumference of the earth is

$$C_e = 2\pi R_e = 40074km$$

and South Africa is approximately 1600km in breadth and length. The pass time over South Africa would thus be approximately

$$\frac{1600}{40074} \times 96.9 = 3.86min$$

Substituting the height into (2.6) gives:

$$t_{view} = \frac{2cos^{-1}\left(\frac{R_e}{R_e+600}\right)}{\frac{360}{88.49}\left(\frac{R_e}{R_e+600}\right)^{\frac{3}{2}} - \frac{260}{1436}} = 14.18min$$

This is the viewing time from a single point. If we add to this the pass time across South Africa, then the maximum time in which the satellite can be viewed from South Africa is approximately 18 minutes. This is, of course, assuming a pass directly over South Africa. We will work with a conservative value of 12 minutes per pass in order to take passes just to the left and right of South Africa into account, when the viewing time (and thus communication time) will be less.

Although the satellite orbits at 14.85 times per day, it is important to know how often the satellite passes close enough to South Africa for communication to be possible. Consider that the earth turns through approximately 360° each 24 hours, and thus $\frac{360}{24} = 15°$ per hour, and $\frac{96.9}{60} \times 15 = 24.23°$ with each orbit of the satellite. This implies that if the satellite passes directly overhead, it will pass 24.23° West of us on the next pass.

It was calculated in Section 2.2 that a ground station cannot be more than approximately 2400km from the centre of the satellite's footprint if it wishes to communicate with the satellite. If we take the 1600km of South Africa's breadth, and add to this 2400km on either side, we end up with a band 6400km wide in which the satellite must pass for any communication to take place. Translating this distance on the earth's surface into degrees we find that $\frac{6400}{C_e} \times 360° = 57.5°$. If we see the first pass of the satellite as one passing directly over the centre of South Africa and having an offset of 0°, then communication is possible on all passes which have values of between 331.25° and 28.75° ($360 \pm \frac{57.5}{2}$), and also between 163.37° and 220.87° ($180 \pm \frac{57.5}{2} + \frac{24.23}{2}$). Note that the one band occurs when the satellite passes from North to South over South Africa, and the other when it passes from South to North.

Beginning at 0°, we find that passes in the next 24 hours occur at 24.23°, 48.46°, 72.69°, 96.92°, 121.15°, 145.38°, 169.61°, 193.84°, 218.07°, 242.3°, 266.53°, 290.76°, 314.99° and 339.22° West of the original point. Of these values 6 fall within the bands defined above, and thus during 6 of the passes communication is possible with South Africa. A conservative figure of 4 passes per day will be used for any

further calculations, as some of the passes will only allow a very short period of communication.

2.5 Link Budget

When the project was begun, the possibility of making use of an S-band link was explored. Link budget calculations show that using fixed antennae will not provide a feasible link. As the ground stations will be located in rural areas the set-up should be low-cost and low-maintenance. As a result of this tracking antennae are out of the question, and the only feasible option is to drop the carrier frequency. A UHF link was proposed, and the link budget calculation is presented here.

A link budget is used to determine how much stronger the received signal is than the noise present in the system. This gives an idea of how reliable communication across the link will be. The result is expressed as a signal-to-noise ratio (SNR), and is generally expressed in decibels (dB). A value of approximately $12dB$ and upward is considered a good SNR. The reason for this is shown at the end of Section 4.10.6 on page 53.

2.5.1 Factors Affecting Link Budget

Free-space loss

This is generally the largest contributor to loss of signal strength. It is calculated as

$$loss = \left(\frac{\lambda}{4\pi r}\right)^2 \tag{2.7}$$

where r is the distance over which transmission takes place, and λ is the wavelength in metres.

$$\lambda = \frac{c}{f} \tag{2.8}$$

where c is the constant speed of light and f is the carrier frequency.

The frequency band allocated is at approximately $400MHz$, therefore

$$\lambda = \frac{c}{400 \cdot 10^6} = 0.6892m$$

and as $r = 600km$ (for an overhead pass)

$$FreeSpaceLoss = \left(\frac{0.6892}{4\pi \cdot 600 \cdot 10^3}\right)^2 = 8.36 \cdot 10^{-15} = -140.78dB$$

For a ground station at the edge of the footprint $r = 2600km$, and thus

$$FreeSpaceLoss = \left(\frac{0.6892}{4\pi \cdot 2600 \cdot 10^3}\right)^2 = 4.45 \cdot 10^{-16} = -153.52dB$$

which is approximately $13dB$ worse than for a station in the centre of the footprint.

Troposphere Losses

The main losses caused here are by the presence of precipitation (rain or snow) in the troposphere, and these have more pronounced effects for higher frequencies. Detailed material on this subject can be found in [Maral-99, pp. 46-55], and a simpler analysis in [Hess-93]. An estimated conservative value of $0.3dB$ is used.

Attenuation Due to Other Gases

This is negligible at $400MHz$.

Antenna Efficiency

Not all of the power supplied to the antenna is transformed into radio waves, as some of it is reflected back into the amplifier. An antenna efficiency of 90% ($-0.458dB$) is assumed for each antenna, so

$$efficiency = -0.46 - 0.46 = -0.92dB$$

Antenna Gain

The antennae used are low gain antennae, so as to be able to communicate when the satellite is low above the horizon. A gain of $0dB$ is used.

Satellite Receiver (Uplink)

The following is known about the satellite receiver:

- IF bandwidth $B = 20kHz$
- Satellite antenna noise temperature $T_{ant} = 290K$

- Satellite receiver noise figure $F = 6dB$

The noise figure can be converted to a noise temperature using

$$T = (F - 1) \cdot T_o \tag{2.9}$$

where F is not in decibels, and $T_o = 290K$, so

$$T_{rec} = (10^{0.6} - 1) \times 290 = 864.5K$$

and

$$T_{tot} = T_{ant} + T_{rec} = 290 + 864.5 = 1154.5K$$

Received noise power is dependent on the bandwidth of the IF filter (as all other noise is filtered out by the receiver), and the antenna and receiver noise temperature. It is calculated as

$$N = kT_{tot}B \tag{2.10}$$

where k is Boltzmann's constant. Therefore,

$$N = 1.38 \cdot 10^{-23} \times 1154.5 \times 20 \cdot 10^3 = 3.186 \cdot 10^{-16} = -154.97dBW$$

Ground Station Receiver (Downlink)

An ICOM dual-band radio was used for communication with the SUNSAT, South Africa's first satellite. The specific model is the IC-821H VHF/UHF All-mode Transceiver. The user manual specifies its sensitivity as $0.18\mu V = -122dBm$ for $12dB$ SINAD. The receiver was measured using a Hewlett Packard RF Communications Test Set, to determine the validity of this specification. The following signal was used:

- $430MHz$ carrier
- $1kHz$ modulating signal
- $2.5kHz$ deviation

The results are shown in Table 2.1 and in Figure 2.4.

It is not possible to detect and demodulate a signal smaller than $-119dBm = 0.251\mu V$ with the IC-821H, contrary to the threshold of $0.18\mu V$ that was specified. The SNR at $-119dBm$ is $16dB$.

RF Amplitude [dBm]	RF Amplitude [μV]	Distortion [%]	SINAD [dB]	SNR [dB]
-119	0.251	18	14	16
-118	0.282	13	16	17.5
-117	0.316	11	19	18.8
-116	0.354	9.5	20	20
-115	0.398	8.5	21	21
-114	0.446	7.5	22	22
-113	0.501	7	23	22.8
-112	0.562	6.2	24	24
-111	0.630	5.3	25	25
-110	0.707	5	26	25.9
-109	0.793	4.6	26.5	27.2
-108	0.890	4	28	27.8
-107	0.999	3.6	29	29
-106	1.121	3.2	29.5	30
-105	1.257	3	30.5	30.8
-100	2.236	2	34	35.6
-95	3.976	1.5	36.4	40.2
-90	7.071	1.3	37.6	43.8
-85	12.574	1.6	36	46.3
-80	22.361	1.2	38.5	47.5

Table 2.1: IC-821H receiver measurements

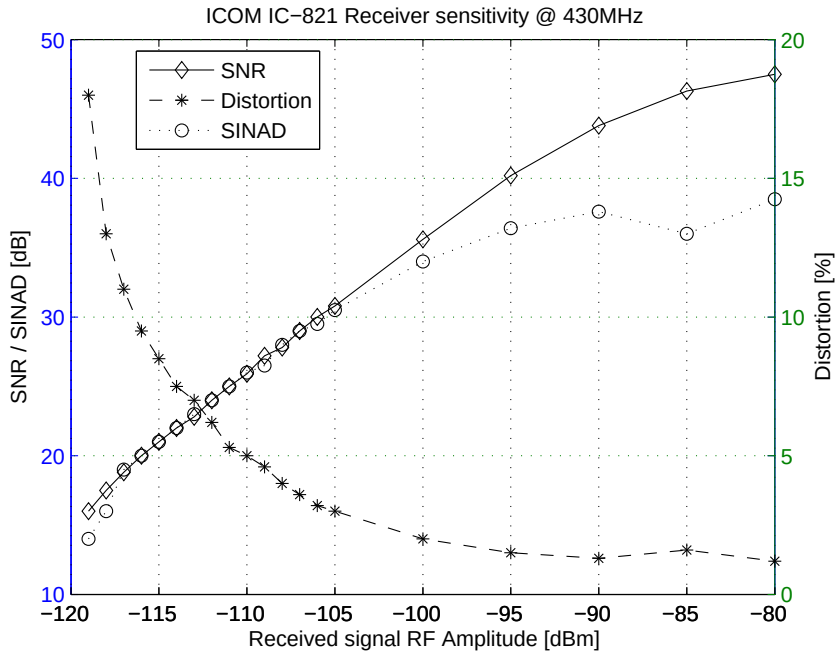


Figure 2.4: IC-821H receiver measurements

Transmitter Power

Power supplied to the TX antenna is $10W = 10dBW$ at the ground station and $5W = 6.99dBW$ on the satellite.

2.5.2 Calculations

Factor	*Value*
Transmitted Power	6.99 dBW
Free Space Loss	-140.78 dB
Troposphere Loss	-0.3 dB
Antenna efficiency	-0.92 dB
Antenna gain	0 dB
Received Signal Power	**-135.01 dBW**

Table 2.2: UHF Downlink

The power received by the ground station can thus be calculated as in Table (2.2). The received signal power of $-135.01dBW$ is equivalent to $-105.01dBm$, which is well above the $-118dBm$ threshold required by the ICOM receiver. The SNR at this point, as can be seen in Table 2.1, is $30.8dB$ which is very good.

A station on the edge of the footprint has a received signal power of $-105dBm - 13dB = -118dBm$ which is on the ICOM threshold, so communication will not be very reliable there, even though the SNR is good ($16dB$).

Factor	*Value*
Transmitted Power	10 dBW
Free Space Loss	-140.78 dB
Troposphere Loss	-0.3 dB
Antenna efficiency	-0.92 dB
Antenna gain	0 dB
Received Signal Power	**-132 dBW**
Less Receiver and Antenna Noise	-(-154.97) dBW
Signal to Noise Ratio	**22.97 dB**

Table 2.3: UHF Uplink

The SNR for the uplink can be calculated as in Table 2.3. Note that the SNR of $22.97dB$ is worse than for the downlink.

A station on the edge of the footprint has a SNR of $22.97dB - 13dB = 9.97dB$ which is less than the $12dB$ that defines a good link, as specified at the start of Section 2.5. This results in a poor BER, and means that a message from a station on the edge of the footprint is more likely to have errors and require retransmissions.

A value of $13dB$ will be used for further decisions. This is the SNR for the uplink of a station approximately $1700km$ from the centre of the footprint.

2.6 SNR to BER

For digital communication a SNR value on its own does not mean much. It is necessary to convert this to a bit error rate (BER) to know exactly what effect it will have on communications.

The chip being used for demodulation on the satellite is the CMX589A, a GMSK modem from Consumer Microcircuits Limited. The data sheet supplies a graph (see Appendix B) which gives the BER characteristic curve for a given SNR. Using the conservative case where $BT = 0.3$, and the calculated SNR value of $13dB$, it can be read from the graph that

$$BER = 10^{-4.6} = 2.51 \cdot 10^{-5}$$

2.7 Hardware Considerations

2.7.1 Rise-time and Stabilisation-time

Rise-time

When a station decides to transmit data, the data is not sent immediately. There is a small delay which results from physical constraints of the transmitter. The transmitter cannot instantaneously transition from an output of zero watts to its required output. It takes a few milliseconds for it to ramp up its power, and transmission can only begin once it is transmitting the carrier at full power.

Stabilisation-time

When the carrier ramps up to full power one will normally observe oscillations in the carrier frequency, which last for a few milliseconds before dying away. One needs to wait until the carrier has stabilised before transmitting or else the data may be

corrupted by the oscillations.

A typical hand-keyed voice radio has a combined rise- and stabilisation-time of approximately 150ms, and when these radios are used for data transfer a delay must be built in so that transmission does not begin before the carrier has stabilised at full power. Starting transmission too early generally results in lost bits. At a data rate of 9600bps, 150ms is a long time, and is equivalent to the transmission time of 1440 bits. Luckily there are data radios which have been improved upon in this regard.

The data sheet of the TAIT TM8115 voice and data radio (see Appendix C) shows a rise time of less than 10ms, and such a radio should be used for this application. There is no specified stabilisation time, so it is assumed that it is either negligible or that it included in the rise-time. This should be confirmed with measurements before making a final decision on what radio to use.

The effect of the 10ms delay can be likened to increasing the propagation time by 10ms.

2.7.2 Processing Time

Another physical factor that affects the efficiency of the system is the turnaround time at each station, or the time which it takes a station to process the received data packet and decide what the next step should be. With processors becoming faster and faster this is generally a negligible amount of time, but a conservative value of $t_{process} = 1ms$ will be assumed.

2.8 Summary

- Ground stations up to 2400km from the centre of the satellite's footprint should still be able to communicate with the satellite, although the SNR (and thus BER) worsens towards the outside of the footprint and results in more retransmissions being required for these stations
- Propagation time is $2ms$ for a station in the middle of the footprint and $8.67ms$ for a station at the edge
- The satellite will pass within communication range of South Africa approximately 4 times per day

- During each of these passes a least 12 minutes of communication time is expected
- Fixed antennae will be used
- An average BER of $10^{-4.6} = 2.51 \cdot 10^{-5}$ is expected
- A rise- and stabilisation-time delay of 10ms is expected

Chapter 3

Overview of Protocols

In order for two (or more) parties to exchange data, there needs to be a consensus amongst these parties as to how this is done (when to be silent and when to transmit data). This is especially important in the case of several parties sharing a common communications channel. A set of rules, known as the data transfer protocol, to which all parties should subscribe, is formulated to facilitate data transfer.

In order to make an informed decision as to which protocol will be best suited to our application we need to make a study of the various kinds of protocol that are already in existence and in use. This chapter examines various existing protocols, and the advantages and disadvantages of each. These advantages and disadvantages are weighed against the system constraints which were determined in Chapter 2, and the best protocol for our purposes is selected.

3.1 OSI-7 model

In an attempt to standardise protocols the International Standards Organisation specified a 7-layer model for protocols. The layers are:

1. Physical layer - The hardware used, and also the modulation and demodulation scheme.

2. Data link layer (or Medium Access Layer) - Manages data transmission and error control.

3. Network layer - Performs routing and congestion control.

4. Transport layer - Establishes and deletes connections across the network.

5. Session layer - A type of user interface for the transport layer. Exercises dialogue control and token management.

6. Presentation layer - Syntax and semantics which govern the format in which information is transmitted and presented.

7. Application layer - Interface for the end-user. Supports the end-user application process, which includes file formatting and terminal compatibility.

It is important to note that certain protocols may cover more than one level of this protocol stack. A single protocol may, for example, define how routing is done, establish and delete links and handle dialogue, and thus covers layers three to five (network, transport and session). The protocols that we will be examining are mostly of this type, covering layers three to five.

More detail about the various layers can be found in [Black-93, p. 229].

3.2 Basic Types of Communication

Communication methods can be compared on the following criteria:

3.2.1 Channel Access

Simplex: Communication in one direction only, e.g. television.

Half-duplex: Communication in both directions, but not simultaneously, e.g. simple walkie-talkies.

Full-duplex: Communication simultaneously in both directions, e.g. telephone.

For our application we have a full-duplex channel, and both the satellite and the ground stations are thus able to sense the channel or receive data packets while transmitting.

3.2.2 Timing

Synchronous transmission: A fixed frame size is used. All packets have the same long header and footer with the address and data in between. The header is used for the receiver to synchronise with the transmitter.

Asynchronous transmission: Variable packet sizes are used. The packets consist of a start bit, header, fixed or variable length of data, error control bits (such as a CRC) and one or more stop bits.

3.3 Multiple Access Protocols

We are concerned with the case where there are several users communicating via a single data channel. Protocols for this type of system are known as multiple access protocols. They can be divided up into three main groups, namely fixed assignment, centrally scheduled and contention protocols. In Sections 3.4, 3.5 and 3.6 we will examine each of these with satellite communications in mind. Note that this is not an exhaustive list. Several protocols were omitted as they are not suited to satellite communications (e.g. Token Ring).

3.4 Fixed Assignment Protocols

3.4.1 TDMA (Time Division Multiple Access)

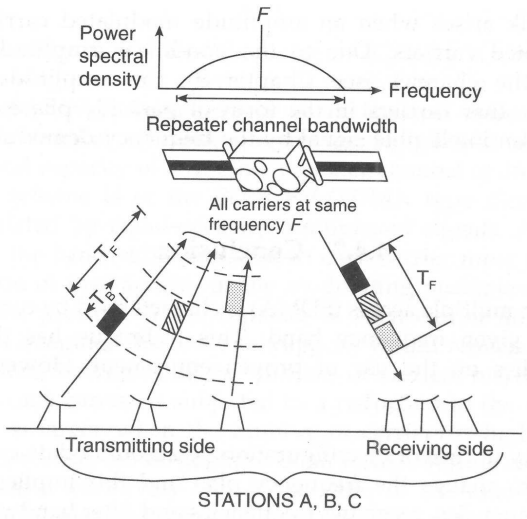


Figure 3.1: Operation of TDMA

Each ground station is allocated a fixed time slot within a larger frame in which it may transmit using the full channel bandwidth. As each of the transmissions is received the satellite places them in their slot in its message frame and relays this frame back down to the receiving stations.

Note that the ground stations need to be synchronised so as not to overlap their transmissions.

One of the advantages is that all users can simultaneously receive any broadcast that is made by the satellite.

There are two main types of TDMA:

Synchronous TDMA

Each ground station is assigned a fixed slot, whether it has data to transmit or not. Bursty ground stations (stations which have high traffic for a small percentage of the time and little or no traffic for the rest of the time) thus cause such a protocol to become inefficient.

Statistical TDMA

This protocol rests upon the fact that diversity generally exists between different ground stations and their instantaneous traffic demand. It is rare that all ground stations will need to transmit data within a specific time frame. The number of slots that are available is always less than the number of stations, and only those stations which have data to send will transmit. It will occur from time to time that more stations will have data to send than what there are slots in the frame. In this case some of the stations will have to wait until the next frame. It is thus necessary for all stations to have a large enough buffer to handle unserviced data.

This clearly causes a delay. The mean expected delay will need to be considered with the specific application to determine whether this delay is acceptable or not. Someone downloading an e-mail will not be aware of a one second delay, but someone trying to have a telephone conversation across the same link will have difficulty.

A further consideration is that all parties need to know which stations have data to send. This results in additional overhead which decreases effective information throughput.

Advantages

- All stations will receive a single broadcast
- For synchronous TDMA all stations are guaranteed a chance to transmit their data
- Statistical TDMA improves channel efficiency by decreasing wasted time

Disadvantages

- The ground stations need to be synchronised, which is seldom possible
- Statistical TDMA requires additional overhead
- Statistical TDMA can result in channel delays

3.4.2 FDMA (Frequency Division Multiple Access)

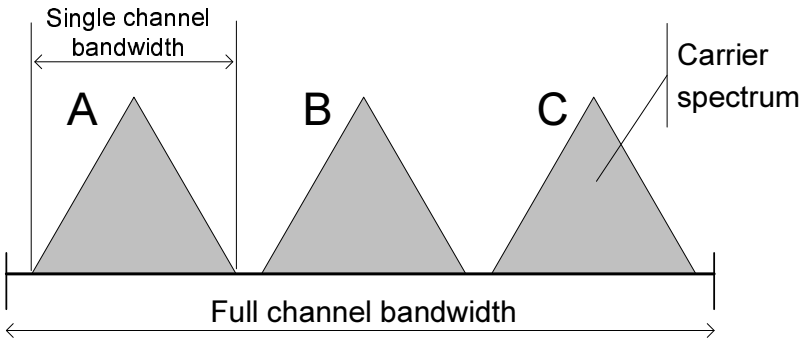


Figure 3.2: FDMA bandwidth sub-division

The total available bandwidth is divided into sub-bands (as shown in Figure 3.2) and each station is allocated a specific band. The station is free to transmit on that band all the time. There is no chance of transmissions overlapping in time, as we saw with TDMA, but the signals may overlap in the frequency domain. The sub-bands thus need to be spaced a small distance apart, and these spaces are referred to as guard bands. These lead to a loss of usable bandwidth.

Advantages

- Simple to implement (and thus widely used)
- No collisions (overlapping of transmissions)
- Efficient under high data loading

Disadvantages

- Guard bands are necessary and thus bandwidth is wasted
- Bursty stations result in inefficient use of bandwidth

3.4.3 CDMA (Code Division Multiple Access)

Stations transmit continuously and all over the same frequency. Each station first combines its data with a binary signature or *code,* which is unique to that specific station. The satellite has knowledge of these codes and is able to use this to distinguish the different transmissions from one another. The set of codes must:

- be easily distinguishable from a replica of itself shifted in time
- be easily distinguishable regardless of other codes used on the network

The combination of this code with useful data results in a signal which has a wider spectrum than the data on its own, and this is thus referred to as a *spread spectrum* technique.

There are two main types of CDMA which are used: Direct Sequence (DS-CDMA) and Frequency Hopping (FH-CDMA). See [Maral-99, pp. 172-184] for a comprehensive explanation of the two.

Advantages

- Simple to operate as no synchronisation is necessary
- Offers useful protection properties against interference
- Whole bandwidth available to all users

Disadvantages

- Larger bandwidth is required
- More difficult to implement

3.5 Centrally Scheduled Protocols

Centrally scheduled protocols generally acknowledge one of the users (in our case the satellite) as the server which arbitrates use of the communication channel.

3.5.1 Round-Robin Polling

The satellite polls each of the ground stations in turn to enquire whether it has data to transmit. A ground station with data to send waits until it is polled before sending its data.

If a large number of stations is present, the time between transmissions for a particular station can become large. Bursty users further decrease efficiency. A large propagation time, as is common with satellite communication, also results in a slow polling cycle.

This is a very simple protocol and there are many variations on it. Consult [Tobag-80] for a review of the various round-robin protocols. Here are but a few:

Adaptive Polling

Groups are polled as a whole to determine the demand for data transmission. Then stations are individually polled to determine how full the buffer is at each. Stations with greater backlog are then polled more frequently than others. Statistics are kept for each station, and the frequency of polling is adapted from time to time.

Priority Polling

Stations with a higher known rate of data input are identified ahead of time and simply polled more frequently. This is useful when the importance of service at some stations is greater than at others.

Binary Tree Polling

This follows the *divide and conquer* strategy. The entire network is broken up in a binary tree format. First one half of all the stations is polled to see if any of them have data to send. If no response is received, the other half is polled. If a response *is* received then that half is split into two groups and the process repeated. This recursion continues until individual stations are identified, and then transmission can take place.

Reservation Round Robin Polling

This is normally a slotted scheme, although it does not have to be. All stations are sequentially polled to determine which have data to transfer. This is the reservation cycles. Those that do have data are reserved a slot in the data retrieval cycle. During this time these stations are successively polled for their data. The idea is that the reservation cycle polling is very short and results in better channel use.

Advantages

- Completely stable

- No collisions
- Deterministic, resulting in easy implementation and throughput prediction
- Very simple to implement

Disadvantages

- Long propagation delays result in inefficiency
- A large number of stations results in a long polling cycle

3.6 Contention Protocols

This is an extremely important group of protocols as it finds widespread use. As a result of this, much attention has been given to the modelling and simulation of these protocols. Many variations and adaptations of the basic protocols have also been suggested and used. The most important work done in the modelling of these protocols was performed by Kleinrock, Tobagi, Lam and Molle, and their work, [Klein-75a, Klein-75b, Klein-75c, Lam-75, Tobag-80], still stands as the basis of much of what has been achieved with these protocols.

3.6.1 ALOHA

A station with data to send simply transmits its data, without checking to see if the channel is available. If it does not receive an Acknowledge (ACK) within a set time period it assumes that the data transmission was not successful, and it re-transmits the data. It should be obvious that this protocol performs well for low traffic levels, and that its throughput deteriorates rapidly under load.

ALOHA was first implemented on a computer network at the University of Hawaii.

Slotted ALOHA

The timebase is divided up into slots and data transmissions are not allowed to continue over slot boundaries.

Consider the unslotted case, where a message is transmitted at time t_0 and takes t_{msg} to transmit. If another station decides at any point between $t_0 - t_{msg}$ and $t_0 + t_{msg}$ to transmit a message, then a collision occurs. The vulnerable period is thus $2t_{msg}$ long.

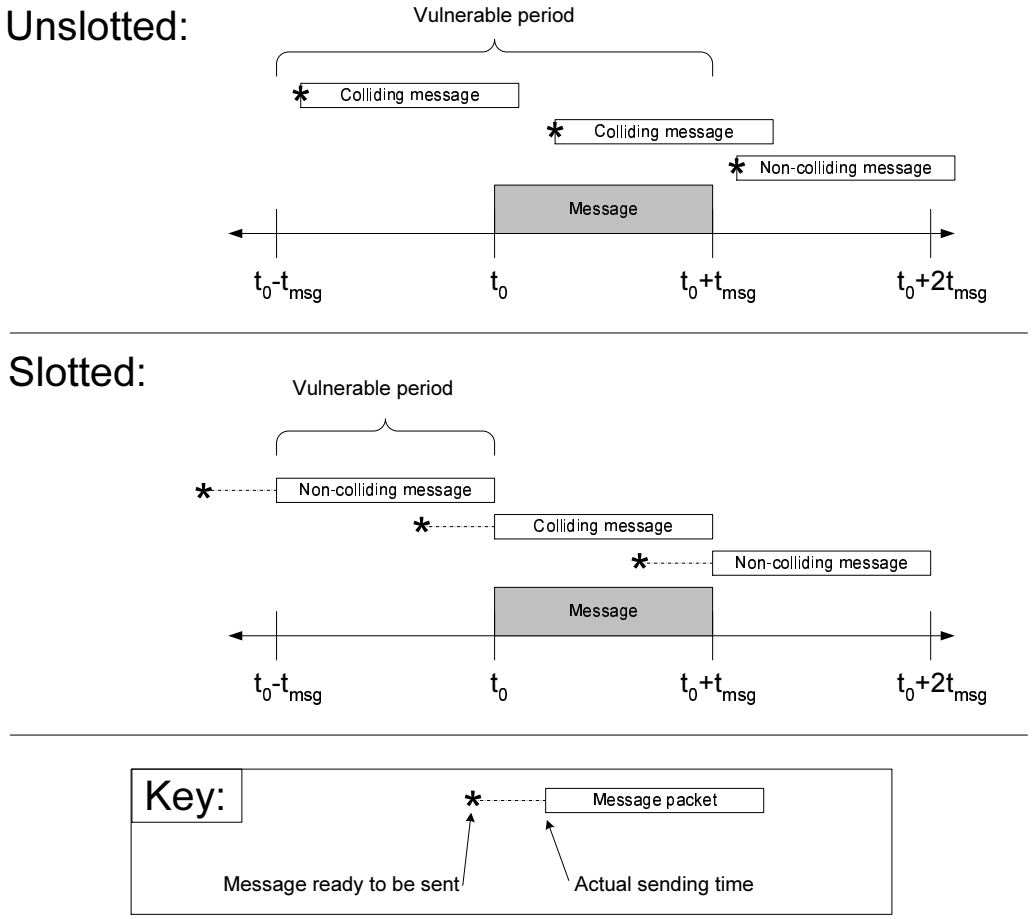


Figure 3.3: Slotted ALOHA vs Unslotted ALOHA

In the slotted case, any station deciding to send in the period $t_0 - t_{msg}$ to t_0 will wait until t_0 and then transmit, causing a collision. A station deciding to send in the period t_0 to $t_0 + t_{msg}$ waits until $t_0 + t_{msg}$ before sending, and thus no collision occurs. This halves the vulnerable period, as is shown in Figure 3.3, which increases the throughput.

Advantages

- Extremely simple and thus easy to implement
- Very efficient under low loading

Disadvantages

- Throughput deteriorates rapidly under loading
- Slotted ALOHA is more difficult to implement, as synchronisation of the

ground stations is necessary

3.6.2 CSMA (Carrier-Sense Multiple Access)

A station with data to send first senses the channel for traffic. On finding it available it will transmit its data. If it does not receive an ACK within a specified time it will repeat the process. Should the channel be sensed busy, the station will back-off for a random amount of time (as determined by a predefined random variable distribution or strategy), and then try again.

This protocol improves upon ALOHA's faults by attempting to avoid unnecessary collisions.

The protocol described above is known as non-persistent CSMA, as the station always backs off on finding the channel busy. An alternative protocol is 1-persistent CSMA. In this case, on finding the channel busy, the station continues to monitor the channel, and transmits its data as soon as it is found to be clear.

Optimisation Techniques

Many suggestions have been made for ways in which to optimise this protocol. One method, suggested by Lam and Kleinrock [Lam-75], involves dynamic adjustment of the back-off strategy. Their strategy, however, relied upon stations knowing the status of all other stations in the network, and this is seldom the case.

Another adaptation that they suggested involved rejecting input data packets under heavy loading conditions. In most cases this is impractical.

They contend that no known solution exists for the practical case where stations do not possess complete knowledge of the total network state. The only information available to stations is statistics which can be garnered from the traffic passing each individual station. They provide two empirical back-off control algorithms to minimise throughput delay and system lockup.

The first approach is based on system estimation, where each station keeps a record of channel history for a given number of transmission periods.

The second simply increases the back-off linearly with an increase in observed channel traffic.

Simulations have shown both these algorithms to be effective.

Slotted CSMA

This is basically the same as standard CSMA, except that the timebase is slotted and fixed-length packets are used. In this case a further variation can be used, known as p-persistent slotted CSMA:

If a station with data to send senses that the next slot is free, it transmits the data with a probability p. If it does not send the data then the process is repeated at the next available slot.

Advantages

- Less collision prone than ALOHA and thus has better throughput under loading
- Synchronisation is necessary for slotted CSMA

Disadvantages

- Collisions still occur
- Throughput drops significantly under heavy load conditions, as can be seen in Figure 3.4

3.6.3 CSMA/CD (Carrier-Sense Multiple Access with Collision Detection)

This is similar to 1-persistent CSMA. A full-duplex channel is required. Stations monitor the channel continuously, even during transmission. In this way they are able to detect when another station happens to transmit at the same time, causing a collision. This protocol provides improved performance over CSMA, as failed transmissions are immediately detected. It is thus not necessary to sit expecting an ACK when the packet was never received on the other side.

The assumption is made here that all stations are able to detect when other stations are transmitting. In the case where stations' antennae have a narrow beam width, stations will not necessarily be able to detect one another's transmissions. A station's physical location may also cause it to be unable to detect other stations' transmissions. Such stations are termed "hidden terminals".

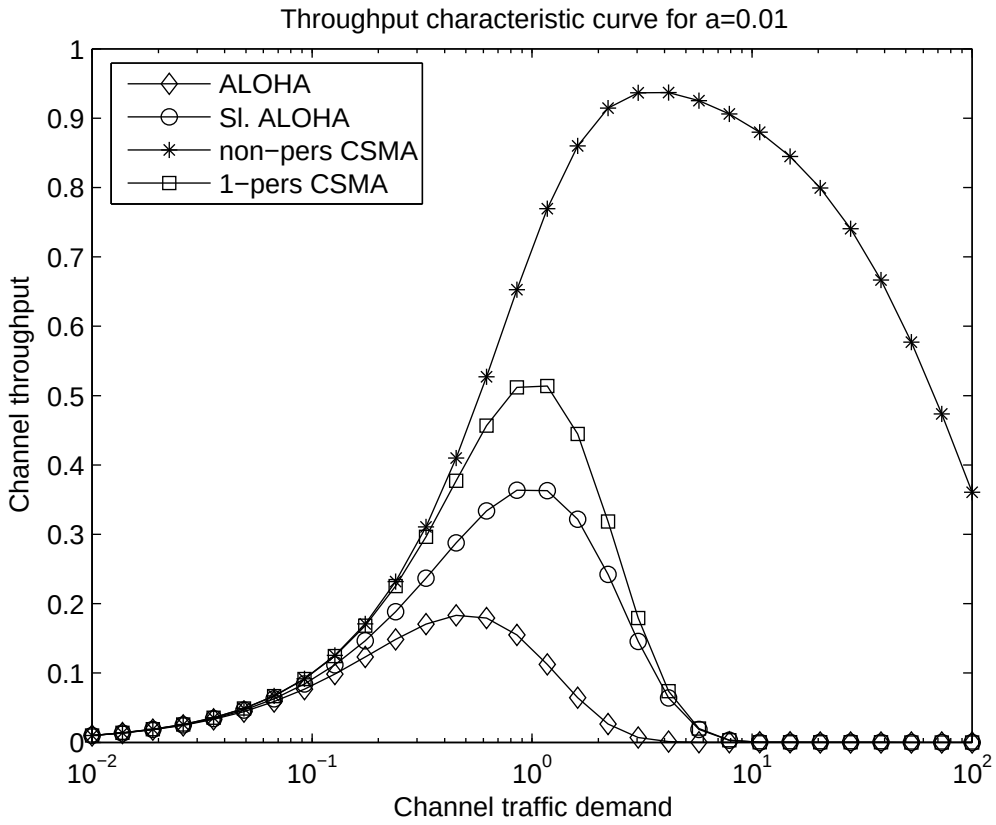


Figure 3.4: Throughput characteristic curves of ALOHA and CSMA

Advantages

- Failed transmissions are immediately detected

Disadvantages

- Full-duplex communication is required
- Hidden-terminals make collision detection impossible
- Collisions still occur

3.6.4 CSMA/CA (Carrier-Sense Multiple Access with Collision Avoidance

This protocol builds upon CSMA/CD, and takes further steps to avoid collisions. When a station has data to send it senses the channel. On finding the channel available it transmits a short Request To Send (RTS) packet, which contains the length of the data packet that it would like to send. If the server is not busy

with another data transaction it broadcasts a short Clear To Send (CTS) packet addressed to the requesting station, and this packet contains the same data as the RTS. This station then responds by transmitting its data. If it does not receive an ACK within a specified time it will repeat the process.

The other stations that receive this CTS are then aware that another station will be using the channel for as long as it takes to transmit the amount of data specified in the CTS, and they thus back-off until this time has passed. This is known as virtual carrier-sense (VCS).

The main principle is that the RTS and CTS packets are short in comparison to the message data packet. Collisions between these shorter packets is less detrimental to system throughput than collisions between message packets. The use of RTS and CTS minimises the number of message packets that collide.

Advantages

- Less message packet collisions
- VCS works for hidden terminals

Disadvantages

- Extra overhead

3.6.5 Bit-map

This is a reservation protocol. There is a contention period with as many slots as there are stations. During this period the stations compete for the channel, and if a station has data to send it inserts a 1 into its slot. When all stations have made their reservations, transmission begins in sequence as per the reservations. There are thus no collisions during the transmission period.

Advantages

- No collisions during transmission period
- Simple protocol and easy to implement

Disadvantages

- Stations must be synchronised

- Bursty users make contention time inefficient

3.6.6 Binary Countdown

This protocol depends on a fixed address scheme where stations with the highest address receive priority. Stations with data to send transmit the lowest order bit in their address. If only one station has a bit value of 1 it will transmit its data, and the other stations will allow it to do so, knowing that they have lost the bidding. If two stations have the same bit value they will then transmit the next bit in their address. This continues until a winner can be decided.

Advantages

- No collisions
- Simple protocol and easy to implement

Disadvantages

- Once addresses have been fixed the stations' priorities are fixed
- Under loading some stations may never receive a chance to send their data

3.7 Factors Influencing Selection of Protocol

There are several factors which influence our selection of a suitable protocol. Some of these factors were quantified in Chapter 2, but further discussion is appropriate.

3.7.1 Presence of Hidden Terminals

The ground station's antenna system must be simple (no tracking) and the antenna thus has a low gain. This should aid stations in sensing when other stations are transmitting. Due to the nature of the application, however, it is conceivable that stations will be large distances apart, and they may well have physical locations that prevent them from sensing other stations' transmissions.

3.7.2 Large Propagation Time

Because the speed of light is finite, satellite communication always has the unavoidable problem of propagation time. For our purposes this propagation time has

been calculated (see Section 2.3) as between 2 and 8.67 milliseconds, depending on whether the station lies in the middle of the satellite's footprint or closer to the edge. This wide range of propagation times, the largest being four times the smallest, will also have an effect on the working of the protocol we choose.

This propagation time makes it virtually impossible to synchronise ground stations via the satellite link. The fact that the stations are in remote rural locations where e-mail access is not available should rule out other methods of synchronisation. This effectively rules out all slotted protocols.

3.7.3 Rise Time of Transmitters

This is another unavoidable quantity which we have to factor in. The rise time of the proposed receiver (see Section 2.7.1) is approximately 10ms. In effect this basically becomes an increase in propagation time, as it adds to the delay between when the station decides to make a transmission and when the transmission is received by the satellite.

3.7.4 Expected Number of Users

No value has been specified for this. A proof of concept implementation will only require a few stations, but the number could conceivably reach as high as 50 under full implementation.

A large number of users will make polling schemes inefficient, and this effect is worsened by the added factors of rise time and propagation time.

3.8 Summary of Findings

- On account of the large propagation- and rise times, user synchronisation will be unpractical. This immediately rules out any form of slotted protocol.
- It is conceivable that a number of the stations may exhibit bursty behaviour, perhaps where a ground station is located in a smaller community and the demand for e-mail services is lower. This, along with the same factors of propagation- and rise time, rules out any form of polling, as this may incur unnecessary time overhead and poor channel utilisation. The presence of bursty users also rules out the fixed assignment protocols. Statistical TDMA diminishes the effect of bursty users, but it requires extra overhead and is more

complicated to implement. The synchronisation requirements of TDMA also rule it out.

- All users should have an equal chance of transmitting their data, and binary countdown is thus not suitable.
- It is important that the throughput of the system does not collapse under load, as this will only cause further backlog and mean that the system will remain under load for a longer period of time. Backlog generally results in a longer mean delivery time, and a very low throughput may result in the delivery time reaching unacceptable levels. The CSMA family of protocols improves upon ALOHA's shortfallings in this regard. CSMA/CA in particular guards well against throughput collapse through the use of RTS and CTS packets, and also the virtual carrier sense.

CSMA/CA appears to be the best protocol for this application. Careful selection of a back-off strategy may help to achieve optimal results with this protocol.

Chapter 4

Development of CSMA/CA Protocol

The basic CSMA/CA protocol is detailed in Section 4.2. The final protocol that is used is built upon this base protocol. The protocol was developed in flow-chart format, beginning with a simple diagram, as can be seen in Figures 4.1 and 4.2. It very quickly developed into a more intricate protocol, as can be seen from the final product in Appendix A.

It is not always possible to detect the more subtle errors in logical flow through examining the diagram, and one needs to rely on simulation to bring such errors to light. The protocol was thus developed and adjusted in conjunction with the coding of the simulation. As errors came to the fore they could be addressed and the protocol refined.

4.1 Considerations

In developing a protocol, one has to have the following goals and considerations in mind:

Throughput

Throughput generally refers to the amount of data that is transferred from one point to another via a communications channel over a certain period of time. One must, however, note that this data includes a certain amount of overhead (such as ACKs, message headers, CRCs etc.) which does not form part of the actual information that will be used at the destination. This overhead is there to orchestrate the safe transfer of the information.

The end user is not interested in throughput rates which include overhead as part of the throughput. This user is only interested in the rate at which usable information is transferred. From here onwards the term *throughput* will refer to the total amount of information that is transferred by the system per period of time, and is usually presented in bits per second (bps).

Efficiency

Efficiency is a measure of how well the available bandwidth is used. Efficiency is calculated as

$$eff = \frac{throughput}{channelDataRate}$$

and is generally expressed as a percentage.

Thus, if we had a throughput of 3000bps over a channel allowing data transfer at 9600bps, the efficiency would be

$$eff = \frac{3000}{9600} = 31.25\%$$

Latency

The latency of a system refers to the average period of time it takes between when a message arrives at one end of the system for delivery, and when the message is delivered and this delivery successfully acknowledged. How crucial latency is depends on the application of the system. As was mentioned in Section 3.4.1, a one second latency is not crucial for e-mail delivery, but it is unacceptable for a telephone conversation.

Power Usage

Power usage is always an extremely important factor in satellite development. A satellite is generally dependent on a battery which is charged by solar-panels, and thus needs to ensure that all systems use the minimum necessary energy. The radio transmitter is one of the larger users of energy on the satellite. If an adjustment to the protocol causes a 10% drop in throughput, but means that the satellite is transmitting for half the time that it otherwise would be, then it would be wise to sacrifice this throughput.

Noise pollution

The CSMA/CA protocol is dependent on the fact that there are no other transmissions taking place on the frequency being used for communication. Other transmissions taking place would mean that the ground stations would frequently sense the channel to be busy and refrain from transmitting. This would effectively jam the system. Even users on a nearby frequency can have negative effects on throughput.

As bandwidth is a very limited commodity we should endeavour to be responsible users of the bandwidth we have been allocated, and minimise traffic where possible.

4.2 Basic CSMA/CA

Our telecommunication network consists of a server - the satellite - and several ground stations connected by two channels - one for the uplink and one for the downlink. Only one station may communicate over a channel at a time. If more than one channel transmits then a collision occurs and the transmitted data is irreparably corrupted. The basic CSMA/CA protocol for such a system can be described as follows (see Figure 4.1 and Figure 4.2):

A station, upon receiving data to be transmitted, senses both channels for activity. If there is no activity the station will send an RTS. The server will respond with a CTS, the station will transmit it's message and receive an ACK if the transmission was successful.

If activity was sensed on the channel the station will back off for an amount of time defined by the selected back-off strategy (see Section 4.9) and then sense the channel again. It will continue to do so until the channel is sensed to be quiet and then send an RTS, and flow will continue as above.

The following sections detail refinements to the above basic CSMA/CA protocol.

4.3 Server Busy

Although the ground station may sense no activity in the channel, the server may be preparing to make or receive a transmission. In this case it needs to inform the ground station that it should remain silent. There are two possible strategies to follow here:

1. The server sends a NAK which includes information which tells the ground station how long it expects to be busy with the transmission (and subsequent

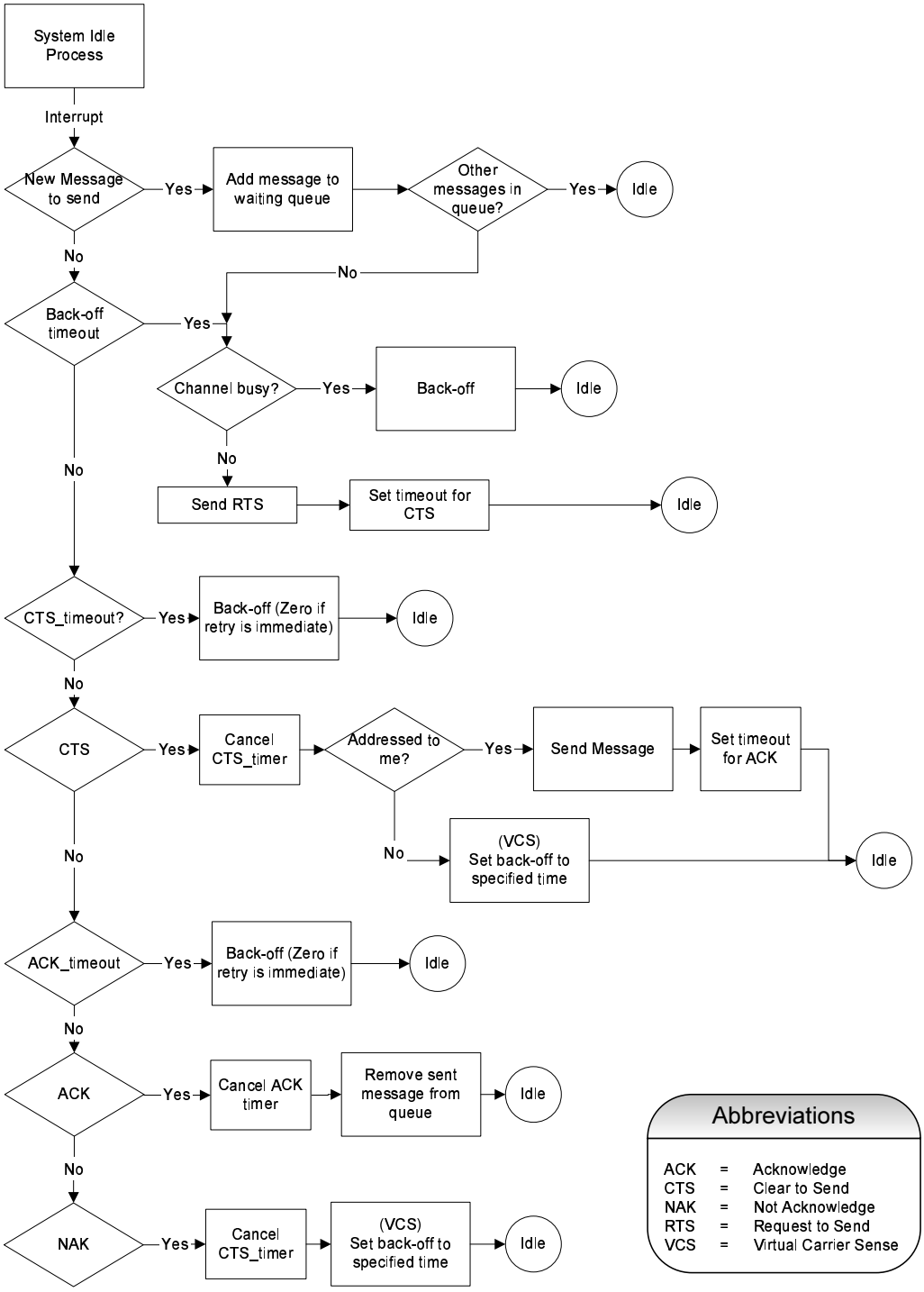


Figure 4.1: Basic CSMA/CA behaviour of the ground station

resulting transmissions e.g. ACK). The ground station sets a timer (see Section 4.5) on sending its CTS. If the timer expires then it assumes that the CTS was lost in a collision and makes a new attempt to send the CTS.

2. The server simply ignores the CTS. When the ground station's timer expires

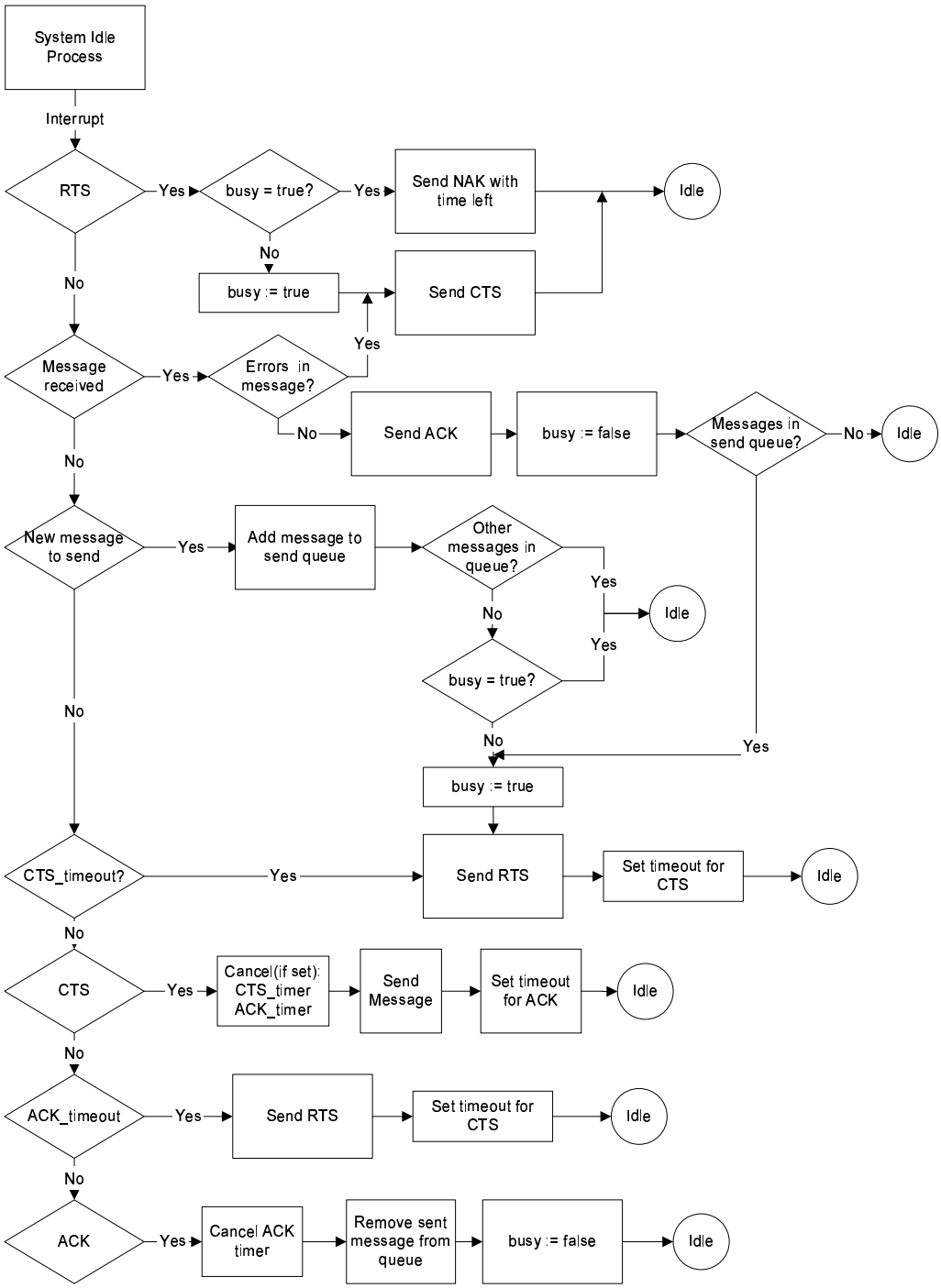


Figure 4.2: Basic CSMA/CA behaviour of the satellite

it assumes that either the server is busy or else the CTS was lost in a collision, and it goes into back-off before making a new attempt to send the CTS.

As the CTS timer in the second option needs to be sufficiently long to potentially allow propagation to and from the farthest station a lot of time will be wasted

waiting when the server is not busy and could handle an RTS, as opposed to the first option which allows the ground station to continue transmission as soon as a CTS arrives. The first option is thus employed.

4.4 Errors in Message

Because of a non-ideal transmission channel and thus a finite SNR, some of the message packets may be corrupted by errors. Sometimes, if the message has not been too badly corrupted, these errors can be corrected by using extra redundant data stored in the messages (see Chapter 5). If this is not the case then the message needs to be retransmitted.

On receiving a message which has errors the station or satellite will respond with a CTS, indicating that there was an error in the previous transmission and that a retransmission is required.

4.5 Timers and Counters

When a ground station sends, for example, an RTS has no knowledge of whether the message actually reached the satellite successfully or not. It could be that the satellite is out of range or that there was a collision of the RTS with another packet. In the basic CSMA/CA setup the station would wait indefinitely for a CTS that never comes. It is thus beneficial to make use of timers. The timers are set when, for example, an RTS is sent. If the timer expires then it is assumed that the RTS did not reach the satellite and a new RTS can be sent. If a CTS *is* received then the timer is simply cancelled.

One does not, however, want a ground station or the satellite to indefinitely continue sending unreturned RTSs. It is thus beneficial to make use of counters. Each time a specific timer expires a counter linked to this timer is incremented. When this counter reaches a previously defined limit, attempts to reach that particular station (or the satellite) are abandoned, and necessary action can be taken. If a CTS (or NAK) is received then the counter is reset, as the transmission of the RTS was successful.

The following timers and counters are used on the ground stations:

Backoff_timer Used to regulate back-off time (see Section 4.9).

CTS_timer Set when an RTS is sent.

ACK_timer Set when message is sent.

message_timer Set when an RTS is received and a CTS sent.

ReRTS_counter Increased when a CTS_timeout occurs .

ReTX_counter Increased when a message is transmitted.

Access_counter Increased when the channel is sensed busy.

The following timers and counters are used on the satellite:

CTS_timer Set when an RTS is sent.

ACK_timer Set when a message is sent.

message_timer Set when an RTS is received and a CTS sent.

ReRTS_counter Increased when a CTS_timeout occurs.

ReTX_counter Increased when a CTS is received and the ACK_timer is running. (Note that the ACK_timer being set implies that a message was sent. It was not received by the ground station so the ground station, on its timer expiring, resent the CTS).

One can calculate an expected arrival time for the ACK, CTS or message that is being awaited, and then set the timer for exactly that time, but occasionally unknown factors may slow down the processing on either side. A small amount of leeway is added to the timeout to compensate for this.

4.6 Virtual Carrier Sense (VCS)

Ground stations employ carrier sense to determine whether there are other stations transmitting and avoid transmitting until the other stations have completed their transmissions. VCS is an intelligent form of behaviour which acts in a similar fashion to carrier sense.

When the server sends a NAK or a CTS it includes in the packet a value in `dataLeft` which is an indication of how much time will still be used before the channel is open for the next transmission. All ground stations receive this packet and can thus set their back-off timers to expire only when the channel is expected to be available. This means that they avoid unnecessary collisions, and this behaviour is known as VCS.

4.7 Distance from Satellite to Ground Stations

As was discussed in Section 2.2 the distance from the satellite is not the same for all ground stations, is constantly varying, and changes with each pass of the satellite. When we set timeout lengths we have to assume that the current ground station may be the furthest one from the satellite (with a maximum possible distance of approximately 2400km) and set our timeout accordingly. This results in wasted time when messages are not delivered and the ground station happens to be close. Consolation can be found in the fact that timeouts (excluding the back-off timeout) do not occur often and are an exception to the norm. Back-off timeouts do not factor the distance in determining their back-off period and thus do not have this waste of time.

4.8 Collisions

4.8.1 Source of Collisions

Although the CA in CSMA/CA stands for Collision Avoidance, collisions do in fact occur. The two main reasons for this are deaf terminals and propagation time.

Deaf terminals are explained in Section 6.4.4. Deaf terminals transmit without realising that they are interrupting another ground stations transmission, and cause collisions.

A message travelling vertically to a satellite at a distance of 600km from the earth's surface takes 2ms (see Section 2.3). The effect of this propagation can be best understood by comparing it to a long-distance telephone call which has a delay on the line. It is difficult to hold a conversation as quite often both speakers will wait a while and then, hearing silence, start talking at the same time. It takes a fraction of a second before the speakers hear that the other person is also speaking, and usually both will stop, and have to try again. This is in effect a collision, and a similar situation can arise in the time that it takes for messages to propagate to and from the satellite. The effect is even worse for a ground station which falls on the edge of the satellite's footprint where propagation time is approximately 8ms.

4.8.2 Collision Handling

Collisions are detected by the satellite. When a collision is detected the satellite takes the following action, as shown in Appendix A:

1. The satellite sends a "`colliding`" message. Ground stations, on receiving this, go into a long back-off.

2. The satellite waits until there are no more transmissions being received, flushes the input buffer as the data is useless, and resets all counters.

3. If the satellite has messages waiting to be transmitted it attempts to send the first message in the queue by sending an RTS.

4. If there are no messages waiting then the satellite sends a CTS with the field `dataLeft` set to zero (CTS-0) and the destination address to a reserved, unused address. The ground stations set their VCS back-offs with the knowledge that there are no transmissions taking place and that the collision is complete, and little time is thus wasted before the next RTS is transmitted.

4.9 Back-off Strategies

Ultimately it is the back-off strategy which determines the throughput of the system. If the back-offs used are too short then more collisions occur and the throughput drops. If the back-offs are too long then there is much time during which no station is transmitting. This time is wasted, and consequently the throughput drops.

The back-off strategy is thus of utmost importance, and time spent on optimising the back-off pays dividends.

4.9.1 1-persistent vs Non-persistent

When a ground station has a message to send and has sensed that the channel is busy, it can either back-off and try again when the back-off timer expires, or else it can continue sensing the channel until it becomes free and then immediately send an RTS. The former strategy is referred to as non-persistent, and the latter as 1-persistent, as there is a probability of 1 that it will send the RTS.

If there is low demand on the system then a 1-persistent strategy yields a better throughput as less time is wasted. As the demand on the system increases the throughput drops to zero as the collision rate becomes extremely high. In such a case it is better to make use of a non-persistent strategy. This behaviour can be clearly seen in Figure 3.4.

It is assumed that the network in question will have a demand close to or exceeding it's capacity and this necessitates selection of the non-persistent strategy.

4.9.2 System Estimation

In the ideal situation stations have knowledge of one another's status and can adjust their back-offs accordingly, allowing stations with a high demand priority to use the channel. In most situations this is not the case, but it is possible to take some steps in this direction.

If one could make educated guesses as to the demand on the system then it would be possible to adjust each station's back-off dynamically as system traffic increases and decreases. Both ground stations and the satellite could keep statistics of messages being transmitted, and this data could be broadcast from time to time. This is an added overhead, but the frequency of the updates could be optimised so that the throughput is higher than what it would be without system estimation.

As system estimation would make the back-off strategy far more complex it was not explored further for this study. For more information see [Lam-75].

4.9.3 Back-off Lengths

Upon examining the flow diagram in Appendix A one can see that a back-off occurs at the ground station under the following circumstances:

A collision occurs and a `colliding` message is received

Stations should go into a fairly long back-off, and a value of the average message length is used. If the collision completes earlier then the satellite will send a CTS-0 and the ground stations cut this back-off short and go into a new back-off.

$$aveMessageLength = 2273$$

$$dataRate = 9600$$

$$messageTime = \frac{2273}{9600} = 236ms$$

$$backoff = 118ms + vcs\_added\_backoff$$

A CTS- or ACK timeout occurs

The ground station should immediately attempt to resend its message, so this back-off should be set to zero.

$$backoff = 0$$

The channel is sensed busy

Stations should go into a fairly long back-off, and a value of half the average message length is used. If, after the back-off is complete, the channel is still busy, an exponential back-off is used. This involves doubling the previous back-off length. This is repeated until the access counter reaches its maximum (see Section 4.5).

```
if access_ctr == 0
  backoff = 118ms
else
  backoff = lastBackoff x 2
end
```

A CTS is received which is addressed to a different station

If a CTS is received which is addressed to another station then the back-off is set according to the `dataLength` field (VCS, see Section 4.6).

$$backoff = \frac{dataLeft}{dataRate} + \frac{ACK\_length}{dataRate} + 2 \times processingTime$$

$$+ expectedPropagationTime + vcs\_added\_backoff$$

For calculation of processing time see Section 2.7.2.

As can be seen in Section 2.3 the propagation time varies from 2ms to 8.67ms. This back-off needs to assume the largest possibility, or else the station may try to transmit too early and cause a collision.

A NAK is received

This results in a VCS back-off, as when a CTS is received for another station.

$$backoff = \frac{dataLeft}{dataRate}$$

A CTS-0 is received

This CTS-0 is interpreted as an indication that a collision is completed (see Section 4.8.2) and the back-off is set accordingly (see Section 4.9.4).

$$backoff = 0 + vcs\_added\_backoff$$

An ACK is received

This indicates a successful message transfer. The same back-off is set as would be set upon receiving a CTS-0, as we want the next transmission to begin as quickly as possible.

$$backoff = 0 + vcs\_added\_backoff$$

4.9.4 VCS Added Back-off

When a CTS-0 or an ACK is received, it is an indication to all stations that the channel is now free for the next transmission. We need to have all the stations follow a back-off strategy that will be the most beneficial to all stations by giving the maximum possible resulting throughput.

A parameter `vcs_added_backoff` is defined and linked to a particular random number distribution. Each station, on receiving the CTS-0 or ACK, generates a random number from this distribution and backs off for that amount of time. This is also eventually gives all stations a fair chance to send messages in their queues.

This parameter is also added onto several of the other back-offs as a means of randomising the time at which stations timeout, to prevent stations which are of equal (or close to equal) distance from the satellite from continually sending packets to the satellite, which collide.

Much attention is given to the optimisation of this parameter in Chapter 7.

4.10 Packet Structure

4.10.1 Address

10 bits were allocated for addressing purposes. This allows for 1024 addresses which is more than sufficient.

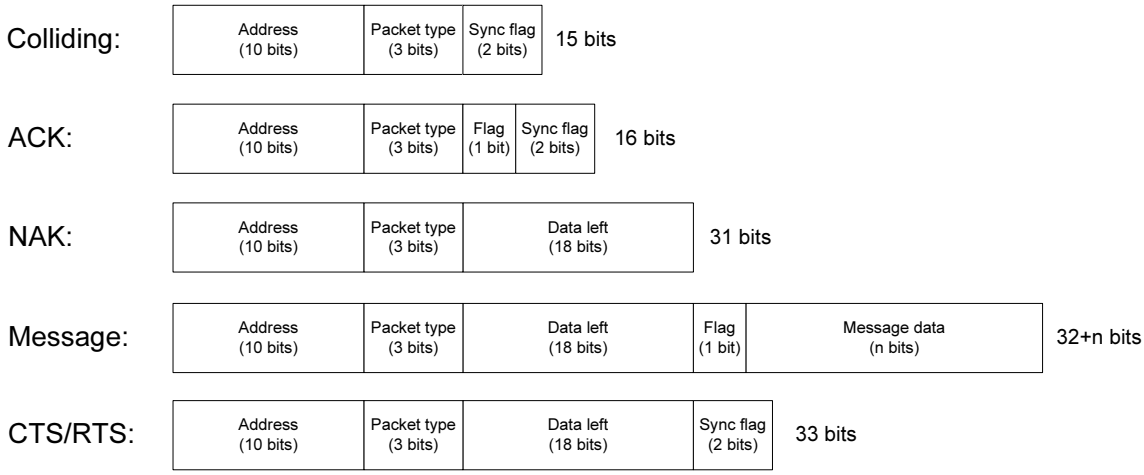


Figure 4.3: Packet structures

4.10.2 Packet Type

The following are the different types of data packets used in this CSMA/CA protocol:

ACK, NAK, RTS, CTS, colliding and message packets.

3 bits are thus necessary to identify the different types of packets.

4.10.3 Data Left (or Data Requested)

In the case of a RTS or CTS this is used to indicate the length of the message packet that is to be sent.

In the case of a NAK this is an indication of the amount of time until the current transmission is expected to be finished, which is expressed in terms of the amount of data that can be transmitted in that time.

In the case of a message packet this is the total packet length.

18 bits were allocated for indicating the amount of data left (in bits).

4.10.4 Flag

It is possible that the usual sequence of expected events can get thrown out as a result of collisions and the influence of the propagation delay. Under these circumstances it is possible that two successive ACKs can be sent for the same message. One needs to be able to determine to what message the ACK refers, so that it can be deleted from the message send queue. To this end a single flag bit is added to all messages and ACKs. This is toggled for each successive message in the send queue. If an ACK is received which does not match the message at the head of the send queue, the ACK is simply ignored.

4.10.5 Sync Flag

When a CTS-0 is transmitted all ground stations back-off and then reply with a RTS, if a CTS has not yet reached them. It is possible that the first two RTSs collide but the rest arrive safely at the satellite. As the satellite has now sent a `"colliding"` message to acknowledge the collision it should ignore these RTSs. Responding to these RTSs would be likely to cause further collisions, and also results in the unnecessary transmission of packets.

This is achieved using a two-bit sync flag, which can have values 0, 1, 2 and 3. Each time a collision occurs the flag's value is increased. When it reaches 3 it loops back to 1, so it rotates through 1, 2, 3, 1, 2, 3 etc. This flag is sent with the `"colliding"` message. Ground stations attach the last flag that they have received to all RTSs that they send.

If we now consider the above scenario, all the arriving RTSs will have the same sync flag, say 1. When the first two RTSs collide the satellite will increment the sync flag to 2 and send it with the `"colliding"` message. The subsequent arriving RTSs' sync flags are still 1, which differs from the satellite's sync flag, so the RTSs are ignored.

The value of zero is reserved for later use. It was intended that this could be used as a wild card, so that if a RTS has a value of zero it will always be accepted.

4.10.6 Optimal Message Packet Length

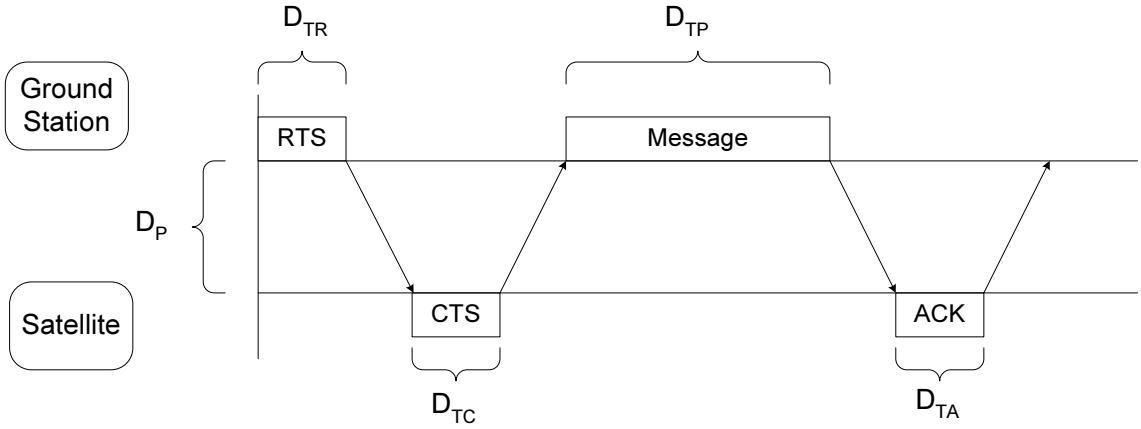


Figure 4.4: Message transmission without bit errors

Figure 4.4 demonstrates the time it takes for a message to be delivered and successfully acknowledged, when using the CSMA/CA protocol (see Chapter 4 for

details of the CSMA/CA protocol). The periods of time are referenced as denoted in the diagram. Note that

$$D_{TP} = L \cdot t_b \tag{4.1}$$

where L is the message length and t_b is the period of one bit, which can be calculated as

$$t_b = \frac{1}{dataRate}$$

The period of time involved in sending a message is thus

$$t_{tot} = D_{TR} + D_{TC} + D_{TP} + D_{TA} + 4 \cdot D_P \tag{4.2}$$

This is of course assuming that no bit error has occurred in the transmission of the message. When the receiver detects an error in the message it resends the CTS to indicate that the sender should resend the message. This is demonstrated in Figure 4.5.

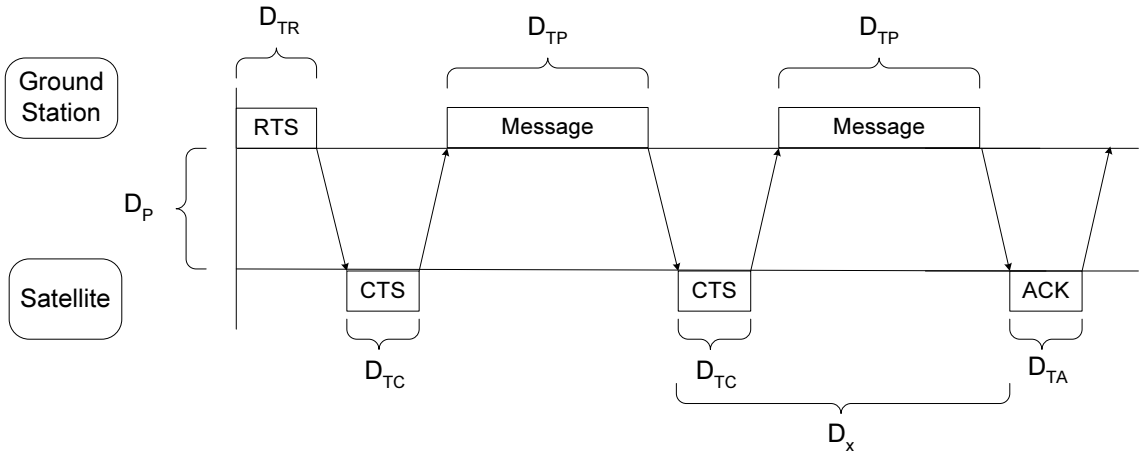


Figure 4.5: Message transmission with bit errors

In this case the time involved is

$$t_{tot} = D_{TR} + D_{TC} + D_{TP} + D_{TA} + 4 \cdot D_P + D_x$$

where

$$D_x = D_{TC} + D_{TP} + 2 \cdot D_P \tag{4.3}$$

If the probability of a bit error occurring in a particular bit is p_b (which is the BER) then the probability that no errors will occur in the message is

$$q = (1 - p_b)^L$$

The probability that an error *does* occur in the message is thus

$$p = 1 - (1 - p_b)^L \tag{4.4}$$

The probability that a transmission occurs successfully without errors is thus

$$p_0 = 1 - p \tag{4.5}$$

It follows that the probability of needing only one retransmission is

$$p_1 = p \cdot (1 - p)$$

and in the general case, the probability of needing n retransmissions is

$$p_n = p^n \cdot (1 - p) \tag{4.6}$$

Making use of this equation it is clear that the expected time $E(x)$ for a successful message transmission can be calculated as

$$E(x) = p_0 \cdot S \ + \ p_1 \cdot (D_x + S) \ + \ p_2 \cdot (2D_x + S) \ + \ p_3 \cdot (3D_x + S) \ + ... \tag{4.7}$$

where D_x is as defined in (4.3) and

$$S = D_{TR} + D_{TC} + D_{TP} + D_{TA} + 4 \cdot D_P \tag{4.8}$$

When writing (4.7) out in full

$$E(x) = (1-p)\cdot S \ + \ p\cdot(1-p)\cdot(D_x+S) \ + \ p^2\cdot(1-p)\cdot(2D_x+S) \ + \ p^3\cdot(1-p)\cdot(3D_x+S) + ...$$

one notes that it can be split up into a constant term and several series:

$$\begin{aligned} E(x) = (1-p)\cdot S \ + \ p\cdot(1-p)\cdot(D_x+S) \ + \ p^2\cdot(1-p)\cdot(D_x+S) \ + \ p^3\cdot(1-p)\cdot(D_x+S) + ... \\ + p^2 \cdot (1-p) \cdot D_x \ + \ p^3 \cdot (1-p) \cdot D_x \ + \ ... \\ + p^3 \cdot (1-p) \cdot D_x \ + ... \end{aligned}$$

Note that the first row, excluding the first term, is a geometric series with first term $a = p \cdot (1-p) \cdot (D_x + S)$ and multiplier $r = p$. The infinite sum for a geometric

series can be calculated using

$$\sum^{\infty} = \frac{a}{1-r}$$

Thus $E(x)$ can be rewritten as

$$E(x) = (1-p)\cdot S \; + \; \frac{p\cdot(1-p)\cdot(D_x+S)}{(1-p)} \; + \; \frac{p^2\cdot(1-p)\cdot D_x}{(1-p)} \; + \; \frac{p^3\cdot(1-p)\cdot D_x}{(1-p)} + ...$$

Here again it can be seen that the first two terms are constant and the rest form part of a geometric series, therefore

$$E(x) = (1-p)\cdot S + p\cdot(D_x+S) + \frac{p^2\cdot D_x}{(1-p)}$$

This can be simplified to

$$E(x) = S + D_x\cdot\frac{p}{(1-p)} \tag{4.9}$$

Substituting (4.3) and (4.8) back into (4.9) gives us

$$E(x) = (D_{TR}+D_{TC}+D_{TP}+D_{TA}+4\cdot D_P) \; + \; (D_{TC}+D_{TP}+2\cdot D_P)\cdot\frac{p}{(1-p)} \tag{4.10}$$

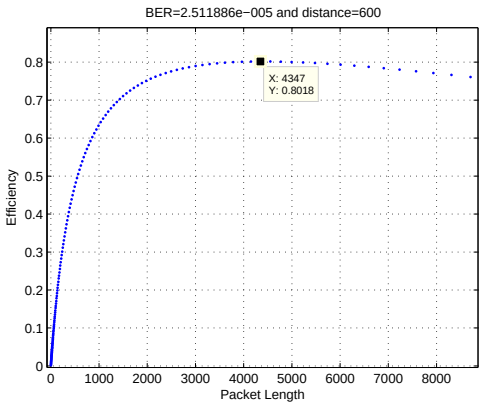
The lengths of the various packets and the message header are fixed, as shown in Section 4.10, but the length of the message packet may be varied.

The efficiency of the protocol can be defined as the time used for transmitting the message packet, divided by the total expected time it will take to successfully send the message from the sender to the receiver, or

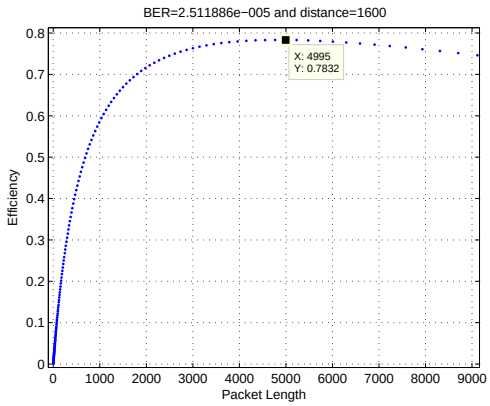
$$protocolEfficiency = \frac{D_{TP}}{E(x)} \tag{4.11}$$

Using (4.10) and (4.11) we can vary the length of the message packet to determine the optimal length. A MATLAB script, `optPacketLength.m`, produces the following results, as shown in Figure 4.6:

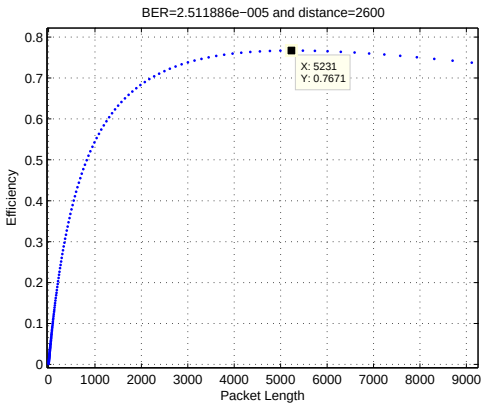
- For *distance* = 600 (overhead pass), an optimal length of 4347 bits giving an efficiency of 80.18%
- For *distance* = 1600, an optimal length of 4995 bits giving an efficiency of 78.32%



(a) d=600km



(b) d=1000km



(c) d=2600km

Figure 4.6: Optimal packet lengths when rise-time=10ms

- For *distance* = 2600, an optimal length of 5231 bits giving an efficiency of 76.71%

The middle value of 4995 bits will be used, as 1600km is likely to be close to the average distance between satellite and ground station.

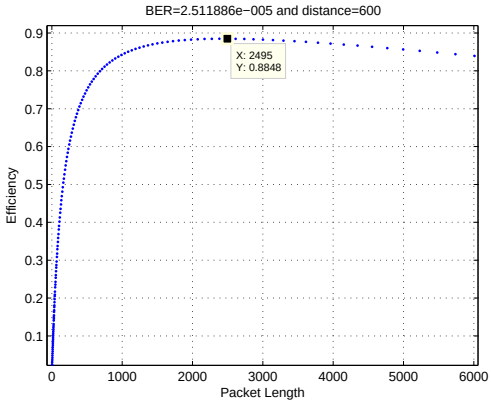
Effect of Rise-time

Figure 4.7 shows the results obtained from `optPacketLength` when it is run without including the 10ms rise-time.

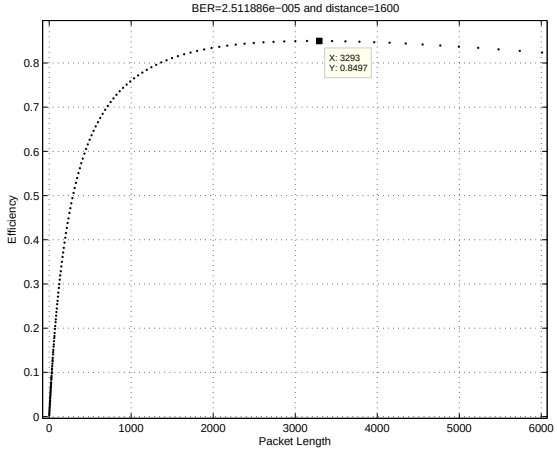
- For *distance* = 600 (overhead pass), an optimal length of 2495 bits giving an efficiency of 88.48%
- For *distance* = 1600, an optimal length of 3293 bits giving an efficiency of 84.97%

- For *distance* = 2600, an optimal length of 3963 bits giving an efficiency of 82.33%

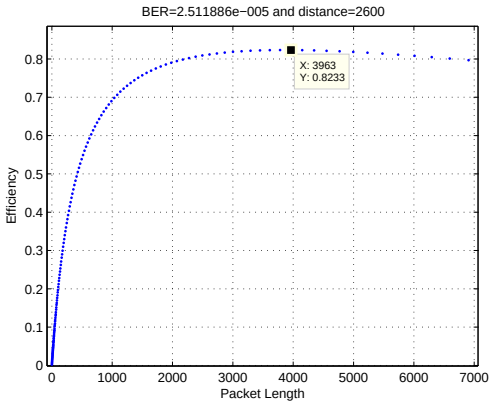
Notice that having the rise-time delay results in much longer optimal packet lengths, and also a decrease in efficiency.



(a) d=600km



(b) d=1000km



(c) d=2600km

Figure 4.7: Optimal packet lengths without rise-time

Reason Why 12dB is Cut-off SNR

It has not yet been explained why 12dB and above is considered a good SNR for a data link, and this deserves some comment. The key factor here is protocol efficiency, as defined in (4.11) on page 51. When the efficiency of the data transfer drops below 50% it means that more time is spent on overhead (RTS, CTS, etc.) than is spent on the transfer of the actual information, and this is unacceptable performance.

The process of converting SNR to BER (as in Section 2.6), and calculating the resultant optimal message length and protocol efficiency, can be repeated for a range of SNR values. The results are presented in Table 4.1.

SNR	BER	Optimal message length	Best possible efficiency
14dB	$10^{-5.1}$	8704	86.85%
13dB	$10^{-4.6}$	4995	78.32%
12dB	$10^{-3.9}$	1979	59.93%
11dB	$10^{-3.4}$	1035	43.11%
10dB	$10^{-2.8}$	430	23.35%
9dB	$10^{-2.5}$	258	15.46%

Table 4.1: Optimal efficiency for given SNR

It can be clearly seen from the results that when the SNR drops below 12dB the protocol efficiency drops below 50%, and this is why 12dB is used as the cut-off point for what is defined as a *good* SNR.

4.11 Summary

- Throughput, efficiency, latency, power usage and the combatting of unnecessary noise pollution are factors that were taken into consideration in the development of the protocol

- Packet structures are defined in Figure 4.3
- Flags are used to allow the server to ignore certain packets, with the aim of minimising noise pollution
- The optimal message packet length was calculated at approximately 5000 bits, which gives an efficiency of 78.32%

- The final protocol used is CSMA/CA, with a few refinements
- The non-persistent version of the protocol is used

- Various timers are used to prevent stations and the satellite from waiting indefinitely for a reply

- Counters are used to terminate attempts at communication where repeated attempts have been unsuccessful
- Ground stations employ VCS (virtual carrier sense) to help avoid unnecessary collisions

Chapter 5

Error detection and correction

When the CSMA/CA protocol was discussed in Chapter 4, it was noted that a ground station, upon receiving a message with errors in it, would send a CTS to request a retransmission from the satellite. What was not explained is how the receiver knew that the received message had errors in it. This chapter deals with how the errors occur, how the errors can be detected, and what options we have in dealing with them.

No specific error detection or error correction implementation is decided upon in the scope of this thesis, but as error detection is necessary and affects the throughput of the system, a brief overview is provided. For a detailed examination of error detection and correction techniques, refer to [Wolhu-03, Ch. 4-6].

5.1 Source of Errors

On the whole, noise in the system is modelled as Gaussian noise and is quantified in the SNR, and the system is designed to function within these noise levels and expected error rate. From time to time the noise distorts the transmitted signal, and a 1 is received as a 0, or vice versa. This noise would typically only affect a few bits in a message.

Burst Errors

Spurious transmissions from other devices or sources may often corrupt a whole group of bits at a time. Such transmissions may result from the weather (lightning), aeroplanes, faulty equipment, other radio users, or one of many other possible sources. Burst noise is generally difficult to model and the best that one can do

is to assume that a certain percentage of transmitted packages will be irreparably corrupted by burst noise and require retransmission.

Burst noise was not modelled or accounted for in the simulations run, and this should be taken into account when evaluating the results.

5.2 Error Detection

In order to determine whether a message has been successfully transmitted, one needs some additional information about the message. The message length is a good start, and this is often included in the message header, but this will only tell us if the desired number of bits have been received, and gives us no indication as to the validity of the received bits.

Parity

The simplest of error detection methods involves appending a single bit to the end of the message. For even parity, this last bit is set to a value such that the total number of ones in the message is even. The receiver of the message counts the number of ones in the message. On finding the sum even it assumes no errors have occurred, and on finding it odd it knows that at least one error has occurred. It is easy to see that this is not a fail-safe method, as any even number of errors (e.g. 4 bits corrupted) will not be detected by the parity check.

Note also that the extra bit is redundant data. It is not further data which has arrived, but only verification of the data which we have already received.

CRC

A Cyclic Redundancy Check (CRC) is a more complex and reliable form of parity check. It is based upon the principle of treating bit strings as polynomials, of which the coefficients are either one or zero. A *generator polynomial* $G(x)$ is defined ahead of time. Each time a message is transmitted the sender appends a series of bits to the message, with the goal of making the message bit-string divisible by $G(x)$. The receiver divides the received message by this same $G(x)$. If the answer is non-zero then an error has occurred in the message.

There are several polynomials that have become widely accepted. The typical 16-bit CRC-16 will catch:

- All single and double errors

- All errors with an odd number of bits
- All burst errors of length 16 or less
- 99.997% of all 17 bit errors
- 99.998% of all errors longer than 18 bits

Making use of checksums like those mentioned above enable one to detect most of the errors that occur, but the only option in response is to request a retransmission. This is not a problem when the propagation delay is small in comparison to the message which is transmitted, but when the propagation delay is significantly larger than the transmission time of the message packet being sent, retransmissions are avoided at all costs. A good example of this is when data is being sent to Earth from space probes far away. In this case one would like to be able to correct the errors that have occurred.

Note that the level of error detection necessary is dictated by the user application. Users receiving e-mails in which some of the characters have been replaced with other random characters will not be happy, especially if they are paying for the service!

5.3 Error Correction

The principle of forward error correction (FEC) is to include further redundant data with the message packet in order to allow a certain number of errors to be corrected, and thus remove the necessity of a retransmission each time an error occurs. There are many FEC techniques, including Hamming codes, cyclic codes such as BCH (Bose, Chaudhuri and Hocquenghem), Reed-Muller and Reed-Solomon, and convolutional codes, which are particularly useful for continuous data streams.

5.4 Coding Gain

The effect of FEC is basically to improve the SNR of the link. Increasing the power of the transmitter would have much the same effect. This improvement in SNR is referred to as *coding gain*, and is expressed in decibels. A comparison of a few FEC methods is shown in Figure 5.1, as determined in [Labus-06, p. 28].

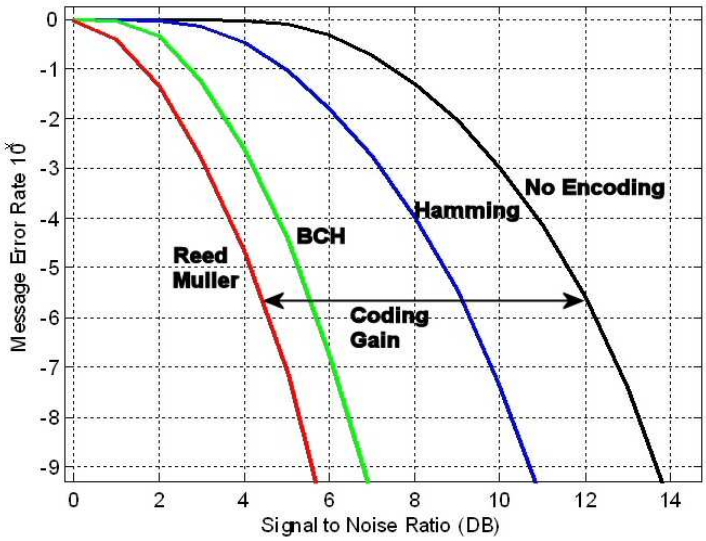


Figure 5.1: Coding gain for various FEC methods

5.5 Efficiency

The coding gain comes at a price. The SNR of the link has been improved, and less errors will occur. However, the redundant data that has to be transmitted means that the throughput drops, as the redundant data forms part of the message overhead.

Recall from Section 4.10.6 that we can calculate the optimal message packet length to use. An increase in SNR (and thus BER) results in a longer optimal packet length (as less errors occur), as well as improved efficiency. If this improved efficiency outweighs the drop in throughput, then the use of FEC increases the overall throughput of the system. For many applications this is the case.

5.6 Summary

- Errors are caused by Gaussian noise, which typically corrupts only a few bits, and burst noise, which can corrupt a whole group of bits.
- Provision was not made for burst noise, and this should be taken into account when considering the simulation results.
- Error detection can be effected by means of a parity bit or CRC checksum
- Forward error correction (FEC) employs redundant data to correct errors in received data, and minimises the need for retransmissions.

- The increased overhead of FEC must be weighed up against the coding gain it achieves in order to determine if it improves the system throughput.
- The system presented in this thesis will need to, at very least, make use of error detection techniques.

Chapter 6

Simulation

The CSMA/CA protocol that was developed in Chapter 4 is a non-deterministic protocol, i.e. it is difficult, for any given state of the system, to determine what the next state will be, as a result of the random numbers which are used in the back-off strategy and also the random fashion in which silent ground stations may initiate communication.

One thus needs to make use of simulation tools in order to determine the performance of the protocol.

6.1 Simulator Requirements

No programmed piece of code is ever without bugs, and these generally result in errors when the code is first run. In order to pinpoint the source of the problem with as little frustration as possible, there are a few criteria which the chosen simulator should fulfil:

Suitable for modelling given system. One must be able to build the desired system for simulation within the constraints of the chosen simulator. E.g. If one is modelling communication with a satellite one wants to be able to model propagation delay and noise in some way.

Reproduction of errors. Critical errors which occur as a result of bugs in the programmed code are very difficult to pinpoint if it is not possible to reproduce these errors. The simulation should always have a known initial state. It should be possible to specify this initial state. This state should include the state of the (pseudo)random number generators.

Stepping through simulation. It is very useful to be able to run the simulation up to a given point, pause the simulation and then step further, one event at a time.

Watches. When stepping through a simulation one may note unexpected behaviour. It is then useful to be able to inspect the values stored in the various variables used to code the simulation.

Statistics. As the aim of the simulation is optimisation, we need some way to measure the performance of the system. Statistics should be gathered from the simulation and output to file.

6.2 OMNeT++

The OMNeT++ Community Website [Varga-06] describes OMNeT++ as follows:

> OMNeT++ is a discrete event simulation environment. Its primary application area is the simulation of communication networks, but because of its generic and flexible architecture, is successfully used in other areas like the simulation of complex IT systems, queueing networks or hardware architectures as well.
>
> OMNeT++ provides a component architecture for models. Components (modules) are programmed in C++, then assembled into larger components and models using a high-level language (NED). Re-usability of models comes for free. OMNeT++ has extensive GUI support, and due to its modular architecture, the simulation kernel (and models) can be embedded easily into your applications.
>
> Although OMNeT++ is not a network simulator itself, it is currently gaining widespread popularity as a network simulation platform in the scientific community as well as in industrial settings, and is building up a large user community.

OMNeT++ fulfils the above stated requirements, as is shown in the following sections.

6.2.1 Suitability for Modelling CSMA/CA

OMNeT++ provides simple base classes upon which the components of practically any system can be built for modelling. The components are coded in C++ and

the functionality of each component can be fully defined. There are built-in classes which handle noisy links and allow the user to specify BER, propagation delay and data rate. There are also base classes for data packets which can be extended by the user. Each of the components of the CSMA/CA network (satellite and ground stations)can be programmed as separate modules and then linked using the provided data channels.

6.2.2 RNG Seed

A seed can be specified for the random number generator (RNG), which means that for a specific seed the exact same random numbers will be returned by the RNG each time the simulation is run. This guarantees that an error which occurs will occur in exactly the same place run after run until changes are made either to the code or to the random seed.

6.2.3 Tkenv

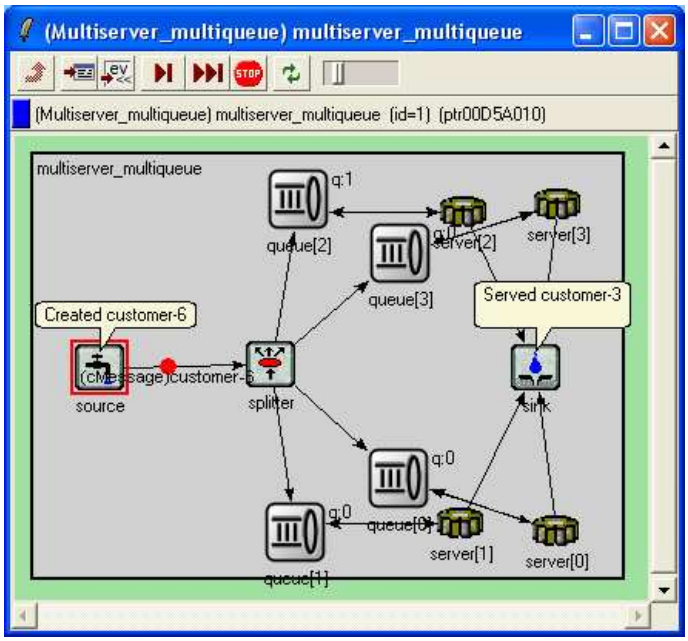
OMNeT++ provides a graphical user interface for running compiled simulations, Tkenv. Amongst other things Tkenv allows you to:

- pause and restart simulation
- set the speed at which animation of the simulation occurs
- set the simulation to run up to a specified point and then pause
- traverse the simulation tree which contains the various modules which make up the simulation
- investigate the properties and current status of the above modules, and also data packets
- view the status of selected variables (these are selected using a *Watch(x)* statement)

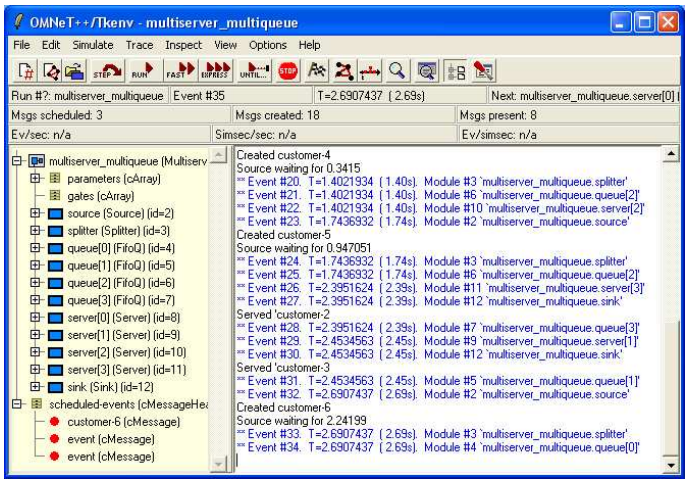
Some screenshots of OMNeT++ running simulations are shown in Figure 6.1.

6.2.4 Statistics

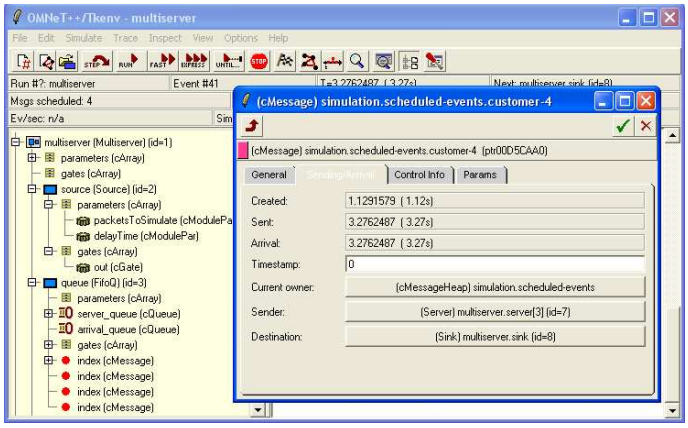
OMNeT++ provides several classes for collecting values and returning statistics of these values.



(a) Running a simulation under Tkenv



(b) Tkenv output window

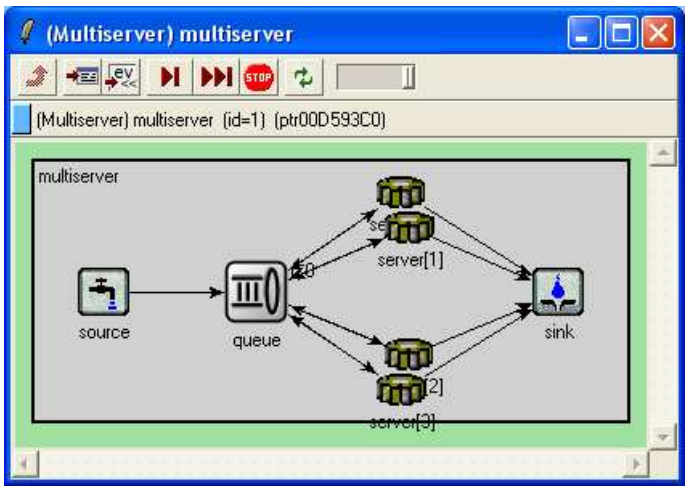


(c) Tkenv object explorer

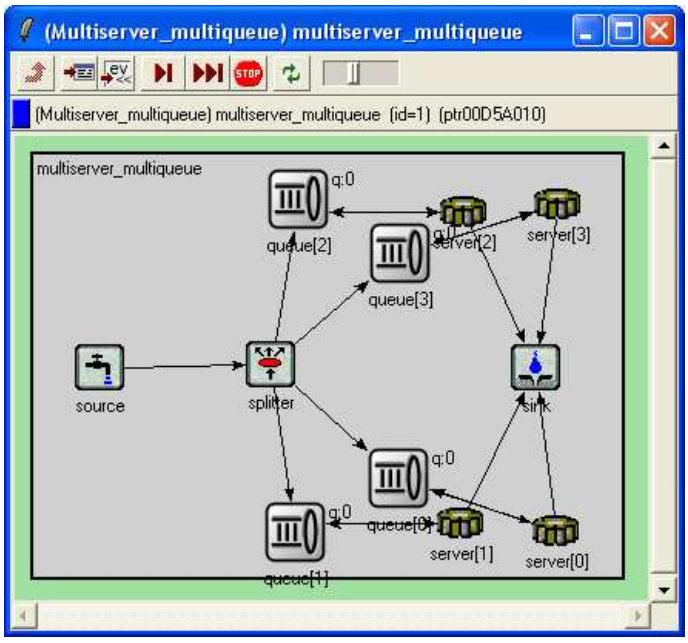
Figure 6.1: OMNeT++ screenshots

6.3 OMNeT++ Trial Run

In order to test some of the OMNeT++ functionality and ensure that the statistics objects returned the correct values, the following was simulated.



(a) Single queue and multiple servers



(b) Multiple queues and multiple servers

Figure 6.2: OMNeT++ trial runs

- a single queue with multiple servers (Figure 6.2 (a))
- multiple queues making use of the same group of servers (Figure 6.2 (b))

Using queueing theory and the formulae as stated in [Wolhu-03] we can calculate the expected mean waiting time of people in the given queues.

Single Queue with Multiple Servers

Given:

- number of servers = 4
- arrival rate = 77 per hour
- average service time = 2.5 minutes

The theoretically calculated mean waiting time is 4.38 minutes.

The OMNeT++ simulation statistics object returned a mean value of 4.40 minutes.

Multiple Queues with Multiple Servers

Given:

- number of queues = 4
- number of servers = 4
- arrival rate = 77 per hour
- average service time = 2.5 minutes

The theoretically calculated mean waiting time is 12.5 minutes.

The OMNeT++ simulation statistics object returned a mean value of 12.693 minutes.

Although only a limited test, it would appear from the above that the OMNeT++ simulation and statistics objects are accurate.

6.4 Building the Simulation

6.4.1 Configuration File

All of the parameters for the simulation are grouped together into one configuration file, `omnetpp.ini`. This file can be changed and the simulation rerun without the need for recompiling and linking the simulation classes. Amongst others, the following parameters are set here:

- Channel delay, BER and data rate
- Back-off random variables
- Number of ground stations
- Source message generation rate
- Distances of ground stations from satellite

6.4.2 Basic Building Blocks

The simulation comprises the following modules:

Server: Representing the satellite and its functionality. (The terms server and satellite are used interchangeably hereafter).

GndStation: Representing the ground stations and their functionality

Source: Used to model arrivals to the system by generating new message packets

Each of these are coded in separate C++ files, and the functionality of `Server` and `GndStation` is as shown in Appendix A.

The channels linking the server and ground stations are of type `cBasicChannel` and their parameters are specified in the `omnetpp.ini` file. The way in which the ground stations and the server are linked via the channels are defined in `csma_ca.ned` using NED - the network descriptor language of OMNeT++.

6.4.3 Packets

OMNeT++ provides the class `cMessage` which can be extended as the user desires. The simulation kernel manages the transmission and receiving of these packets.

Self-messages

Messages can be used to simulate timeouts by sending the message from a certain module directly to itself (not via any data channel). This self-message can be scheduled to arrive at a given time, and the message handler routine can recognise it as a specific self-message being used for a specific timeout.

pktMessage

`pktMessage` is an extension of the `cMessage` class and has the following added fields:

srcAddress The address of the station (or satellite) sending the message

destAddress The address of the station for which the message is destined

dataLeft The amount of data that is still to be sent

serial A serial number used for identifying packets

flag A flag which is used to match ACKs to their corresponding messages

The `cMessage` class has a field `length`. This is used by the `cBasicChannel` to determine the time that transmission takes, and also what the likelihood is that a bit error occurs.

6.4.4 Channel Sense

The CSMA/CA protocol requires that a ground station first listen to the channel to hear if there is a transmission in progress before attempting to send a RTS. OMNeT++ does not provide any direct way to sense whether the channel is busy or not - one has to check whether any of the output gates are busy or not.

Each of the ground station modules are linked to the satellite module via an extra channel having a propagation delay of 0. This channel is used for sending channel-sense messages. The process is as follows:

1. A ground station sends a channel sense message to the server via the zero-propagation-time channel.
2. The server, upon receiving this message tests to see if any of it's output gates are busy transmitting.
3. If there are gates busy a `sensed_busy` message is sent back to the ground station via the same channel, else a `sensed_clear` message is sent.

It is very important to note that for the purposes of this simulation it is assumed that all ground stations are of the *deaf terminal* type. A *deaf terminal* is a ground station which can not hear when other ground stations are transmitting, either as a result of its physical location, or as a result of the gain pattern of the receive antenna.

This assumption results in a conservative value for the average data throughput in the system, as deaf terminals are more likely to interrupt other transmissions. Interruptions come in the form of collisions between packets arriving at the satellite and this results in the need to resend packets, resulting in a lower average data throughput.

6.4.5 Collisions

In practice collisions are handled in the following way:

1. The server recognises that there is more than one station transmitting and sends a `"colliding"` message.
2. The server waits until there are no more transmissions being received, flushes the input buffer (as the data is useless) and then takes action according to the protocol detailed in Appendix A.
3. The ground stations, upon receiving a `"colliding"` message, will stop transmissions, flush their output buffers and take action according to protocol (usually some form of back-off).

The following problems present themselves in the coding of a simulation:

- Messages are handled one at a time, as they arrive, and no indication is given of collisions
- Messages are received intact, and there is no input buffer as such
- One cannot interrupt transmission and just allow the first part of a transmission to propagate to the satellite. Either the message is cancelled and nothing reaches the satellite or the whole message must be allowed to reach the satellite. This again results in a conservative result, as the theoretical throughput will be lower than in practice.

The `handleMessage` interrupt is called when the tail of the message arrives. Messages which have been transmitted but have not yet fully arrived are stored in the Future Events Set (FES). One can examine the next message in the FES to determine at what time the head of this message arrives, and can then determine whether this next message overlaps (and thus collides) with the current message which is being handled.

6.4.6 Propagation Times

Kleinrock and Tobagi [Klein-75a] simplify their modelling by assuming that all the stations lie an equal distance from the satellite. If one selects the distance as equal to the furthest possible distance then one achieves a conservative result.

In this case, however, it is deemed necessary to run simulations with varied propagation times, as this has an effect on the workings of the protocol. Some of the refinements to the protocol were brought about to combat anomalies resulting from this variation of propagation time across stations.

Note that for simulations of 15 or less stations the distances to the stations were hard-coded. Firstly, it is impossible to justify the selection of any random distribution for these distances as being better than arbitrarily selected values. Secondly, this makes it easier to determine the effect of changes of other parameters on the system. If the distances were changed from run to run then several simulations would have to be run and the average results taken, for each single change to other parameters. This would not be time efficient.

6.5 Testing the Simulation

As a small error in the coding of a simulation can give rise to large errors in the results generated from such a simulation it is of utmost importance that sufficient attention is given to the testing of the simulation code.

The simulation code was tested in the following ways:

6.5.1 All-branch Testing

In most code the program flow will follow a fairly fixed path of execution 95% of the time, and only on occasion make use of other lines of code. Errors in code within the execution path that is regularly used quickly come to light, but one can go through several runs of a simulation before discovering errors in the less-executed code. It is thus important to ensure that all possible branches of code are executed.

A simple counter is installed in each logical branch in the code, and the counter is incremented each time that branch of code is executed. At the end of each simulation run the values of these counters are output, and this gives and indication as to which sections of code need to be more vigorously tested.

6.5.2 Stress Testing

Some coding errors do not always result in program errors and are thus not always picked up. By ensuring that one piece of code is repetitively called we can ensure that it is working properly. This is especially true when complex pointer manipulation is done. Messy pointer programming will often not cause trouble until that particular piece of code is placed under stress.

One application of this technique to the simulation presented here is to run simulations with a low SNR (or high BER). This results in more messages having bit errors, which in turn means that more messages that need to be resent, resulting in a heavier traffic load.

6.6 Statistics

The main aim of any simulation is to gather statistics about the performance of the system, and this one is no different. These statistics are useful in determining the expected behaviour of such a system when put into practice. The following statistics are of particular importance to this system:

Throughput: The amount of usable data that is transmitted through the system.

Waiting time: Latency - the time from arrival of a message in the system until it is delivered and an ACK is received.

Efficiency: Indicates what percentage of the available bandwidth is being used.

Of lesser importance are:

Wasted time: Time during which no transmissions are taking place, when all ground stations are in back-off.

Collision rate: How often packets collide.

6.7 MATLAB Scripts

The simulation parameters are set in the `omnetpp.ini` file. If one wants to run the simulation for several different sets of parameters, one must each time adjust the omnetpp.ini file, run the simulation, and then access the output files for the simulation results. This is extremely labour intensive, open to human error, and

inefficient, especially when noting that the manual work takes several times longer than the simulation runs! It thus makes sense to make use of a MATLAB script which automatically adjusts the `omnetpp.ini` settings file, sets up and executes the simulation runs, extracts the resulting data from the output file, and presents it in a usable form, such as a graph.

OMNeT++ allows the `omnetpp.ini` settings file to be broken up into several parts. `omnetpp.ini` then simply imports these other files. It is thus broken up into the following files:

`omnetpp.ini` contains parameters applicable to the simulation which remain constant for all runs

`cmdenvParams.ini` contains parameters for the setup of the command line simulator environment

`general.ini` contains parameters normally falling under the [General] section in `omnetpp.ini`

`generalParams.ini` contains parameters common to all the simulation runs

`runParams.ini` contains the specific settings for each of the simulation runs

`stationDistances.ini` contains the distance parameters for simulations with more than 15 stations

The MATLAB scripts used are the following:

omnetSim.m for running simulations with a normal distribution back-off variable, and a range of mean and standard deviation values

omnetSim2_varyN.m for running simulations with a normal distribution back-off variable, and for various amounts of ground stations

omnetSim3_optimalDistr.m for running simulations with a range of different back-off variable distributions

6.8 Summary

- OMNeT++ was selected for simulation of the protocol
- A trial run of OMNeT++ gave values concurring with those given by theoretical calculations

- The simulation is built up of several modules, including those for ground stations, satellite and message source
- All stations are assumed to be deaf terminals
- Hard-coded distance values are used for simulations of 15 stations or less
- Statistics pertaining to the performance of the system are collected
- MATLAB scripts are used to automate the running of simulations and changing of parameters between runs

Chapter 7

Optimisation

One of the main aims of this thesis is to optimise the parameters of the communications system in order to maximise throughput. Once we have optimised the parameters we can run them through the simulation that we have built and verify the optimisation with our results. The most important of these parameters is one used in several of the back-offs: `vcs_added_backoff`. The reason for this is explained in Section 7.1. This chapter is dedicated to determining the ideal random number distribution for `vcs_added_backoff`.

An added benefit of running optimisations is that we can get a good idea of how the system should respond to changes in the various parameters without the need for running full simulations. Having an insight into this behaviour allows us to make more efficient use of our time spent running full simulations.

7.1 vcs_added_backoff

The main aim of `vcs_added_backoff`, as is mentioned in Section 4.9.4, is to randomise the time at which stations attempt to send their CTS.

Consider first, for simplicity's sake, the case in which all ground stations are equally distant from the satellite. After a collision has completed (or a message successfully received at the satellite) the satellite will transmit a CTS-0 (or ACK). All the ground stations receive this packet at the same time and know that the previous transmission has been completed. They all sense the channel and, finding it clear, each transmit an RTS to the satellite at the same time. These all collide. The satellite transmits a `"colliding"` message followed by a CTS-0 when the collision is completed, and the cycle begins again. No data would ever be successfully transmitted. This is known as a deadlock.

To combat this we would like only one of the stations to send an RTS and all the rest to back-off. It would be easy if all the stations had knowledge of the status of the other stations, but as it is they can only guess. The solution is to assign each station a random back-off time, `vcs_added_backoff`, and choose the distribution for the random number well, so that most of the time an RTS is transmitted and arrives at the satellite without colliding.

Consider now the case where stations are all different distances from the satellite. It is still possible here to end up with two stations being a similar distance from the satellite so that their RTSs would usually collide. It is thus still a good idea to make use of the random back-off time to avoid such collisions.

7.2 Period to be Minimised

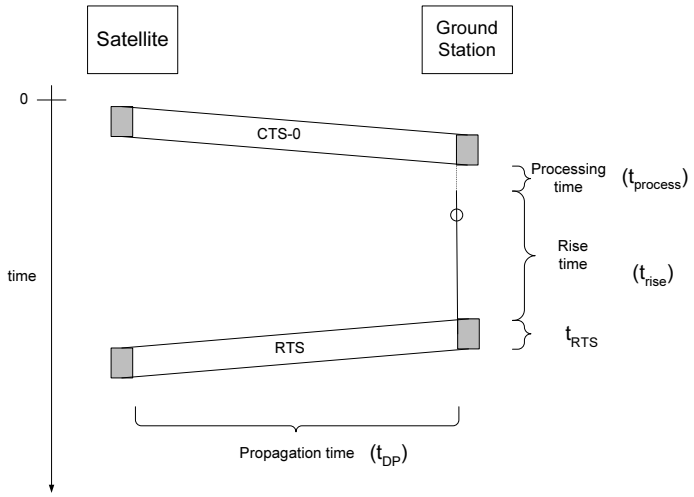


Figure 7.1: Period to be minimised

The period of time that we want to minimise for our optimisation is the time between when the satellite successfully receives a message and when the first non-colliding RTS reaches the satellite. This period is made up of the following parts, as can be seen in Figure 7.1:

$$t_{total} = t_{DP} + t_{process} + t_{rise} + t_{RTS} + t_{DP}$$

where

t_{RTS} is the time it takes to transmit an RTS,

t_{DP} is the propagation time from ground station to satellite,

$t_{process}$ is the time it takes to process a received packet,
t_{rise} is the rise time of the transmitters.

Note the following:

- that for this optimisation processing time ($t_{process}$) was considered negligible
- that rise time (t_{rise}) simply results in a delayed transmission and thus delayed arrival of the packet, and can therefore be simply modelled as an increase in propagation time (t_{DP})

The above value of t_{total} assumes that the RTS arrives without colliding. If the first RTS were to collide then the satellite would send a `"colliding"` packet. When the collision completed the satellite would send a CTS-0, and on receiving this packet the ground stations would set their back-offs to `vcs_added_backoff`, and the process would begin again. In this case the value of t_{total} is:

$$t'_{total} = t_{DP}+t_{process}+t_{rise}+t_{RTS}+t_{DP}+t_{collision}+t_{CTS} \ + \ t_{DP}+t_{process}+t_{rise}+t_{RTS}+t_{DP}$$

$$= t_{unsuccessfulAttempt} \ + \ t_{DP} + t_{process} + t_{rise} + t_{RTS} + t_{DP}$$

If the second attempt were also to fail because of a collision between the first two arriving RTSs, then t_{total} would be:

$$t''_{total} = 2 \times t_{unsuccessfulAttempt} \ + \ t_{DP} + t_{process} + t_{rise} + t_{RTS} + t_{DP}$$

The process for calculating the time it takes for an RTS to be received can thus be repeated until a successful RTS is observed to have arrived, and the total time is increased with each repetition.

Apart from two colliding RTSs, there is also another type of collision that can occur. Consider the following situation, as shown in Figure 7.2:

There is a station 600km from the satellite and another 2400km from the satellite. It thus takes an extra 6ms (as $\frac{(2400-600)\times 1000}{c} = 6ms$) for any message to propagate to the farthest station, after the closer station has received it.

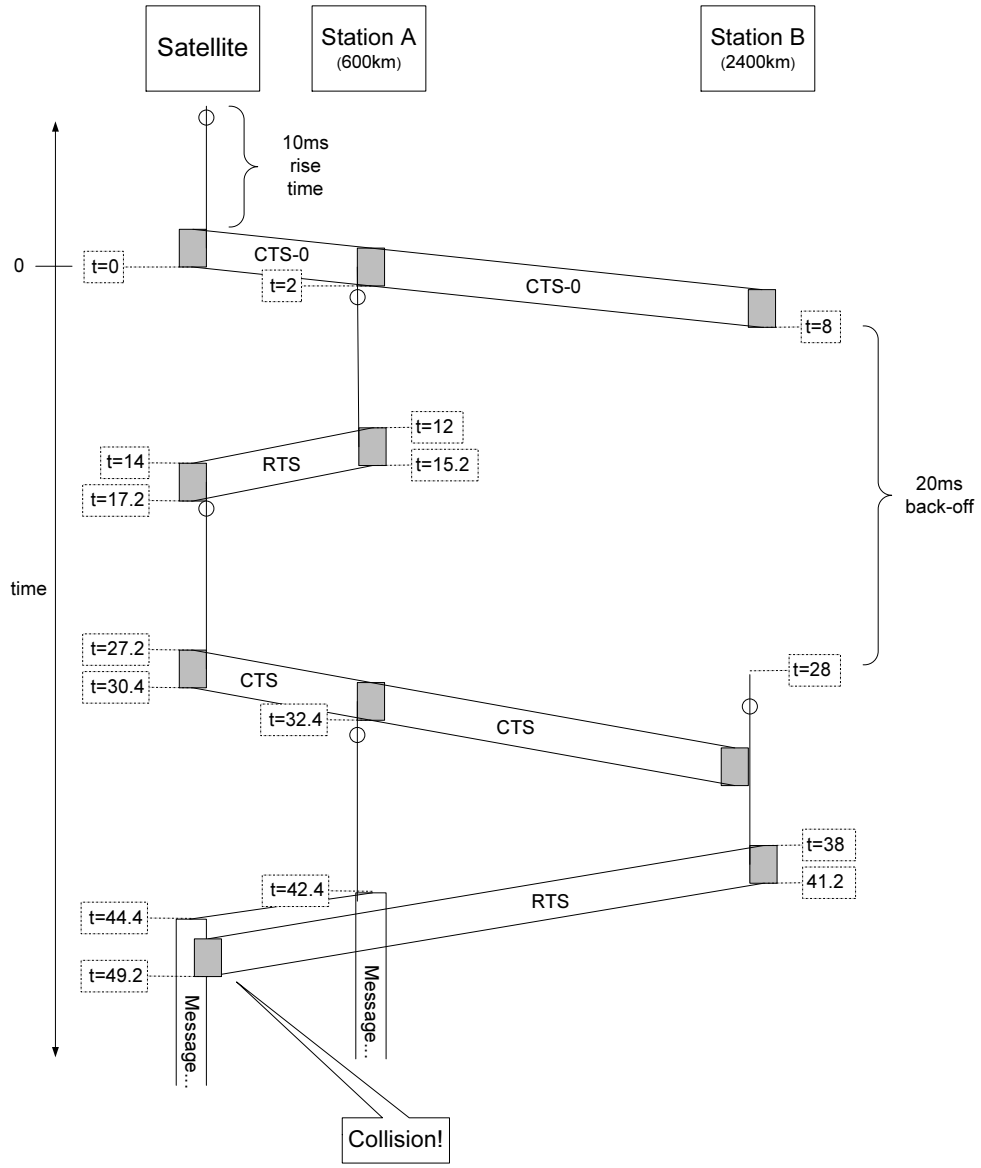


Figure 7.2: Message collision

At time $t = -13.2ms$ the satellite sends an CTS-0 which is received at the closer station at $t = 2ms$ (10ms rise time, 3.2ms transmission time and 2ms propagation) and at the farthest at $t = 8ms$ (processing time is ignored for this illustration).

The closest station happens to choose a `vcs_added_backoff` value of zero. An RTS also takes 3.2ms to transmit, so the closer station's RTS is received at the satellite from $t = 14ms$ to $t = 17.2ms$. The furthest station happens to choose a `vcs_added_backoff` value of 20ms and so it times out at $t = 28ms$ and its RTS arrives at the satellite from $t = 46ms$ to $t = 49.2ms$.

The satellite sends a CTS (another 3.2ms) to the closest station which arrives at $t = 32.4ms$ (and arrives at the furthest station at $t = 38.4ms$).

The closer station responds by transmitting its message of 2400 bits, which arrives at the satellite from $t = 44.4ms$ to $t = 294.4ms$ (as$\frac{2400bits}{9600bps} = 0.25s$).

One now notes that we calculated the farthest station's RTS to start arriving at $t = 46ms$, which is after when the closest station's message starts arriving ($t = 44.4ms$), and the two thus collide.

This collision wastes more time than when two RTSs collide, as the satellite must wait until the end of the message arrives (at $t = 294.4ms$) before it can send a CTS-0 and re-attempts can be made.

It is important to note that the second station's back-off need not be 20ms. Any value between 15.2ms and 38.4ms would result in this collision. (When the CTS arrives at the furthest station at $t = 38.4ms$ it forces the station into VCS back-off and prevents the colliding RTS from being transmitted). Also note that in this case the CTS arrived before the RTS was physically transmitted (during rise time), but that the RTS had already been sent to the transmitting device.

As the value for t_{DP} varies from ground station to ground station it is rather difficult to calculate theoretically the average time that it will take to receive a successful RTS at the satellite. It becomes especially difficult as we increase the number of ground stations. It is thus easier to make use of a computer program to run a mini-simulation in order to calculate this value.

7.3 MATLAB Optimisation

7.3.1 Why MATLAB?

Consideration was given to the possibility of running the optimising code within the OMNeT++ environment, as opposed to MATLAB. It was decided against this course of action, for the following reasons:

- MATLAB is extremely flexible and adept at manipulating data, and has a wide range of graphing tools and functions. OMNeT++ is rather limited in terms of graphing capabilities, and has very few built-in functions for data manipulation.
- Each change to the C++ code which makes up the OMNeT++ simulation results in one having to recompile the code before running the simulation again. Changes to MATLAB code require only that you save the file before running it with the changes.

- The OMNeT++ platform will drive much of the computing done in any simulation. A MATLAB script is completely transparent to the programmer of the code, and the validity of results are thus solely dependent on this programmer.

7.3.2 Breakdown of MATLAB Simulation

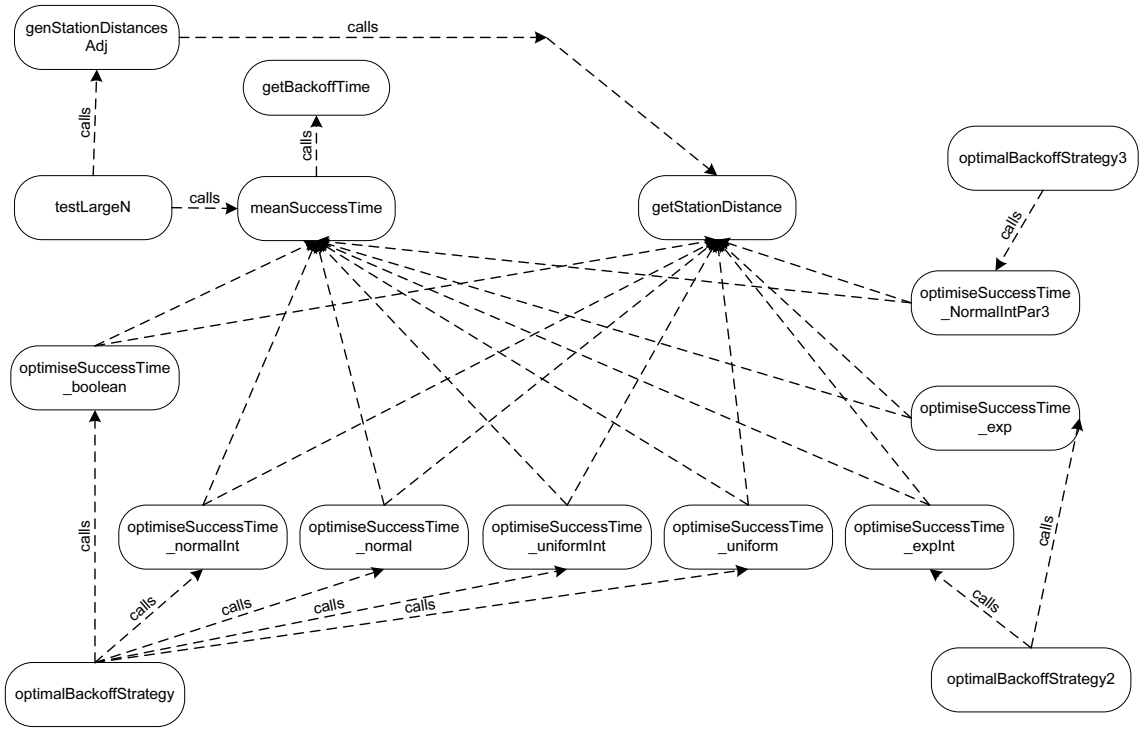


Figure 7.3: MATLAB script hierarchy

The MATLAB simulation used for optimisation comprises several MATLAB scripts, many of which make calls to other scripts. Figure 7.3 shows how the simulation is built up and which scripts make calls to which. Here follows a description of some of the scripts:

getBackoffTime generates a random back-off time from the specified random number distribution.

meanSuccessTime calculates the average time it will take before an RTS is successfully received at the satellite, without colliding. The workings of this script are discussed in detail in Section 7.3.3.

getStationDistance returns the hard-coded values for the distance (in kilometres) between the satellite and 20 ground stations.

genStationDistancesAdj returns the distance between the satellite and the specified number ground stations. Makes use of the first 15 hard-coded values returned by `getStationDistance`, and then generates random distances for further stations. The workings of this script are discussed in Section 7.3.4.

optimiseSuccessTime_X (where X is normal, normalInt, uniform, uniformInt, boolean, exp, expInt or NormalIntPar3) makes repeated calls to `meanSuccessTime` using a specified range of parameters in order to determine which set of parameters gives the lowest mean success time.

7.3.3 meanSuccessTime

`meanSuccessTime` accepts the following parameters:

distrType, par1, par2: These indicate the type of random distribution to be used in calculating vcs_added_backoff, and the necessary parameters, e.g. Mean and standard deviation for normal distribution.

distances: An array containing the distances between the satellite and each of the ground stations.

accuracy, accuracyReps: These specify to what accuracy the result is determined. Every tenth time that an RTS is successfully received the average success time is compared with what it was previously. If the percentage change is less than what has been specified by `accuracy,` then a counter is increased. If the change is larger, then the counter is reset to zero. The simulation result is considered accurate enough when the value of the counter reaches the value of `accuracyReps`. It is helpful to be able to specify a low accuracy for initial simulation runs in order to determine in which region optimal results lie, and then to specify a high accuracy (small value) for the final simulation run to generate accurate results.

par3 (optional): Specify the value of par3 (see Section 7.4 for details). This defaults to $\frac{lengthRTS}{9600}$ if no value is supplied.

The algorithm which `meanSuccessTime` follows is described in the pseudo-code below. Note that there are 3 possible situations that can arise, as explained in Section 7.2. They are:

1. The first RTS arrives at the server without colliding, a CTS is sent an the message packet successfully transferred.

2. The first and second RTS collide.

3. The first RTS arrives at the satellite without colliding, a CTS is sent and the message packet transmitted, but the message collides with an RTS from a station situated further from the satellite.

pseudo-code for `meanSuccessTime`:

```
while not Done
    generate random back-off for all stations
    RTSarrival = 2 × propTime + backoff + t_RTS
    sort RTS arrival times

    if sorted(2)-sorted(1)  < par3     {collision occurred}
        get time of last colliding packets
        add this time to accumulated total time
    else if RTS is sent before CTS arrives and
        RTS tail arrives after message has begun arriving
            get time message packet finishes arriving
            add this time to accumulated total time
    else {successful transfer}
        add RTS arrival time to accumulated time
        store accumulated time in array of successful values
        set accumulated time = 0
        if the number of successes is a multiple of 10
            compare the mean value with the previous mean
            if percentage difference < accuracy
                Done = true
            end
        end
    end
end {while}
```

7.3.4 genStationDistancesAdj

This MATLAB script generates random distances for a given number of ground stations. The algorithm upon which the code is based is shown below.

pseudo-code for genStationDistancesAdj

```
repeat
     repeat
        generate  a random (x,y) pair within a 4800x4800km square
     until within circle with radius of 2400km centred in square
     store (x,y) pair
until N distances generated
use these distances on the earth's surface to calculate distances to the sat
```

The output of a sample run to generate 80 station distances is shown in Figure 7.4.

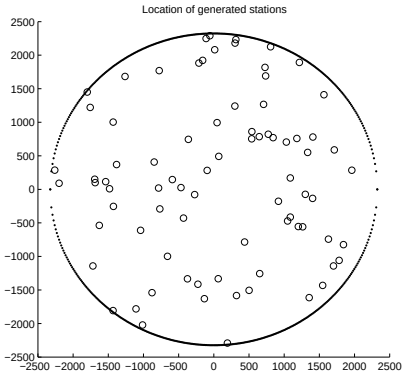
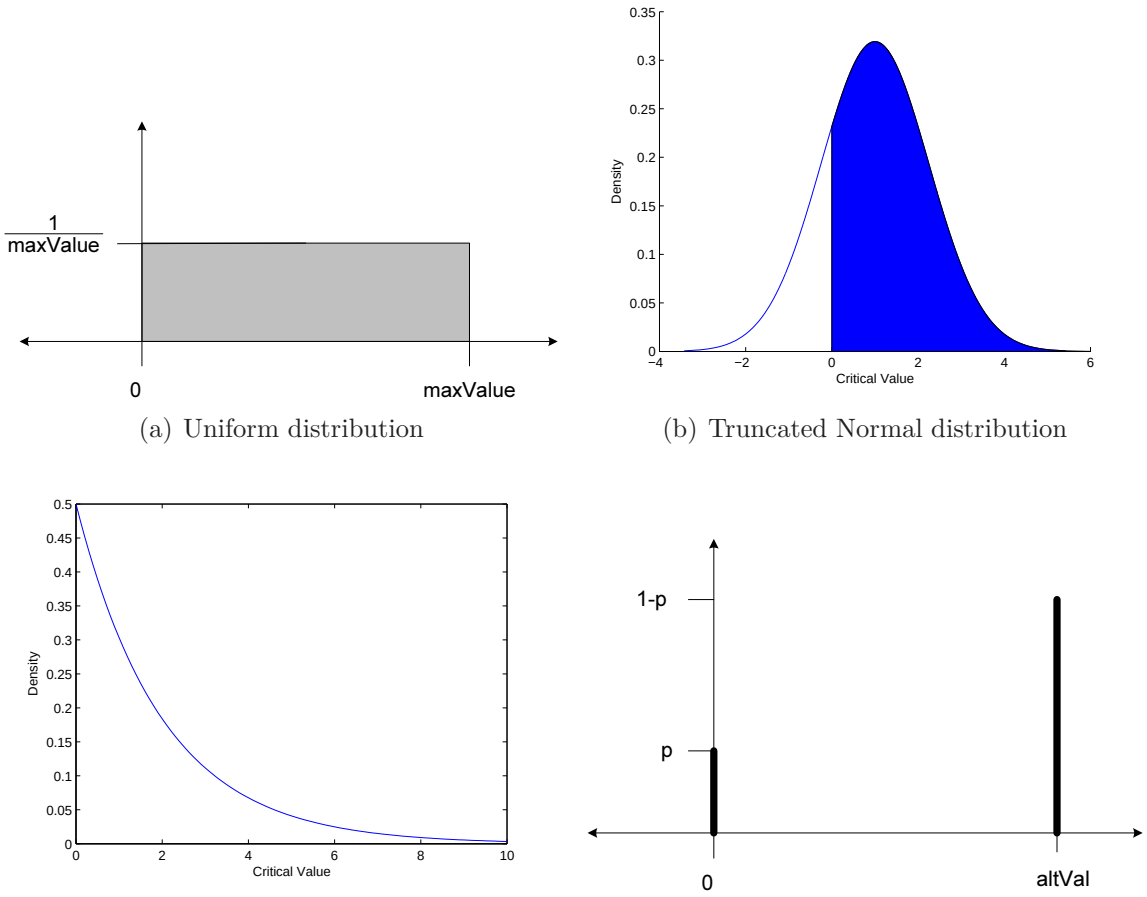


Figure 7.4: The generation of station distances for 80 stations

7.4 Random Number Distributions

A random number distribution needs to be chosen for `vcs_added_backoff`. To this end several distributions are considered and each optimised, and the one giving the best results used.

In the ideal situation, when all the stations select a value for `vcs_added_backoff`, one station would select a back-off time of zero seconds, and all the other stations would select a back-off time that is sufficient to allow the first station to send its RTS without it colliding with the other stations' RTSs. This would be easy if the stations knew what value the other stations were selecting for their back-offs, but this is practically never the case. As it is, one has to make use of a statistically predictable function, and choose it so that it comes close to modelling this ideal behaviour. The distributions selected for testing are discussed below.



(a) Uniform distribution

(b) Truncated Normal distribution

(c) Exponential distribution

(d) Boolean distribution

Figure 7.5: Random number distributions

Uniform Distribution

This is one of the simplest distributions and all values between zero and a specified `maxValue` have an equal probability of occurring.

Normal Distribution

The normal distribution has a bell-shaped curve and has two parameters: mean and standard deviation. The distribution normally extends from negative to positive infinity, but, as there cannot be a negative value for a back-off time, it is truncated at zero.

The normal distribution was selected for testing as it was thought that using this curve with a mean close to zero (and thus a high probability of a small back-off time) would be a good choice.

Exponential Distribution

Initial optimisations of the above normal distribution showed good results for a normal distribution with a negative mean and a relatively large standard deviation. The problem with truncating this distribution at zero is that several values of the normal distribution have to be generated before a positive (usable) value is found. This can become processor intensive as the mean value becomes more negative.

A normal distribution curve which has been truncated on the positive side of the mean value resembles the exponential distribution, and it is thus also considered.

IntUniform, IntNormal and IntExponential Distribution

There is another alternative to the ideal case described at the beginning of this section, where the first station does not necessarily select a value of zero, but where the probability of the second station selecting a time that does not collide is high. Consider the case where all stations are the same distance from the satellite. Instead of using a continuous PDF (probability density function), a discrete PDF is used, where all possible values are greater than the time it takes for a station to transmit its RTS. Now, if only one station selects the smallest value for its back-off, it is guaranteed to transmit a successful RTS.

In practice the stations are not all the same distance from the satellite, and t_{RTS} is thus not guaranteed to be an optimal value. The value can, however, be varied for each of the distributions to determine if optimisation of this parameter is possible. This parameter is referred to as $par3$.

The resulting distributions are called IntUniform, IntNormal and IntExponential distributions, as the PDF contains values which are integer multiples of $par3$ apart.

Boolean Distribution

Consider again the ideal case described at the beginning of this section. It has one station selecting a value of zero and the rest a sufficiently large value. A distribution can thus be defined which has only two possible values: either zero, with probability p, or a larger alternative value `altVal`, with probability $1 - p$.

7.5 Optimisation Results

Initial runs were done with five ground stations, and the results are presented in this section. Note that, contrary to findings in Section 4.10.6, the optimisations were run

with a message length of 2246 bits. This was the result given for initial calculations of the optimal packet length, and the error was only discovered after optimisation and simulation were well under way. It should not severely affect the optimisation of the back-off random variable, although it will slightly affect the efficiency, and therefore the throughput, of the system. A simulation is run with varying message lengths, and the results are presented in Figure 8.10 on page 113.

All the graphs shown plot the average success time for an RTS on the y-axis of 2-dimensional graphs and on the z-axis in the case of 3-dimensional graphs. A small value for this average success time will result in a large throughput, and we are thus concerned with the minimum points on these graphs. The graphs are grouped together at the end of this chapter, and are discussed in the following sections.

7.5.1 Uniform Distribution

It was expected that sweeping `maxVal` would produce a parabolic curve. The resulting curve, however, has an additional spike that peaks where `maxVal`$\simeq 0.039$, as can be seen in Figure 7.7 (a). In order to explain this one must refer again to the situation described in Section 7.2 and shown in Figure 7.2. It was noted that if the first station chooses a back-off value of zero and the second station selects any value between 15.2ms and 38.4ms a collision will result.

When `maxVal`$\simeq 0.039$, the average back-off time selected would be approximately 20ms, which lies in the 15.2ms to 38.4ms band. This results in the observed spike.

An optimal value of $0.08043s$ was achieved where `maxVal`$= 0.01931$.

7.5.2 Normal Distribution

The normal distribution has two parameters, and sweeping these two thus results in the 3-dimensional plot shown in Figure 7.7 (b).

When the standard deviation becomes small the stations back-off times are not diverse enough, and this results in the poor performance shown in the graph. When the standard deviation becomes very large the back-off values become widely spread (extremely diverse), and it is less likely that the first successful station has a back-off close to zero, and this results in decreased performance.

Need for `minProb`

The truncated normal distribution which is used is obtained by repeatedly generating numbers from the normal distribution until a non-negative number is generated. When the mean becomes negative and the standard deviation relatively small, the probability of a random value being non-negative becomes small. Figure 7.6 shows the distribution where $\mu = -0.5$ and $\sigma = 0.26585$. In this case there is only a 3% chance of a generated number being non-negative. Otherwise stated, on average approximately 33 numbers must be generated before a non-negative value is obtained.

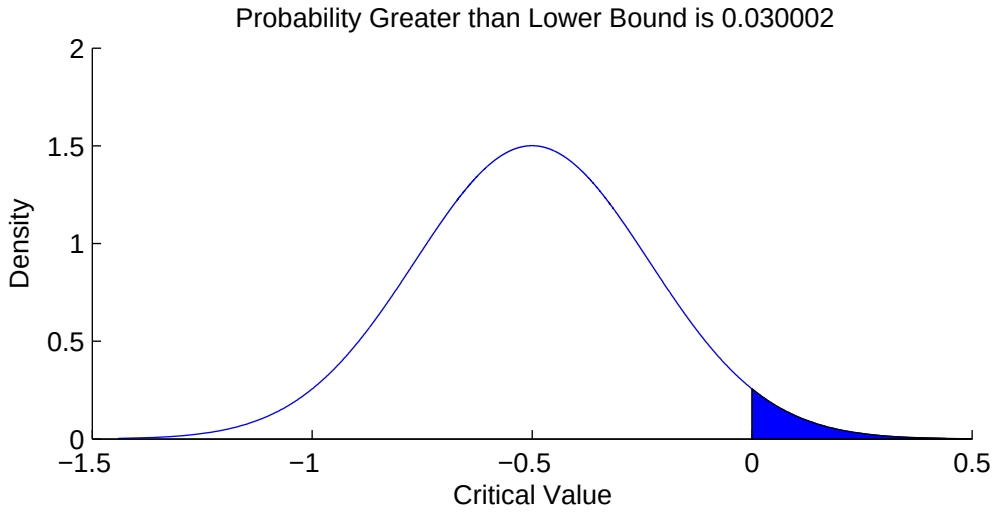


Figure 7.6: Truncated normal distribution with $\mu = -0.5$ and $\sigma = 0.26585$

When the optimisation scripts are run an extremely large number of random number generations are done. If each of these number generations has to be done 33 (or more) times over, then the scripts take very long to run and are no longer useful to us as a shorter alternative to full simulation runs.

Variable `minProb` was defined, and simulation runs that use a distribution with a non-negative generation success rate of less than `minProb` were skipped.

Closer inspection of Figure 7.7 (b) will show that there are a few points missing from the graph in the lower left-hand corner, where the mean becomes more negative and the standard deviation becomes smaller. These were skipped in accordance with the specified `minProb`.

One of the problems with the normal distribution is that its optimal point lies on the boundary specified by `minProb`. This means that as one attempts to get a more optimal value the optimisation scripts run for longer and longer.

An optimal value of $0.1054s$ was achieved where $\mu = -0.08571$ and $\sigma = 0.2314$.

7.5.3 Boolean Distribution

It was expected that sweeping the Boolean distribution's `two` parameters would generate a surface similar to that of the normal distribution, with a single lowest point. However, the same spike is observed as for the uniform distribution, at the same position (values of `altVal`) and for the same reason. Having `altVal` in the region of 15ms to 40ms results in decreased performance. This can be seen in Figure 7.8 (a). The dip on side of the spike where `altVal` is smaller is only slightly lower than on the other side of the spike.

Having a very small $p(0)$ value means that zero is seldom chosen as the back-off value, and the success time is normally roughly equivalent to `altVal`. Having a large $p(0)$ value means that many stations select zero for their back-off value and more collisions will occur, decreasing performance.

An optimal value of $0.0628s$ was achieved where $p(0) = 0.325$ and `altVal`$= 0.01447$.

7.5.4 Exponential Distribution

Sweeping the exponential distribution's mean generates a parabolic curve with a spike as is observed in the uniform and Boolean distribution. This is shown in Figure 7.8 (b).

An optimal value of $0.1794s$ was achieved where $\mu = 0.1794$.

7.5.5 IntUniform, IntNormal and IntExponential Distributions

All three discrete distributions performed better than their continuous counterparts, were optimisable by varying $par3$, and displayed a spike where $par3 \simeq 0.03$. All were erratic for $par3$ values less than 0.03, but were stable for values greater than 0.04, and at a minimum at a value of approximately 0.04. The results are presented in Figures 7.9, 7.10 and 7.11.

IntUniform

Considering again the results for the uniform distribution in Figure 7.7 (a), one notes that there are two minimum points on the graph, at `maxVal`= 0.01931 and at maxVal$\simeq$ 0.2. When using the first, optimal value, the results from sweeping $par3$ are erratic, as can be seen in Figure 7.10 (a). Although it has a minimum point this does not seem to be part of a general trend in the curve, and can not be relied upon as an optimal point. The value of $par3$ cannot be swept higher than maxVal or else all generated values are rounded off to the same value, and collisions occur all the time.

The second, less optimal value of 0.2 was then used, and produced the curve shown in Figure 7.10 (b). This curve has the same behaviour as the other discrete distributions and shows a definite minimum point.

The first minimum point produced an optimal value of $0.06584s$ where `maxVal`= 0.01931 and $par3 = 0.007788$.

The second, less optimal point produced an optimal value of $0.06621s$ where `maxVal`= 0.2 and $par3 = 0.03732$.

The very slight increase in optimal value at the second point is accepted in exchange for reliability.

IntNormal

An optimal value of $0.08184s$ was achieved where $par3 = 0.03815$.

IntExponential

An optimal value of $0.07749s$ was achieved where $par3 = 0.04236$.

7.6 Selection of Distribution

The results given in Section 7.5 are summed up in Table 7.1. The Boolean distribution gives the best results, and is also very simple and fast to generate, and will thus be used.

Distribution	*Minimum time [s]*
Boolean	0.06280
IntUniform	0.06621
IntExponential	0.07749
Uniform	0.08043
IntNormal	0.08184
Exponential	0.1035
Normal	0.1054

Table 7.1: Minimum average success times

7.7 Scalability

These initial runs were done with only five ground stations, but it is important to know how the system will perform with more ground stations.

Number of stations	*Minimum time [s]*	*p(0)*	*altVal*
5	0.06280	0.325	0.01447
10	0.08439	0.1467	0.01447
15	0.0876	0.0775	0.01447
20	0.08937	0.0775	0.01447
40	0.08204	0.0325	0.01447

Table 7.2: Results achieved using the Boolean distribution

Runs were done to optimise the Boolean distribution for ten, fifteen, twenty and forty stations, and the results are listed in Table 7.2. The graphs for ten and fifteen stations can be seen in Figure 7.12.

Runs were also done for a range of larger numbers of ground stations using the distribution as optimised for forty ground stations. The results of these runs are plotted in Figure 7.13. Understandably, the result for 20 stations is worse than its optimal value.

For as many as 140 stations, the average success time is less than a factor three greater than the average success time for five stations. This would seem to suggest that the system will still provide throughput when there is a high level of demand.

7.8 Successful Stations

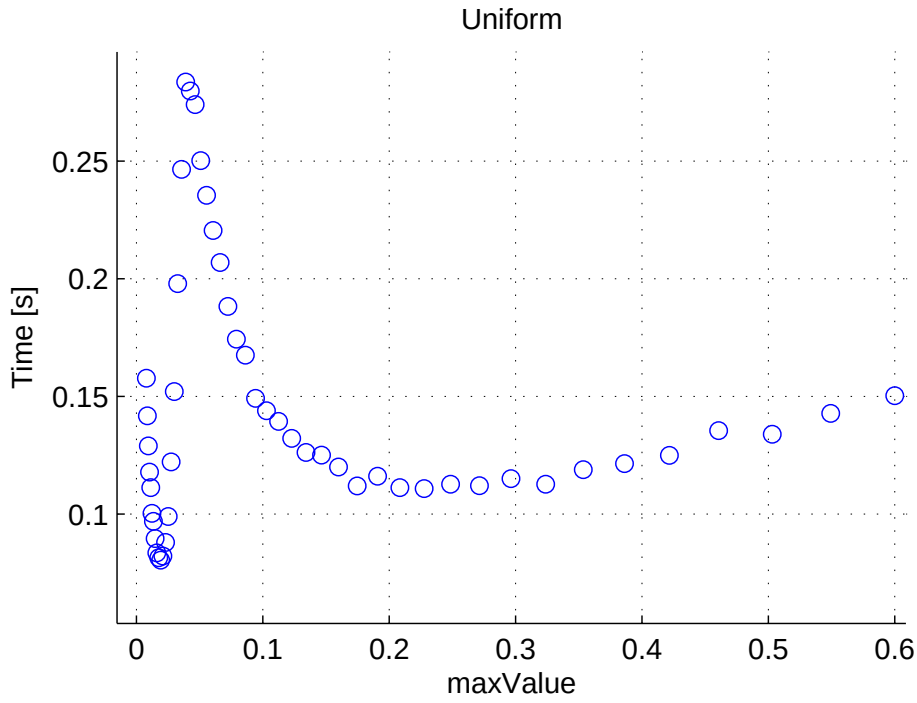
It is expected that stations closer to the satellite will be more likely to send a successful RTS than stations further away, as all stations follow the same back-off

scheme, but the closer stations have a shorter propagation time. A counter was implemented in the optimisation scripts and the results are shown in Figure 7.14. These results confirm expectations.

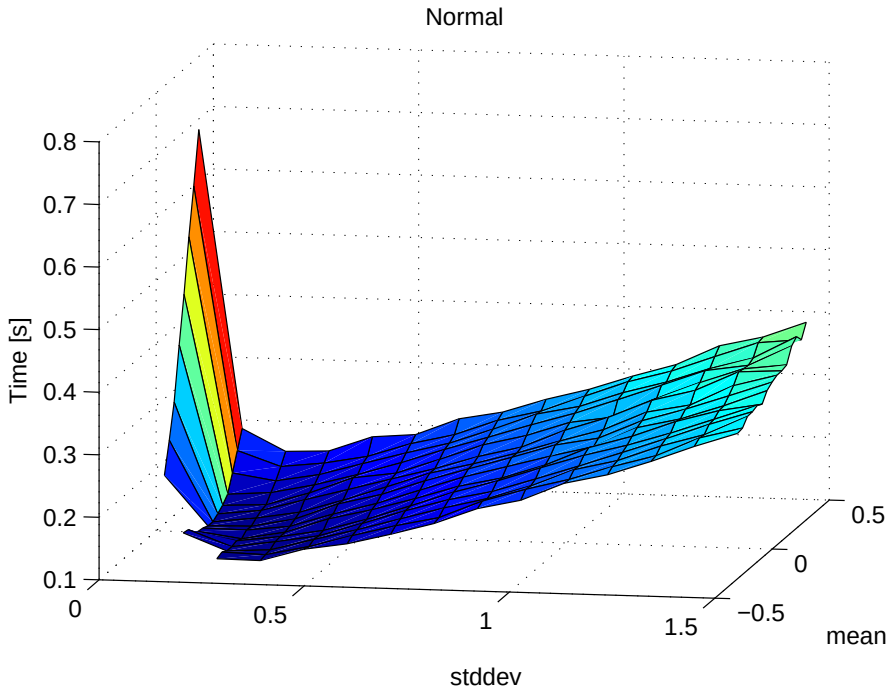
It is important to note that this apparent bias towards closer stations is not something to be concerned about. Each time the satellite passes, it takes a different path, and ground stations which were close to the satellite on one pass may be further away on subsequent passes, and vice versa. The bias will thus be evened out over time, and all stations should ultimately have an equal chance to transmit their messages.

7.9 Summary

- The period to be minimised is defined as the time between when the satellite transmits a CTS-0 or an ACK, and when the first RTS arrives at the satellite without a collision occurring
- MATLAB was selected to run the optimisation scripts
- The following random number distributions were considered for the variable `vcs_added_backoff`:
  - Uniform distribution
  - Truncated normal distribution
  - Boolean distribution (as defined in this text)
  - Exponential distribution
  - Discrete versions of all the above, except for Boolean which is by definition discrete
- The minimum average success times achieved are shown in Table 7.1.
- Boolean distribution gives the best results
- The discrete distributions all perform better than their continuous counterparts
- Scalability does not appear to be an issue
- Stations closest to the satellite are inherently favoured, but each satellite pass is over a different path, so this is not an issue

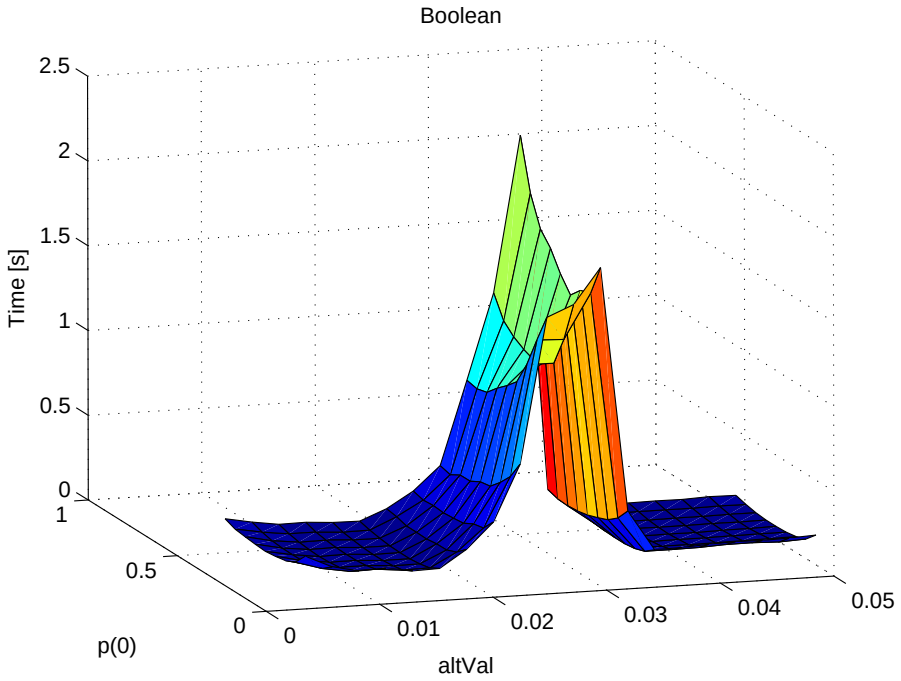


(a) Uniform distribution

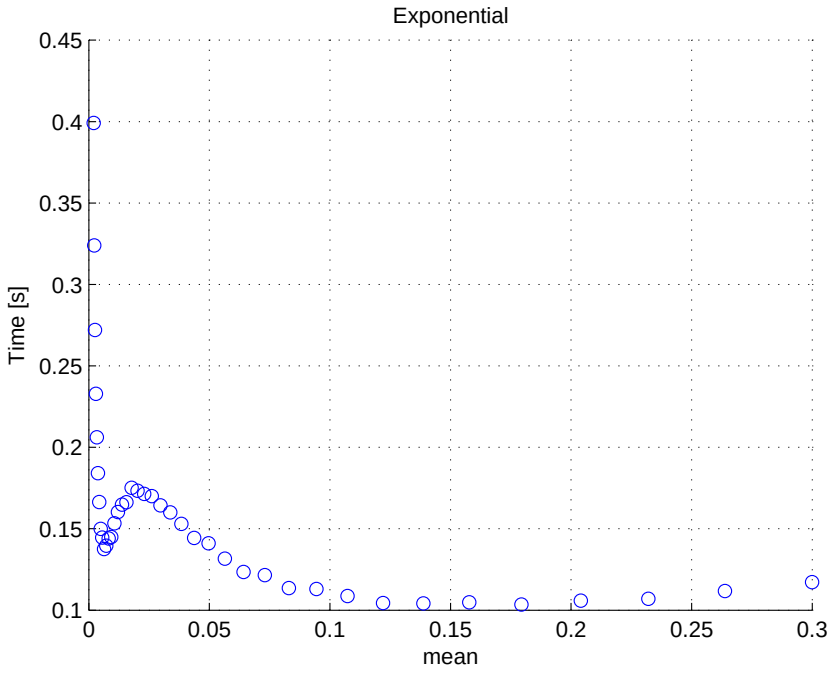


(b) Normal distribution

Figure 7.7: Uniform and Normal distribution results for N=5

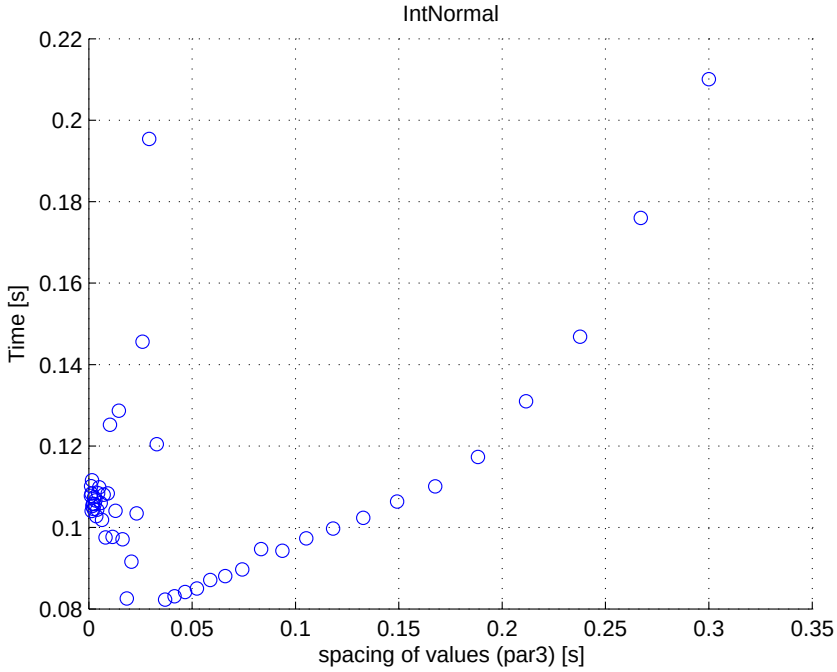


(a) Boolean distribution

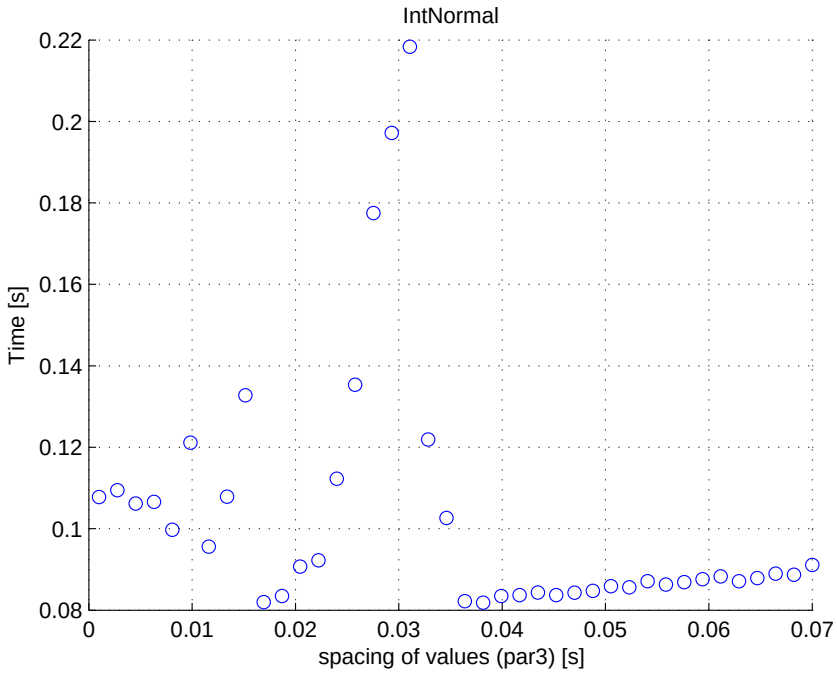


(b) Exponential distribution

Figure 7.8: Boolean and Exponential distribution results for N=5

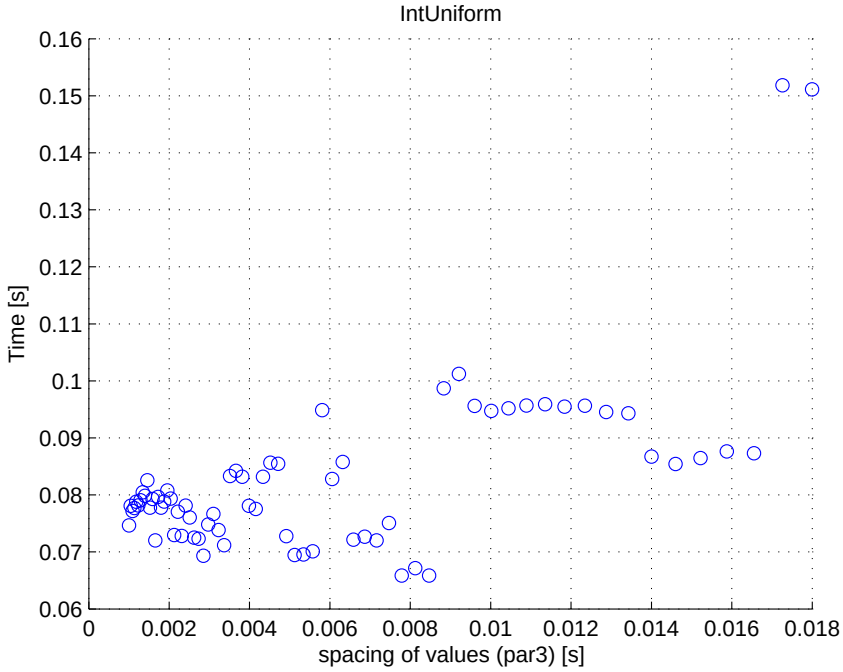


(a) IntNormal distribution

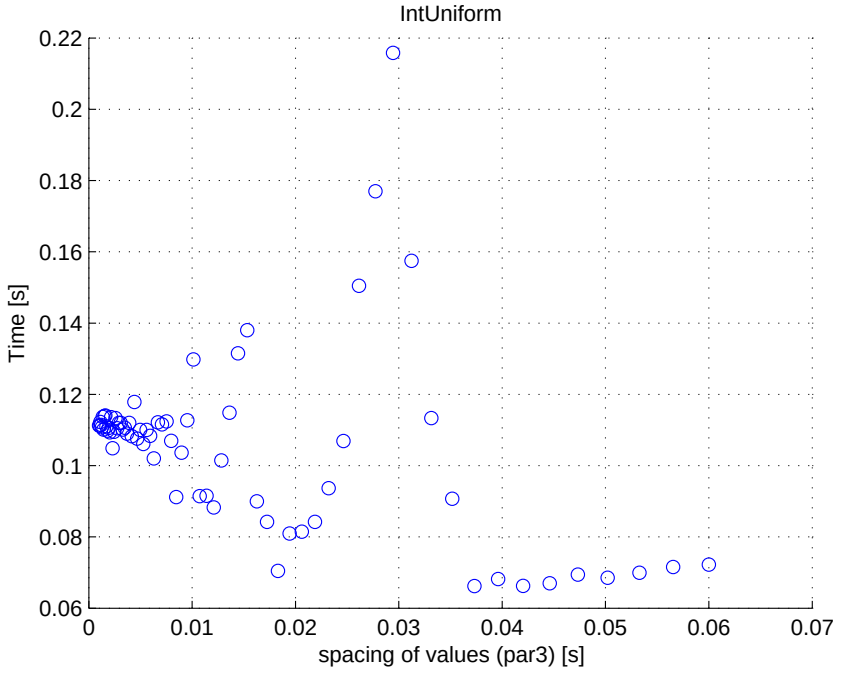


(b) IntNormal distribution

Figure 7.9: IntNormal distribution results for $\mu = -0.45$ and $\sigma = 0.7$



(a) IntUniform distribution results for maxVal=0.01931



(b) IntUniform distribution results for maxVal=0.2

Figure 7.10: IntUniform distribution results for N=5

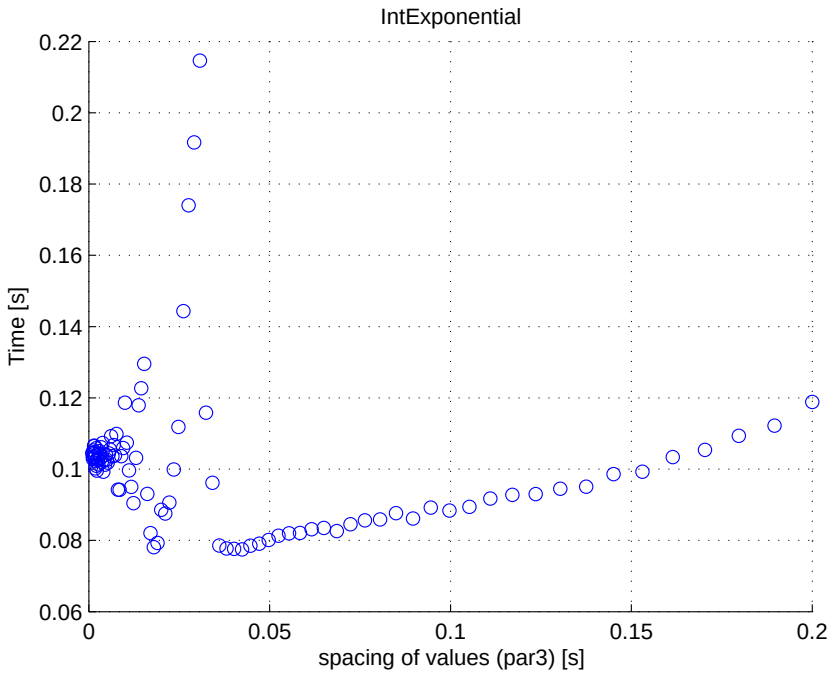
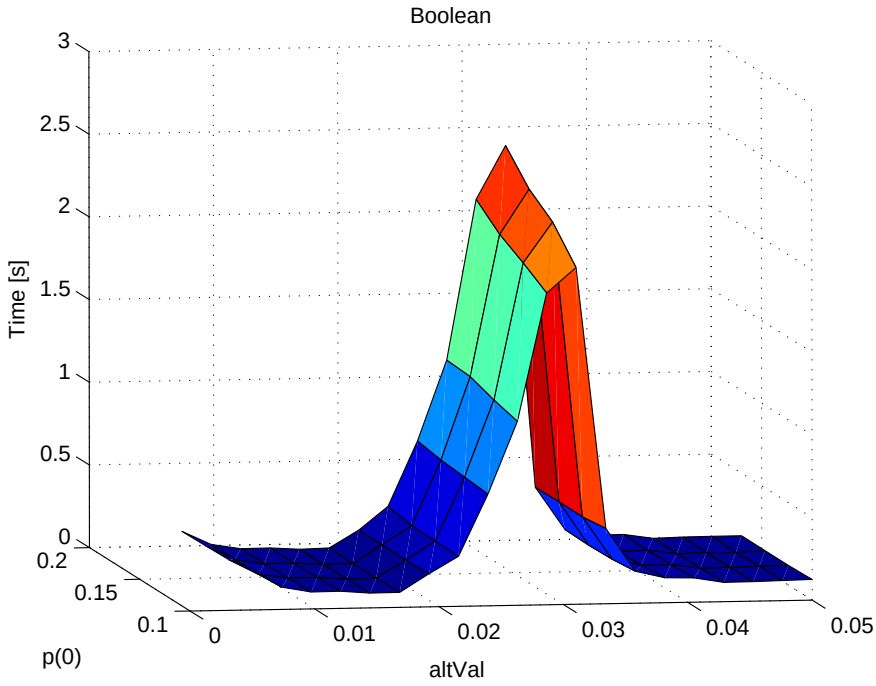
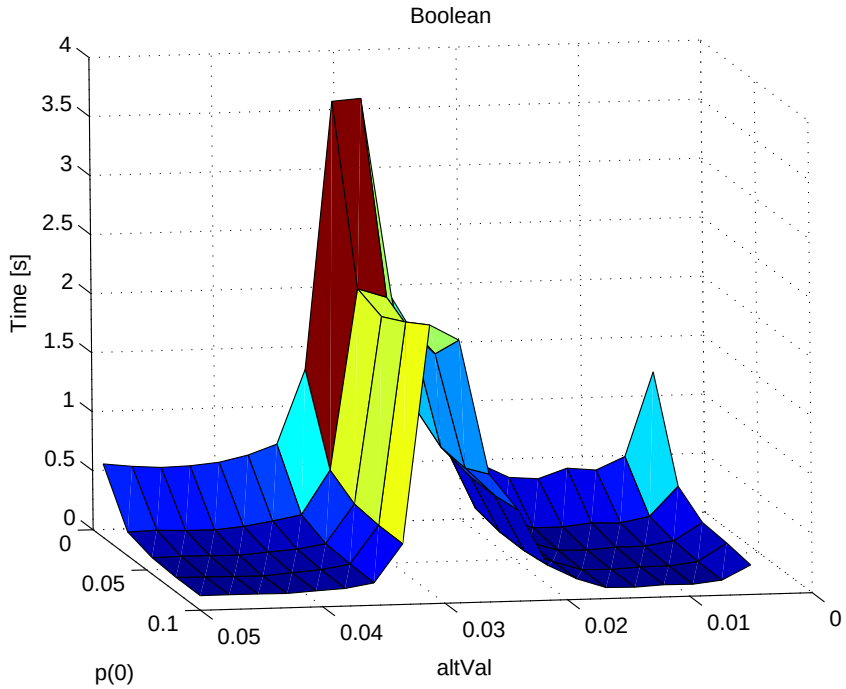


Figure 7.11: IntExponential distribution results for $\mu = 0.1794$



(a) N=10



(b) N=15

Figure 7.12: Boolean distribution results

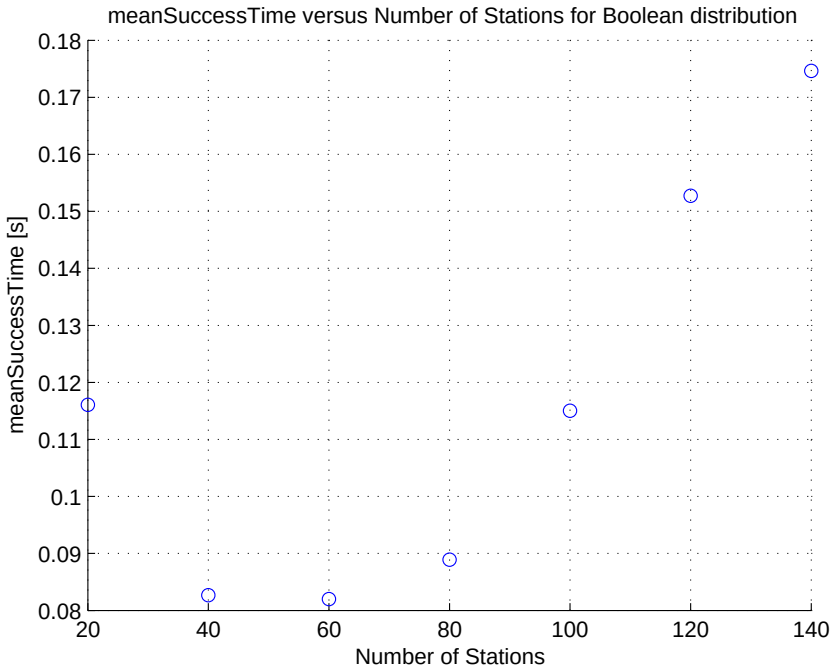


Figure 7.13: Boolean distribution results for varied number of ground stations

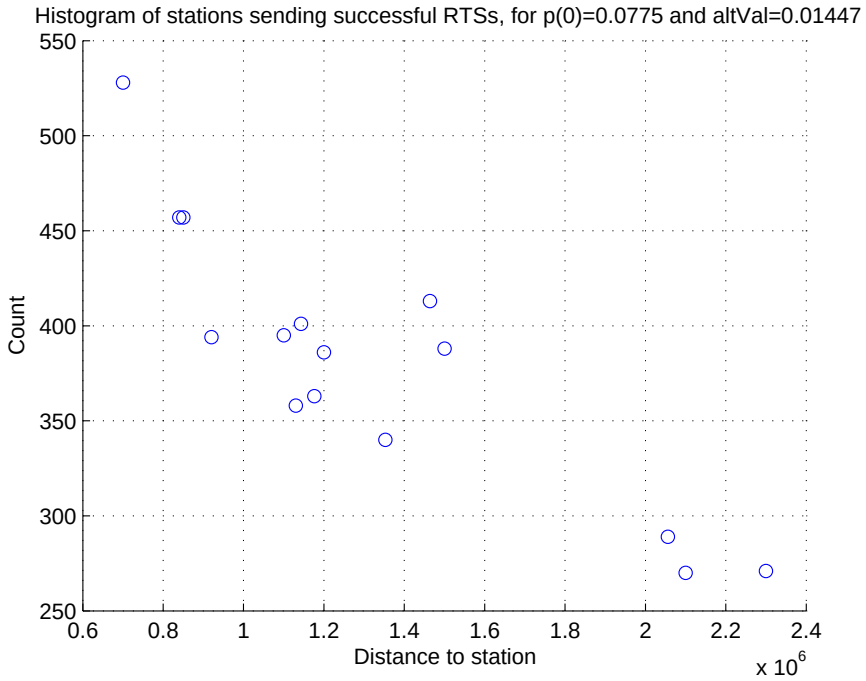


Figure 7.14: Histogram of successful ground stations

Chapter 8

Simulation Results

In the previous chapter MATLAB scripts were used to give an indication of how the choice of random number distribution (for the back-off variable) would affect the throughput of the system. What now remains is to run full simulations of the system. This will allow one to see to what extent the optimisation scripts can be used in selecting optimal back-off distributions, and will also quantify important values such as system throughput.

This chapter presents the results of the OMNeT++ simulations, and compares these results with the predictions of the optimisation in the previous chapter.

8.1 Results of OMNeT++ Simulations

The optimisation scripts that were run plotted mean success time on the y-axis (or z-axis, in some cases). As the mean success time decreases, the rate at which message packets are scheduled and sent increases, and the system throughput thus increases. The converse is also true. We thus expect to see the throughput graphs displaying an inverse behaviour to the mean success time graphs.

Most of the graphs plot half-throughput per 12 minute period. The majority of the simulations were run for 2880 simulated seconds (48 minutes), and the result divided by four. Longer runs were made when 2880 seconds was not long enough to guarantee the convergence (and thus accuracy) of results. Half-throughput is the amount of useful data that is sent through the system. When data is transmitted from one station to another it has to be transmitted twice, first from one station to the satellite, and secondly from the satellite to the other station. The half-throughput value is thus a truer reflection on the throughput ability of the system than the full throughput.

The simulations presented here are for 5 ground stations, unless otherwise noted. A non-optimal average message packet length of 2250 is used, as is explained in Section 7.5. Simulations are run to determine the optimal message packet length, and the results are shown in Section 8.1.6.

8.1.1 Boolean Distribution

The Boolean distribution has two parameters, $altVal$ and $p(0)$. First $altVal$ was swept to find the optimal value, and then this optimal value was used while $p(0)$ was swept to find the optimal combination of the two values.

An initial simulation run of the Boolean distribution produced the results shown in Figure 8.1 (a). Note the dips in throughput at $altVal$ values of approximately 0.03, 0.15 to 0.2, and 0.25 to 0.31.

The first dip is the one forecast by the MATLAB optimisation (as shown in Figure 7.8 (a)). In the optimisation we found local minima on either side of the spike at 0.03. We do not, however, find a local maximum on the left-hand side (smaller value of $altVal$) of the corresponding dip at 0.03, but only on the right-hand side. This does not matter too much, as it was noted in Section 7.5.3 that the left-hand side optimal point would only give very slightly better performance than the right hand side.

The other two dips were not expected, and must thus be due to parameters not modelled by the optimisation scripts. The average message length that is used for the simulations is 2250 bits. The transmission time for a message of this length is $234ms$, and half of this time is $117ms$. Note that $117ms$ falls in the dip between 0.15 and 0.2 as these values of $altVal$ are defined in seconds. The protocol defines that a station finding the channel busy should back-off for half the message transmission time ($117ms$) before attempting again. It would appear that setting $altVal$ to this same time causes a significant drop in system throughput.

The value for the channel-busy back-off was changed to $234ms$ (a full message packet transmission time period) and the results are shown in Figure 8.1 (b). Note that the dip between 0.15 and 0.2 has disappeared. There also seems to be a slight change to the dip at 0.03.

The value was shifted further to $600ms$, and the results are shown in Figure 8.2 (a). Note that this has also lessened the severity of the dip at 0.03, and slightly improved the optimal throughput. One can now see that $altVal$ has an optimal value of 0.04.

Figure 8.2 (b) shows $p(0)$ being swept to find the optimal combination of the

two parameters.

An optimal half-throughput of 151kB was achieved where $altVal = 0.04$ and $p(0) = 0.18$.

8.1.2 Normal Distribution

The normal distribution has two parameters, *mean* and *stddev* (the standard deviation). Both these parameters are swept and the resulting surface is plotted in Figure 8.3 (a). The optimal values of *mean* and *stddev* are then used for the IntNormal distribution, and the results are shown in Figure 8.3 (b), where a clear optimal value of $par3$ is discernable.

Figure 8.4 gives closer inspection to the area in the graph at 0.03, and the expected drop in throughput is clearly visible, to correspond with the predictions in Figure 7.9 (b).

An optimal half-throughput of 149kB was achieved where $mean = -0.08571$, $stddev = 0.1257$ and $par3 = 0.045$.

8.1.3 Uniform Distribution

The uniform distribution has only one parameter, $maxVal$, and a sweep of this parameter gives the results presented in Figure 8.5 (a). The maximum value of throughput occurs where $maxVal \simeq 0.2$. Note that there is a local maximum occurring at $maxVal = 0.01$ and a local minimum at $maxVal = 0.02$. These show the behaviour predicted by Figure 7.7 (a), although the points at which they occur differ slightly from those predicted. Note also that, whereas the first of the local minima (at $maxVal = 0.02$) was the optimal point, it is the second of the corresponding local maxima which is shown to be optimal by the simulation. This second point was also selected as the optimal point in Section 7.5.5 as it is more reliable.

$maxVal = 0.2$ is used for the IntUniform distribution, and the results of sweeping $par3$ are presented in Figure 8.5 (b). A broadish area of maximum throughput is discernable where $par3 = 0.07$.

Note that the graph flattens off when $par3 \geq 0.4$. Remember that the IntUniform distribution takes the generated number and rounds it off to the closest integer multiple of $par3$. When $par3$ is more than double the maximum number that can

be generated ($maxVal$) then all generated numbers are rounded off to zero, and increasing $par3$ further will have no effect. Confirmation of this is shown in Figure 8.6 (b), where $maxVal$ is set to 0.1, and the graph subsequently flattens off at 0.2.

Figure 8.6 (a) gives closer inspection to the area around $par3 = 0.03$, and shows that the expected dip is indeed present.

An optimal half-throughput of 150kB was achieved where $maxVal = 0.2$ and $par3 = 0.06$;

8.1.4 Exponential Distribution

The exponential distribution has only one parameter, its mean, and a sweep of this parameter gives the results presented in Figure 8.7 (a). As with the uniform distribution, the behaviour of the curve fits in with what was predicted (see Figure 7.8 (b)), but the points at which local maxima and minima occur differ slightly.

Using the optimal value of $\mu = 0.1184$, $par3$ is swept and the IntExponential distribution gives the results shown in Figure 8.7 (b). Once again, the dip in throughput can be observed at $par3 = 0.03$, and can be seen more clearly in Figure 8.8.

An optimal half-throughput of 149kB was achieved where $\mu = 0.1184$ and $par3 = 0.07$.

8.1.5 Scalability

In order to investigate the scalability of the system, the Boolean distribution is used with its optimal parameters, and the number of ground stations are varied. The results are shown in Figure 8.9. Throughput initially drops as the number of ground stations increases, and then it stabilises when the number of stations is between 25 and 140. This is good news, as it means that having a large number of ground stations will not cause the system to collapse.

Note that the parameters can be optimised for a specific number of ground stations, which will produce better results than those shown in Figure 8.9, which has its parameters optimised for 5 ground stations.

8.1.6 Optimal Message Length

In Section 4.10.6, an optimal message packet length of 4995 bits was calculated for the given system parameters. The above simulations were run with an average mes-

sage packet length of 2250 bits. The results of varying the message packet length is shown in Figure 8.10. This confirms that 4995 is indeed an optimal value (any value in the range 5000 to 7000 bits gives optimal throughput).

A maximum throughput of 162kB is thus possible.

8.2 Summary

Distribution	*Half-Throughput in 12 minutes [kB]*
Boolean	151
IntUniform	150
IntNormal	149
IntExponential	149

Table 8.1: Optimum half-throughput values for $messageLength = 2250$

The following was noted in comparing the simulation results with the optimisation results:

- The behaviour of the system, as investigated through simulation, is, for the most part, as was predicted by the optimisation results.
- Specific behaviour of the mean success time in the optimisation results is reflected by inverse behaviour in the half-throughput simulation results.
- The spike which is present at $par3 = 0.03$ in the optimisation results is reflected in a dip in throughput at the same point in the simulation results.
- The optimisation appears inaccurate in pinpointing the location of maxima and minima for very small back-off values, although the correct behaviour is predicted.

The simulation runs made the following observations possible:

- In Section 4.9.3, a value of half the average message length was suggested for the back-off that occurs when a station senses the channel busy. In observing the simulation results it became apparent that this value was not a good choice, and a value of $0.6s$ was used for the rest of the simulations.

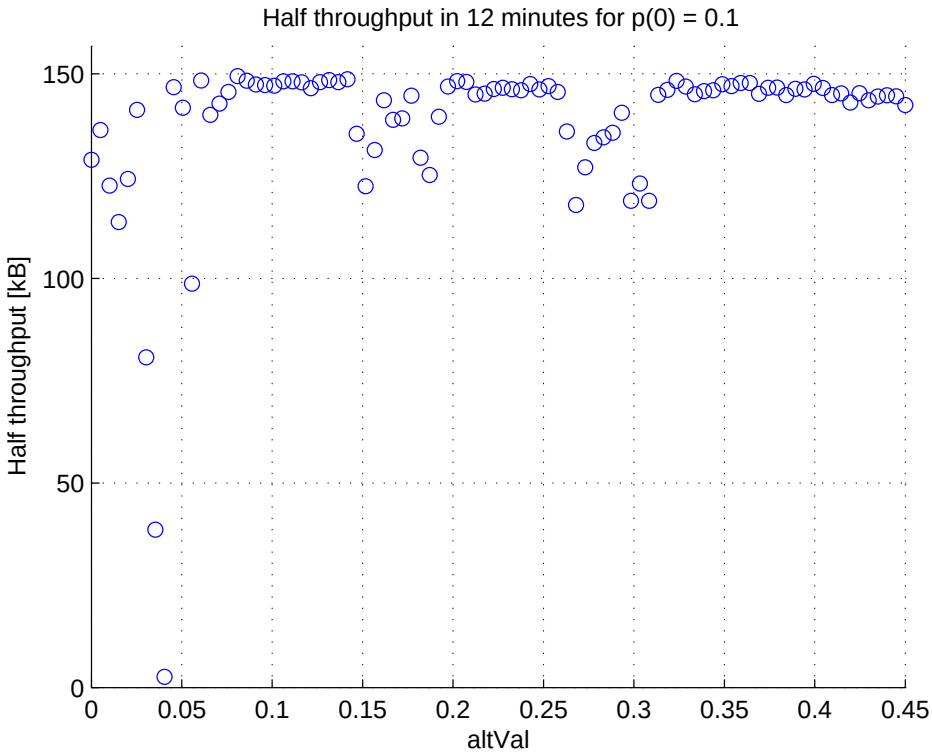
- The average half-throughput values achieved by the various distributions in 12 simulated minutes are shown in Table 8.1. It would appear that it is not critical which distribution is selected for the system. It is, however, important to make sure that the specific distribution is optimised for the system.

- Adjusting the message length showed that 4995 bits, as calculated in Section 4.10.6, is indeed an optimal value of message length, and increases the maximum half-throughput for the Boolean distribution to 162kB.

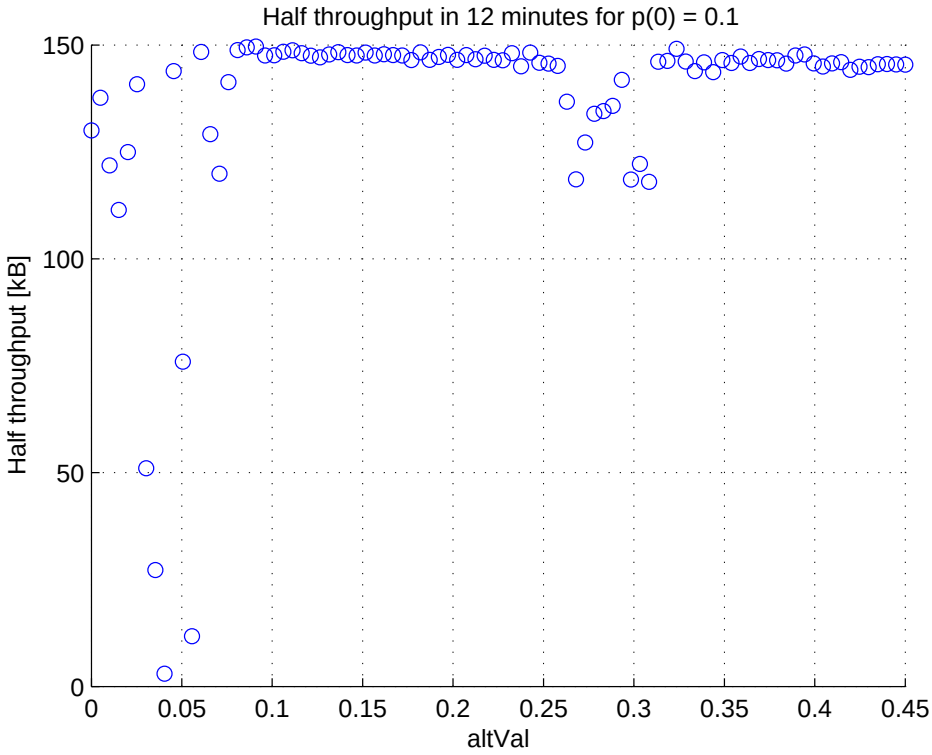
The fact that the optimisations and the simulations agree to a large extent confirms both the validity of the optimisations, and the accuracy of OMNeT++ as a modelling tool for communications systems and protocols.

It was calculated in Section 2.4 that there would be an average of 4 satellite passes per day for an average of 12 minutes each. This gives a total half-throughput of $4 \times 162 = 648kB$ per day. Assuming that the average e-mail is $2.5kB$ in size, this allows $\frac{648}{2.5} = 259$ e-mails to be sent daily.

The concepts of error detection and FEC were introduced in Chapter 5. It is necessary to make use of, at very least, error detection techniques for this protocol. Error detection involves adding redundant data to the message packet, so the usable throughput of the system drops, and the number of e-mails sent will be less. Further investigation is necessary, but one may be able to implement FEC so that the channel is used more effectively, and an increase in the throughput (and thus number of emails) may well be possible.

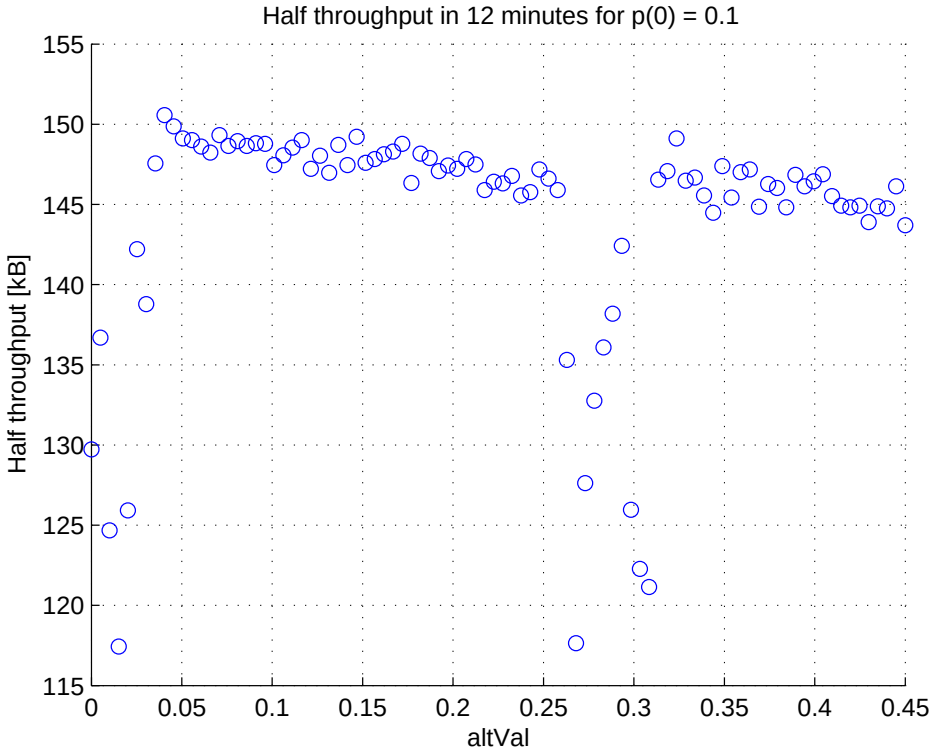


(a) generalBackoff = 0.117 and collisionBackoff = 0.234

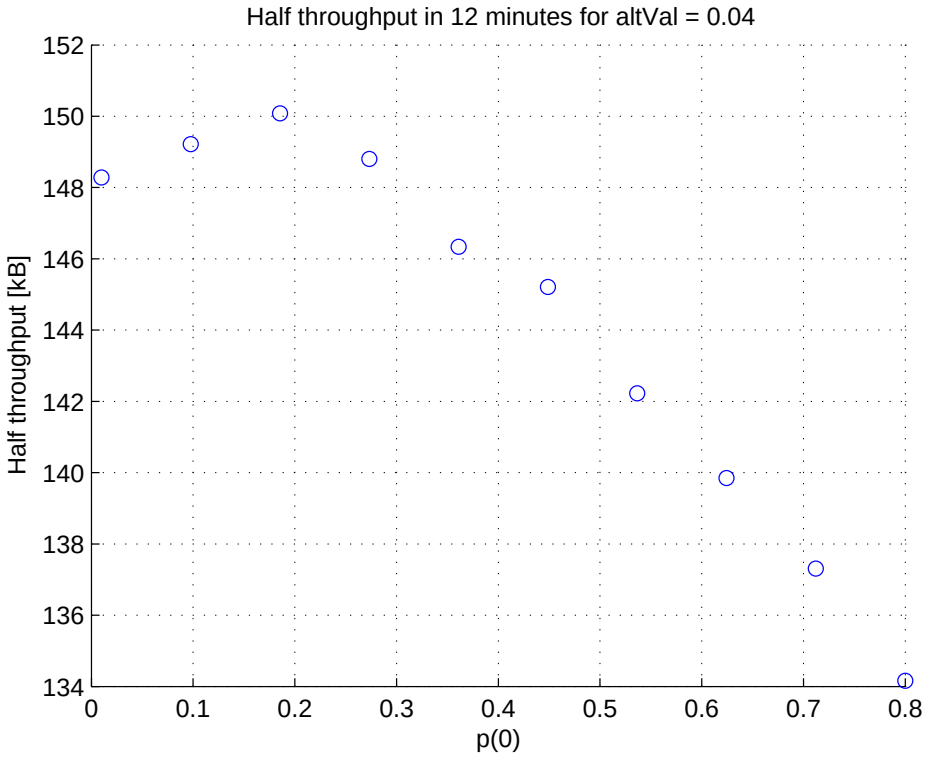


(b) generalBackoff = collisionBackoff = 0.234

Figure 8.1: Boolean distribution - effect of changing general- and collisionBackoff

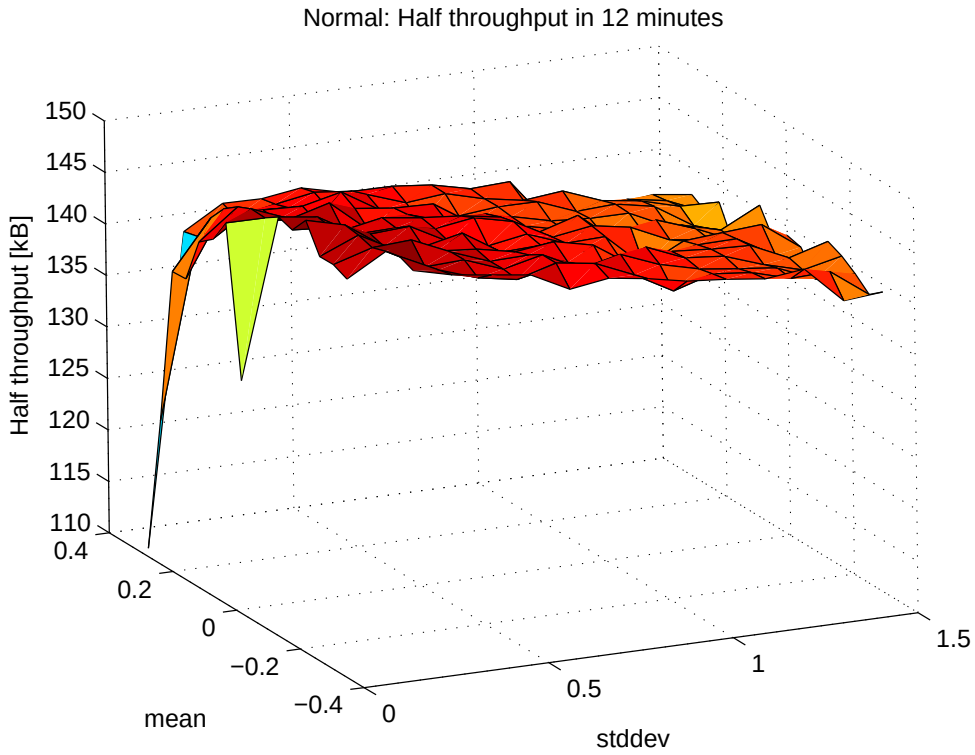


(a) generalBackoff = collisionBackoff = 0.6

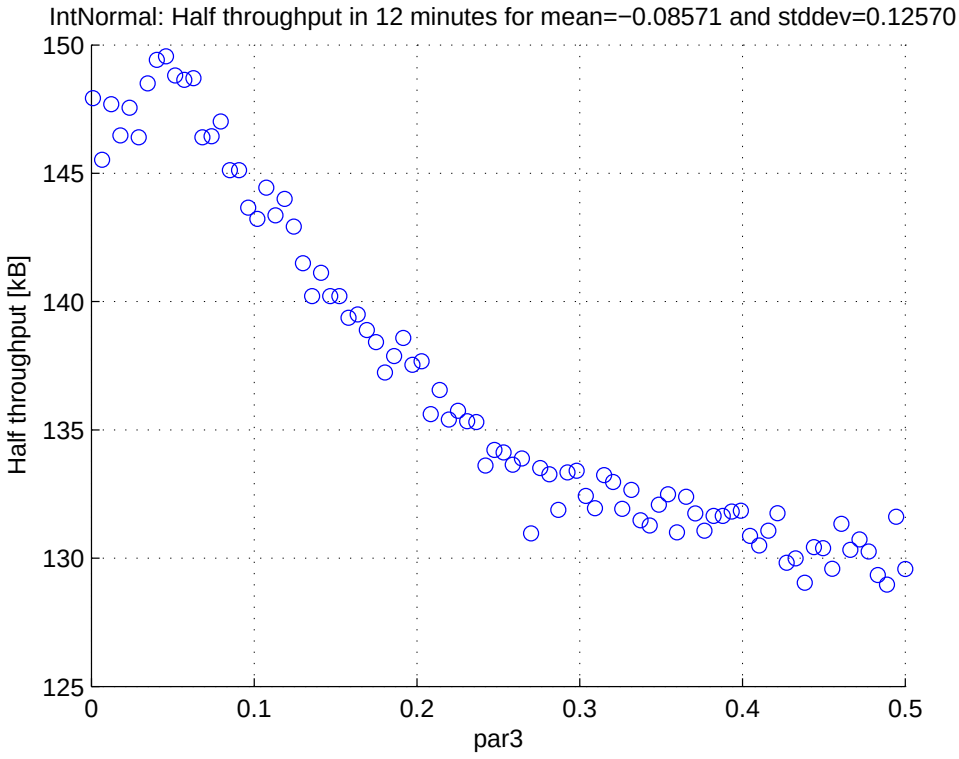


(b) Variation of p(0) with optimal altVal value

Figure 8.2: Boolean distribution - optimal values



(a)



(b) Variation of par3 with optimal mean and standard deviation

Figure 8.3: Normal distribution - optimal values

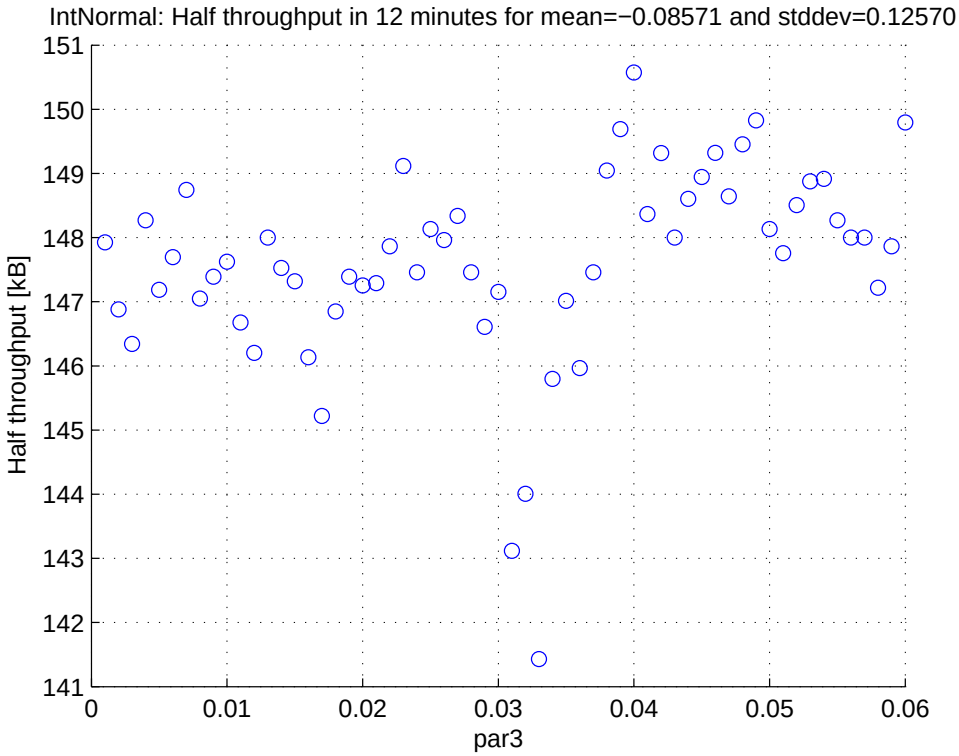
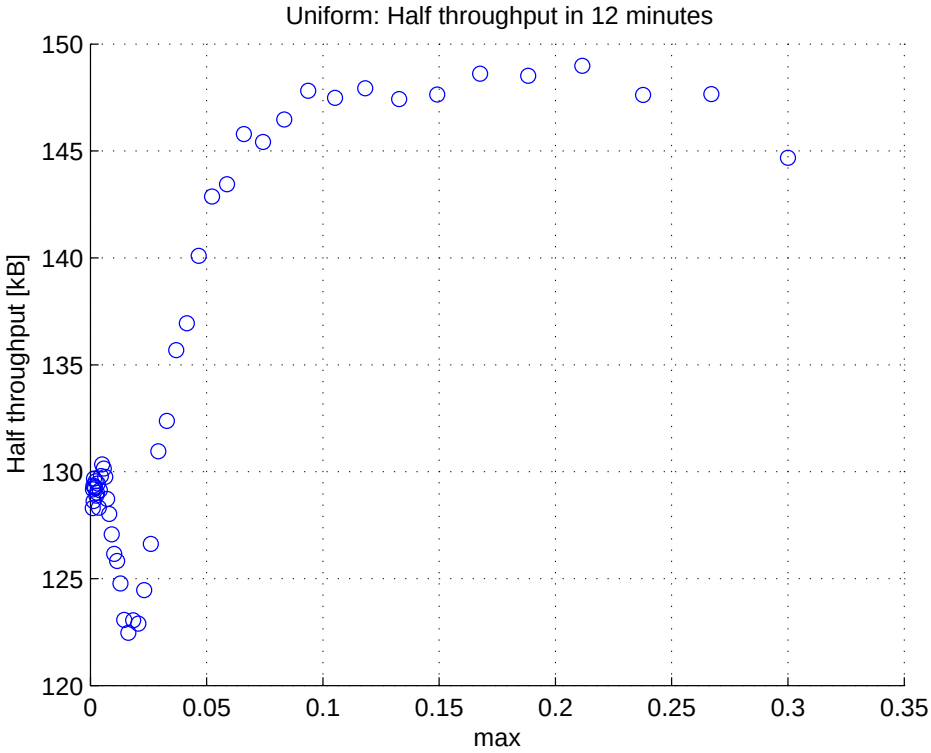
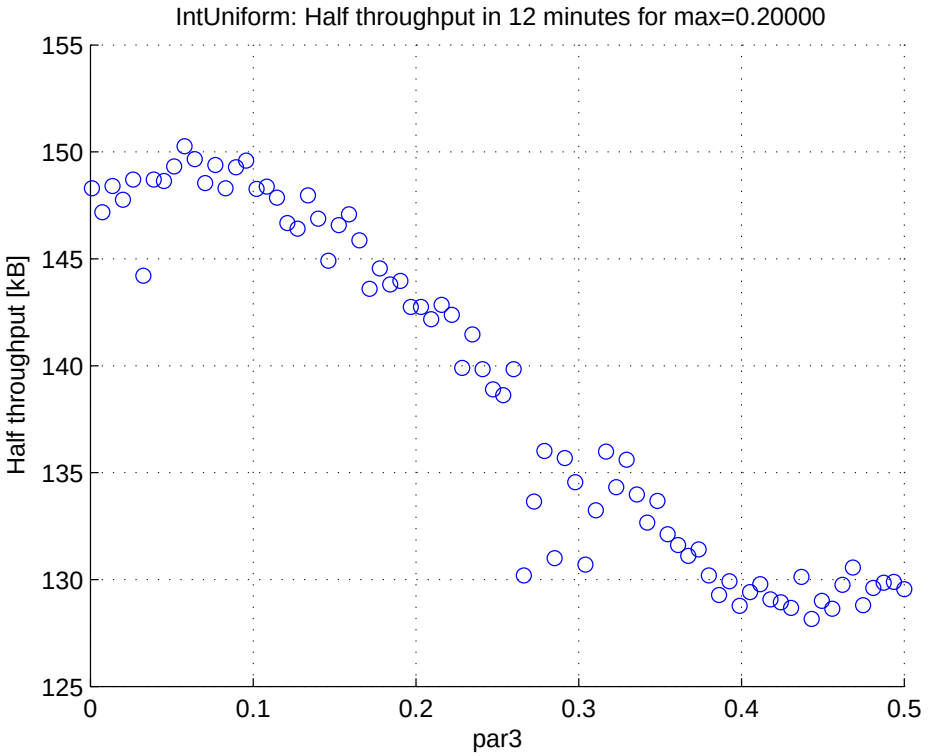


Figure 8.4: IntNormal distribution - closer inspection of dip in graph

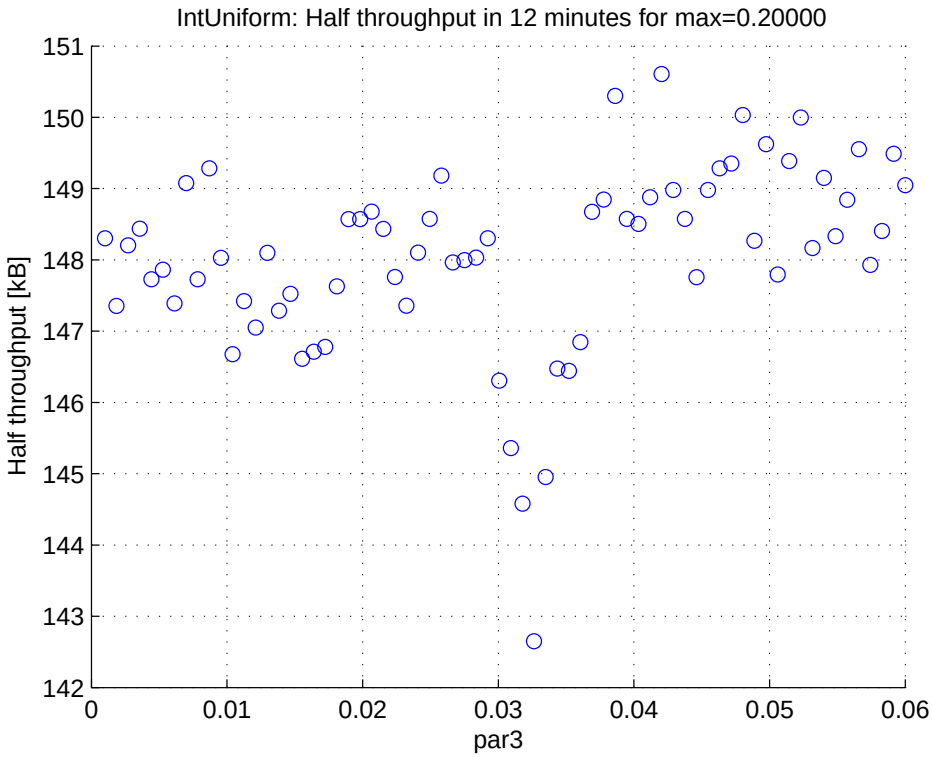


(a)

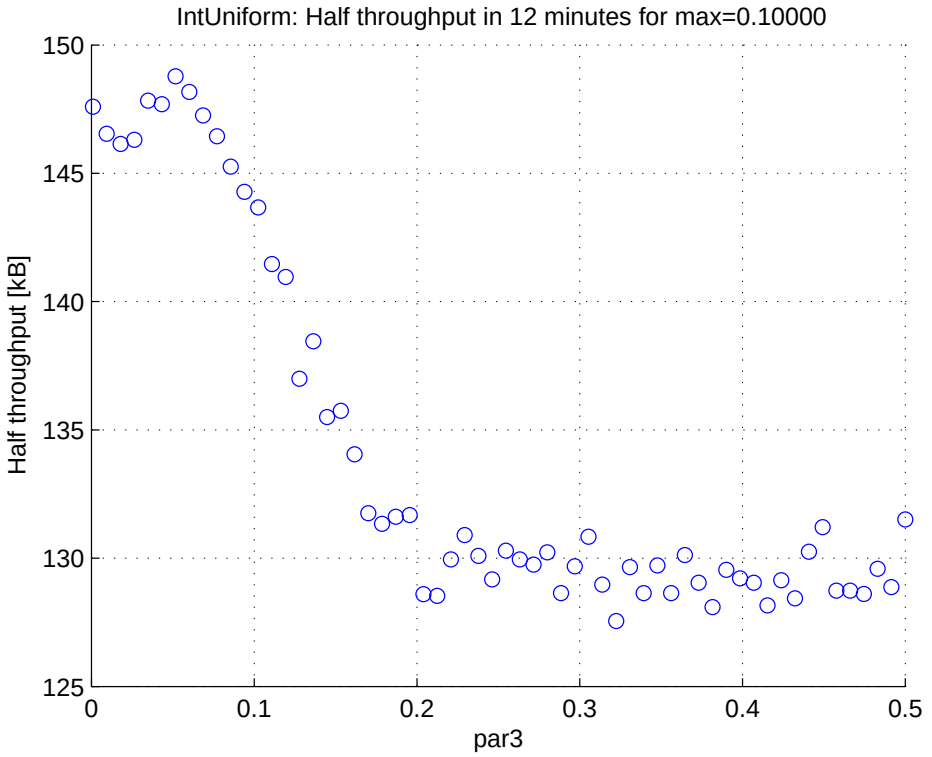


(b) Variation of par3 with optimal maxVal value of 0.2

Figure 8.5: Uniform distribution - optimal values

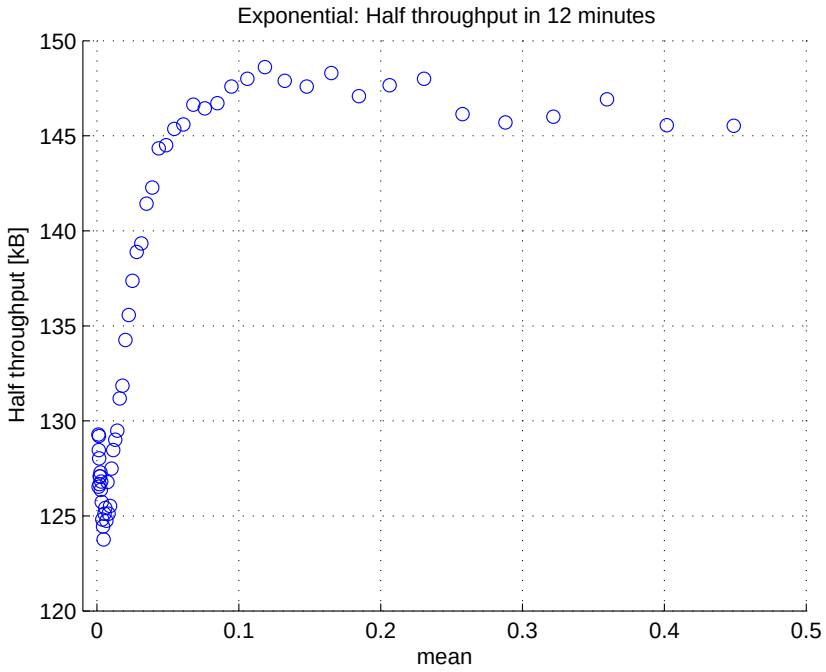


(a) Closer inspection of dip in graph

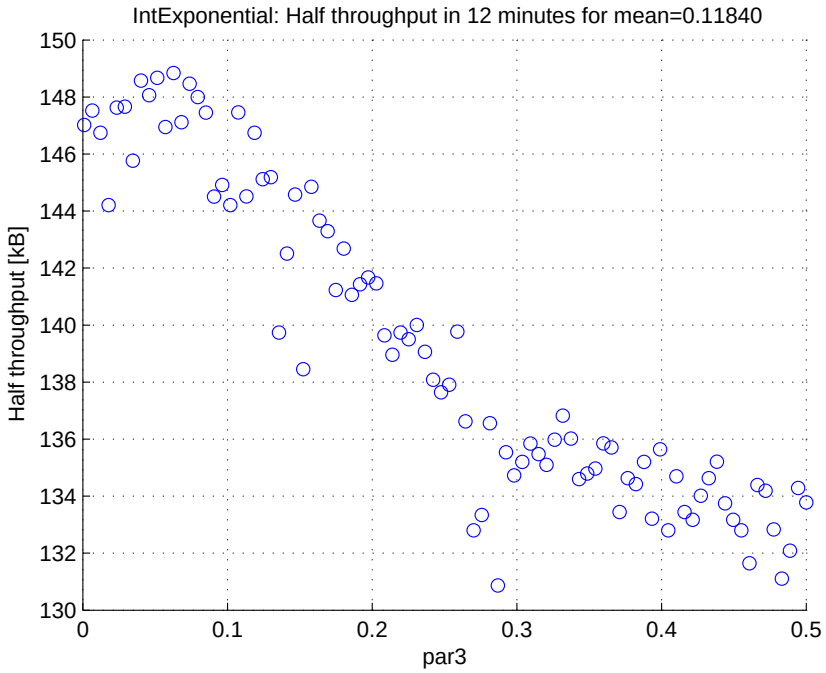


(b) Variation of par3 with maxVal = 0.1

Figure 8.6: IntUniform distribution



(a)



(b) Variation of par3 with optimal mean

Figure 8.7: Exponential distribution - optimal values

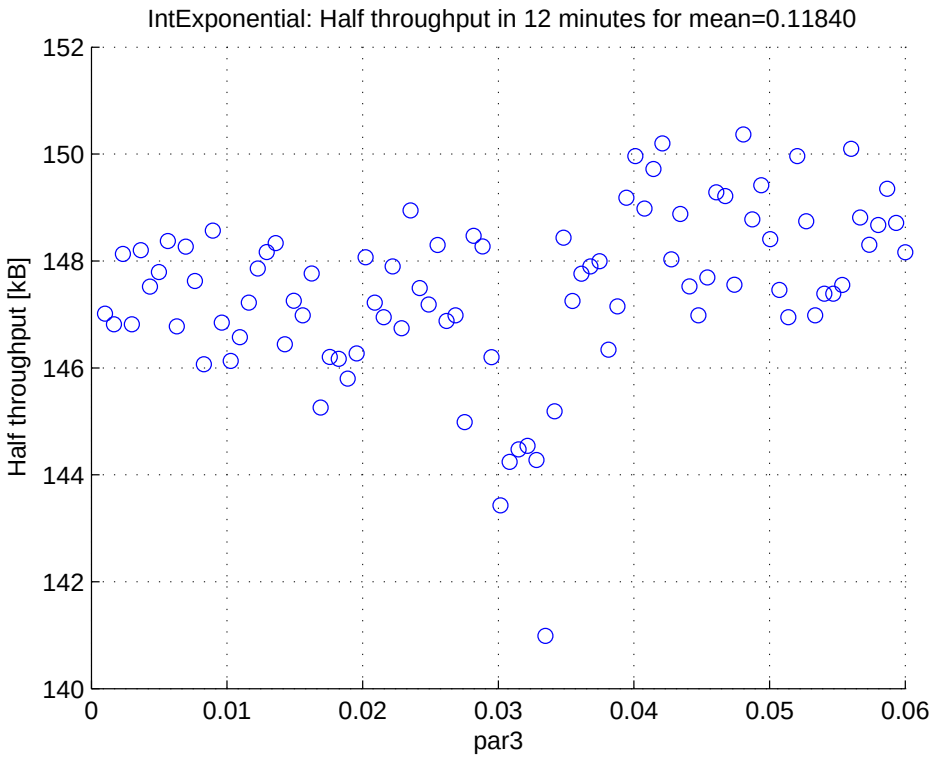
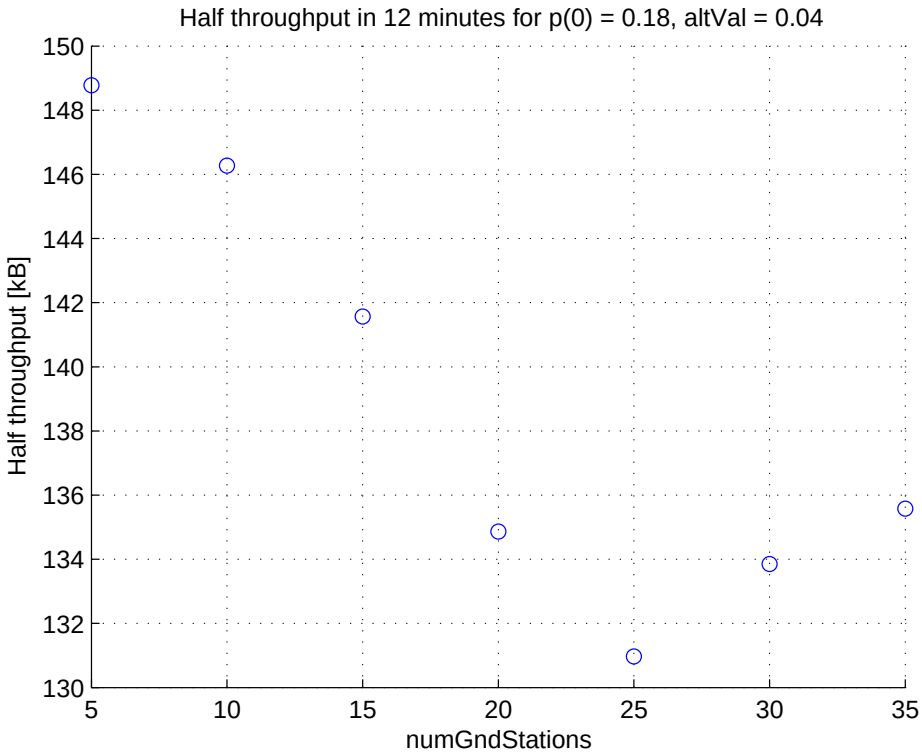
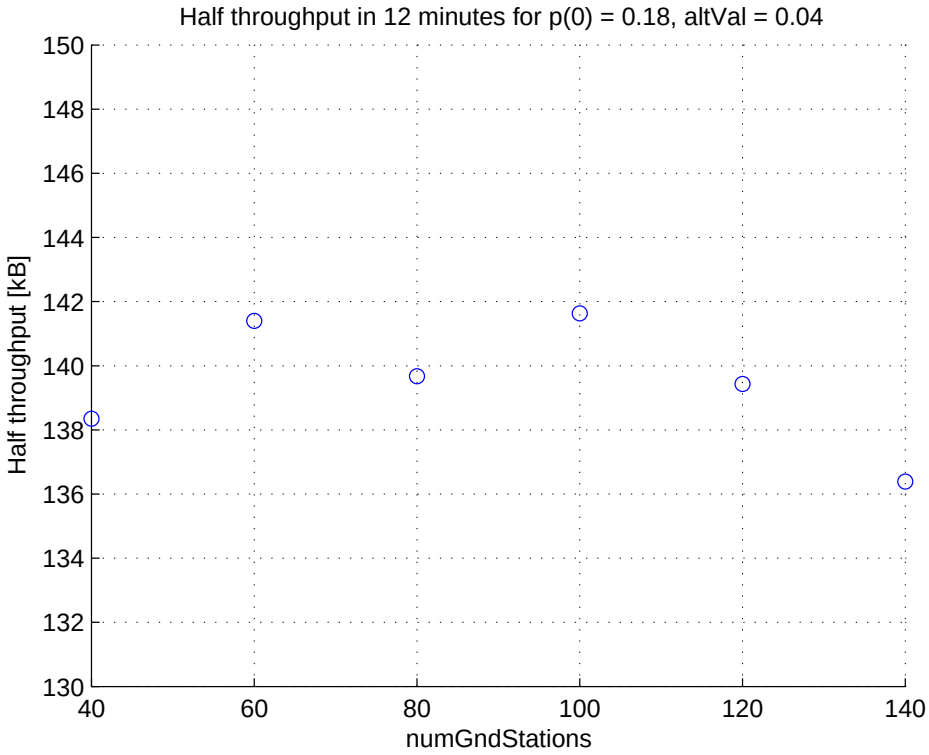


Figure 8.8: IntExponential distribution - closer inspection of dip in graph



(a) N = 5 to 35



(b) N = 40 to 140

Figure 8.9: Boolean distribution - scalability results

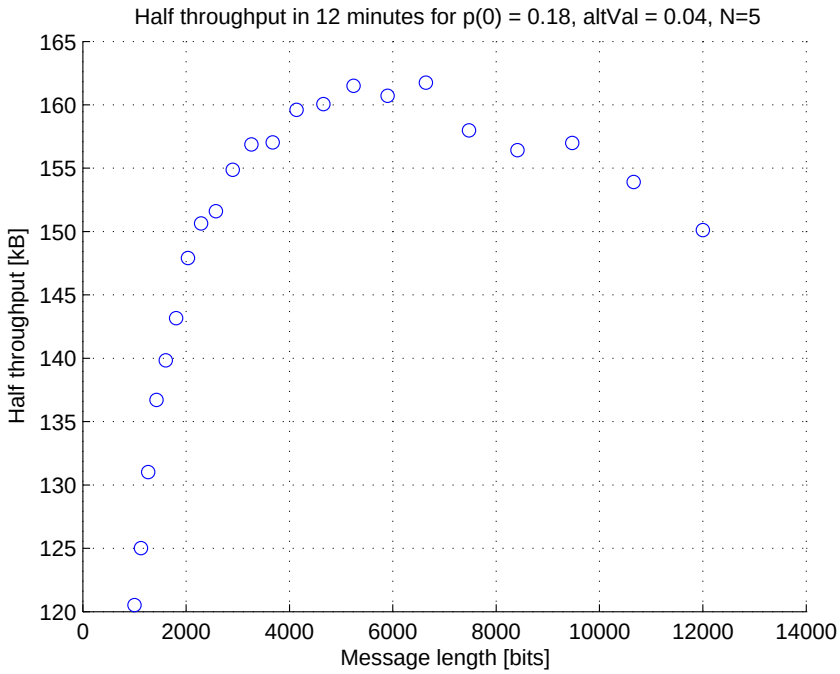


Figure 8.10: Boolean distribution - throughput for various message lengths

Chapter 9

Summary and Conclusion

In concluding this thesis, the following remarks and observations are thought apt:

9.1 Motivation

Sumbandila is the name of a low earth orbit micro-satellite to be launched in December 2006. One of the functions that it will perform is to provide is the necessary infrastructure for e-mail transfer. Ground stations will be set up in rural areas lacking the infrastructure necessary for e-mail, and messages will thus be able to be relayed by the satellite.

9.2 Summary of Objectives

The following were the objectives of this thesis:

- To first research existing protocols used for the multi-user single-channel environment
- Thereafter to make an informed decision as to the best protocol for this specific application (e-mail service)
- To adjust and refine the selected protocol for this specific application
- To build a computer simulation of the system, with the purpose of confirming the protocol's suitability for the system, and of quantifying expected throughput
- To optimise the protocol for maximum throughput

9.3 Summary of Thesis

- First data is gathered and calculations are made to determine what effect the various physical constraints will have on protocol selection. Orbital calculations are made, the limitations of the receiving hardware are measured, and a link budget is calculated.

- Various applicable communications protocols are investigated, and their pros and cons weighed up. CSMA/CA is selected as the best protocol for the application specific to this thesis.

- Various refinements are made to the CSMA/CA protocol in order to improve its efficiency for this application. The protocol is defined in flow-chart format, and the final protocol can be seen in Appendix A. The addition of counters and timers is explained. Back-off strategies are discussed. The packet structure for data transmissions is specified. Sync flags are added to some packets to minimise the number of unnecessary transmissions. The optimal message packet length is calculated.

- A brief study is made of error detection and forward error correction techniques.

- The building of the OMNeT++ simulation of the system and protocol is described in detail. Attention is given to how physical constraints were implemented in software.

- The `vcs_added_backoff` parameter is singled out as being critical for the optimisation of system throughput. Various random number distributions are considered for this parameter, including a new Boolean-type distribution. MATLAB scripts are presented to aid in the optimisation of system throughput. Thorough optimisation is done of the distributions considered, and the results are presented. The Boolean distribution is found to give the best results.

- In Chapter 8 the OMNeT++ simulation is used to quantify the throughput of the system, and to determine the effectiveness of the optimisations done. The results give confirmation of the validity of the optimisation, and of the reliability of the OMNeT++ simulation. A summary of further results can be found at the end of that chapter.

9.4 Contributions

The following are the contributions of this thesis:

- A CSMA/CA-type protocol specifically adapted for LEO satellite applications, defined in flow-chart format
- A flag system for avoiding unnecessary packet transmissions, and thus conserving satellite battery power and maximising throughput
- An extensive working simulation of the above protocol, which allows the user to determine expected behaviour of the system and to estimate throughput and other statistical characteristics
- MATLAB scripts which automatically set up runs of the above simulation for a given range of values of certain parameters, which are used for studying results and making further refinements to the protocol
- A Boolean-type random variable distribution for the back-off strategy
- A comparison of various random variable distributions for the back-off period, and a collection of MATLAB scripts to effect this.
- Increased faith in the accuracy of OMNeT++ for simulating communications systems and protocols - this is a valuable contribution to others in the field who require the use of simulation tools

9.5 Suggestions for Future Work

The line of investigation taken in this thesis is by no means exhausted, and the following are suggested as areas for possible further work:

- Much work was put into the OMNeT++ simulation and it is now about 99% bug free. It can now be used to analyse other system parameters, the most important of which is probably system latency, especially under high traffic demands. All the tools are in place, and it will take little effort to expand the code to allow one to begin analysing this parameter.
- The protocol defined in this thesis has, through simulation, been proven as a viable solution for the given system. The next step is to implement the system

in hardware to practically test the protocol. There are plans to follow *Sumbandila* up with further micro-satellites, and these would provide the perfect testbed.

- Further micro-satellites add another dimension to the system - the possibility of satellite constellations. This allows the relaying of messages from one satellite to another before sending it back to ground. A new protocol, or adjustment of the present one, would have to be used in this case.

9.6 Final Comment

With the increased scope of uses for telecommunications, and the resulting increasing demand for bandwidth in a finite usable spectrum, the value of computer simulations for modelling telecommunications systems can not be over-emphasised. This thesis makes two valuable contributions in this regard - to grow confidence in OMNeT++ as a simulation environment, and to provide a simulation for a specific implementation and adaptation of the CSMA/CA protocol.

Bibliography

[Black-93] Black, U.D.
Data Communications and Distributed Networks
Prentice Hall, 1993

[Boot-96] Boot, J.
Implementation and Integration of Ground and Space Segment Communication Software for the Sunsat Micro-satellite
University of Stellenbosch, M-thesis, December 1996

[Campb-96] Campbell, B.A. and McCandless, S.W. Jr.
Introduction to Space Sciences and Spacecraft Applications
Gulf Publishing Company, 1996

[Cardo-01] Cardoza, A.R.
Middleware for the SUNSAT Field Station
University of Stellenbosch, M-thesis, December 2001

[Erasm-00] Erasmus, J.M.
Design of a Mobile Field Station for LEO Satellite Communications
University of Stellenbosch, M-thesis, March 2000

[Hess-93] Hess, G.C.
Land-Mobile Radio System Engineering
Artech House, Inc., 1993

[Klein-75a] Kleinrock, L. and Tobagi, F.A.
Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-access Modes and Their Throughput-delay Characteristics
IEEE Transactions on Communications, Vol. COM-23, No. 12, December 1975

[Klein-75b] Kleinrock, L. and Tobagi, F.A.
Packet Switching in a Multi-access Broadcast Channel: Performance Evaluation
IEEE Transactions on Communications, Vol. COM-23, No. 4, April 1975

[Klein-75c] Kleinrock, L. and Tobagi, F.A.
Random Access Techniques for Data Transmission over Packet-switched Radio Channels
Proceedings, National Computer Conference, 1975

[Labus-06] Labuschagne, A.S.
The Design of a Telemetry System for Grumeti Reserves
University of Stellenbosch, M-thesis, April 2006

[Lam-75] Lam, S.S. and Kleinrock, L.
Packet Switching in a Multi-access Broadcast Channel: Dynamic Control Procedures
IEEE Transactions on Communications, Vol. COM-23, September 1975

[Maral-99] Maral, G. and Bousquet, M.
Satellite Communications Systems
John Wiley and Sons, Ltd, November 1999

[Roux-98] Roux, J.J.
The Design of a VHF and UHF Receiver Front-end for the Sunsat Microsatellite
University of Stellenbosch, M-thesis, January 1998

[Tobag-80] Tobagi, F.A.
Multi-access Protocols in Packet Communication Systems
IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980

[Wolhu-03] Wolhuter, R. and van Rooyen, G.J.
Elements of Telecommunications Systems Design and Teletraffic Analysis
Stellenbosch, 2003

[Varga-06] Varga, A. and Hornig, R.
OMNeT++ Community Site [Online]
Available: www.omnetpp.org, 13 September 2006

Appendix A

CSMA/CA flow diagrams

CSMA/CA: Notes

26 Sep 2006 v5.2

Ground Station:

ReRTS_counter:	increased:	when CTS_timeout occurs
	reset:	when CTS is received
		when message cancelled
ReTX_counter:	increased:	each time message is sent
	reset:	when ACK is received
		when message cancelled
Access_counter:	increased:	each time channel is sensed busy
	reset:	when channel is sensed available
		when message cancelled

Abbreviations

ACK	=	Acknowledge
CTS	=	Clear to Send
NAK	=	Not Acknowledge
RTS	=	Request to Send
VCS	=	Virtual Carrier Sense

Satellite:

ReRTS_counter:	increased:	when CTS timeout occurs
	reset:	when CTS is received
		when message cancelled
ReTX_counter:	increased:	when CTS is received and ACK_timer is running(set)
	reset:	when ACK is received
		when message cancelled
CTS(0):	This is a CTS with DataLeft set to 0 bytes, and sent to a reserved address.	
	Stations will thus go into VCS-backoff for 0s, before they send RTS's.	
	Used to "wake-up" stations, or alert them to satellite's presence.	

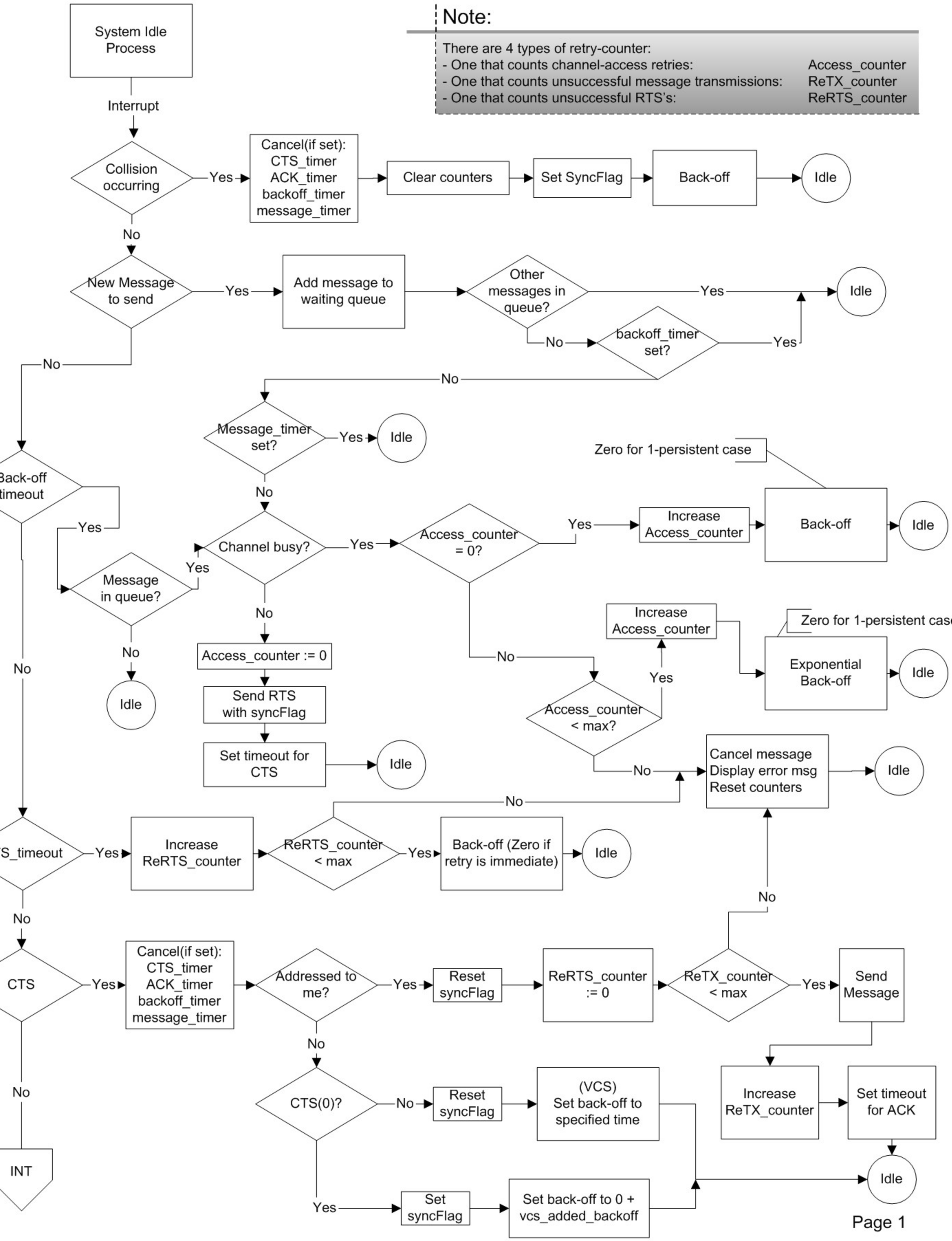
CSMA/CA: GND Station

26 Sep 2006 v5.2

This document is an improvement upon v5.1, and corrects an error in the previous version.

Note:

There are 4 types of retry-counter:
- One that counts channel-access retries: Access_counter
- One that counts unsuccessful message transmissions: ReTX_counter
- One that counts unsuccessful RTS's: ReRTS_counter



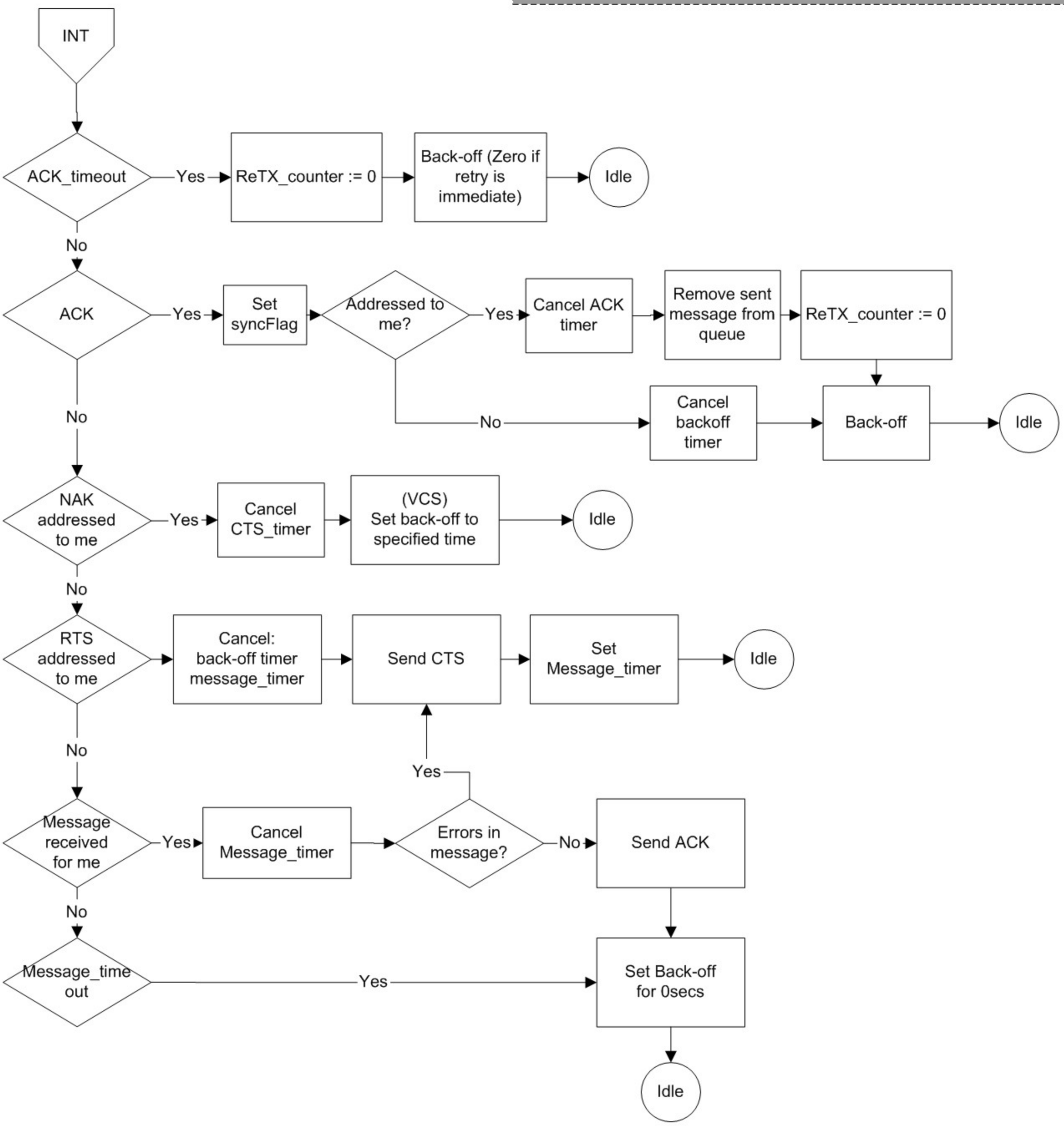
CSMA/CA: GND Station cont...

26 Sep 2006 v5.2

Note:

A NAK is only issued when 2 stations “simultaneously” request transmission.

When a message is received with errors a CTS is immediately sent to that station to repeat the message.



CSMA/CA: Satellite

26 Sep 2006 v5.2

Note: This can be determined by observing the length of the collisions.

Note:

If message has an error then a CTS is immediately sent to ask the station to repeat it's message.

System Idle Process

Interrupt

Collision occurring

Yes

Cancel (if set): CTS_timer ACK_timer

Clear counters

Increase syncFlag

Send "collision occurring" with syncFlag

Wait until collision complete

only RTS collisions and msg_timer set

Yes

Idle

Send CTS(0) with syncFlag

No

Messages in send queue?

No

Cancel (if set) msg_timer

No

RTS

Yes

ACK- or CTS- or message-timer set

Yes

Send NAK with data left

Idle

No

flag=0 or flag=syncFlag

Yes

Send CTS

Set message_ timeout

No

Idle

Messages in send queue?

No

Yes

Message received

Yes

Cancel message_timeout

Yes

Errors in message?

No

destination = satellite?

Yes

Send ACK with syncFlag

No

Add to send queue

No

New message to send

Yes

Add message to send queue

Other messages in queue?

Yes

Idle

No

message_time out

Yes

Messages in send queue?

No

Idle

Yes

No

message_timer set?

Yes

No

Yes

No

CTS_timeout?

Yes

Send RTS

Set timeout for CTS

Idle

Yes

Increase ReRTS_counter

ReRTS_counter < max?

No

Cancel message Display error msg Reset counters

Idle

No

CTS

Yes

Messages in send queue?

Yes

ACK_timer set?

No

Cancel CTS_timer

ReRTS_counter := 0

ReTX_counter < max

Yes

Send Message

Set timeout for ACK

Idle

Yes

Cancel ACK_timer

Increase ReTX_counter

No

ACK_timeout

Yes

Send RTS

Set timeout for CTS

Idle

No

ACK

Yes

Cancel ACK timer

Remove sent message from queue

ReTX_counter := 0

Idle

Appendix B

CMX589A Data sheet

This appendix presents properties of the GMSK modem discussed in Section 2.6.

Consumer Microcircuits Limited

M-I-X-E-D S-I-G-N-A-L S-O-L-U-T-I-O-N-S

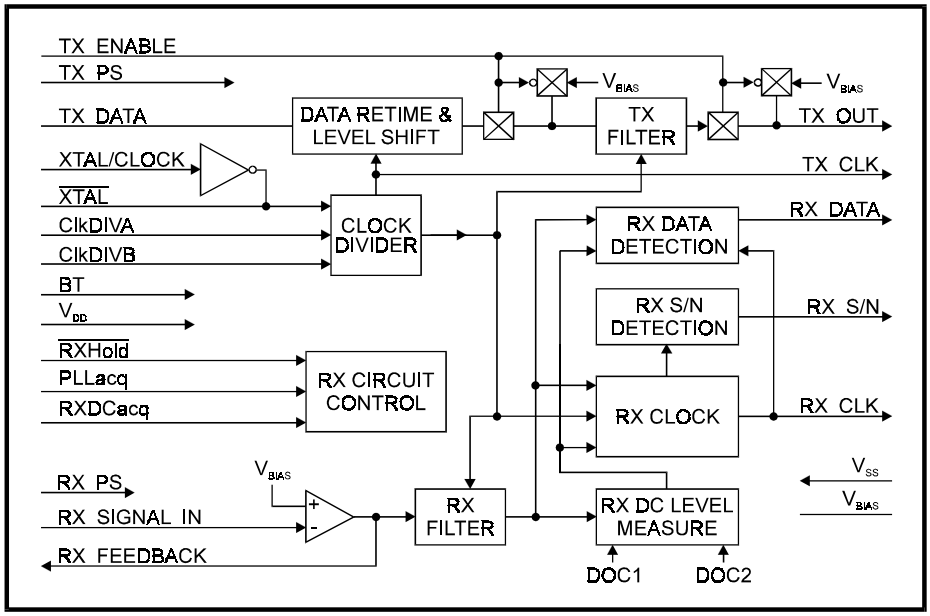
CMX589A

GMSK Modem

D/589A/3 September 1998 Provisional Information

Features and Applications

- **Data Rates from 4kbps to 200kbps**
- **Full or Half Duplex Gaussian Filter and Data Recovery for Minimum Shift Keying (GMSK) Designs**
- **Selectable BT: (0.3 or 0.5)**
- **Low Power**
  - **3.0V, 20kbps, 1.5mA typ.**
  - **5.0V, 64kbps, 4.0mA typ.**
- **Low Current Non-DSP Solution**
- **Small TSSOP Packs fit PCMCIA/PC Cards**
- **Portable Wireless Data Applications**
  - **Cellular Digital Packet Data (CDPD)**
  - **Mobitex™ Mobile Data System**
- **Spread Spectrum Data Links**
- **GPS/Differential GPS Wireless Links**
- **Point-of-Sale Terminals**
- **Low Power Wireless Data Link for PCs, Laptops, and Printers**



Brief Description

The CMX589A is a single-chip synchronous data pump/modem designed for Wireless Data Applications. Employing Gaussian filtering for Minimum shift Keying (GMSK) baseband modulation applications, the CMX589A features a wide range of available data rates from 4k to 200k bps. Data Rates and the choice of BT (0.3 or 0.5) are pin programmable to provide for different system requirements.

The Tx and Rx digital data interfaces are bit serial, synchronized to generated Tx and Rx data clocks. Separate Tx and Rx Powersave inputs allow full or half-duplex operation. Rx input levels can be set by suitable AC and DC level adjusting circuitry built with external components around an on-chip Rx Input Amplifier.

Acquisition, Lock, and Hold of Rx data signals are made easier and faster by the use of Rx Control Inputs to clamp, detect, and /or hold input data levels and can be set by the µProcessor as required. The Rx S/N output provides an indication of the quality of the received signal.

The CMX589A may be used with a 3.0V to 5.5V power supply and is available in the following packages: 24-pin TSSOP (CMX589AE2), 24-pin SSOP (CMX589AD5), 24-pin SOIC (CMX589AD2), and 24-pin PDIP (CMX589AP4).

Contents

Section **Page**

Block Diagram

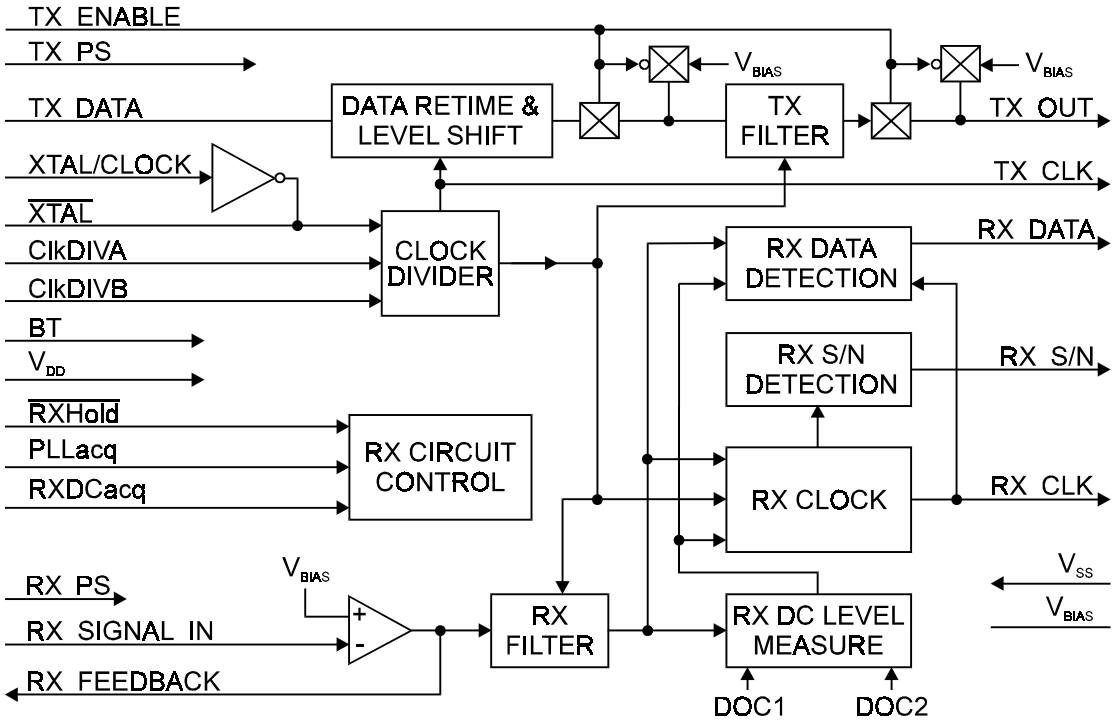


Figure 1: Block Diagram

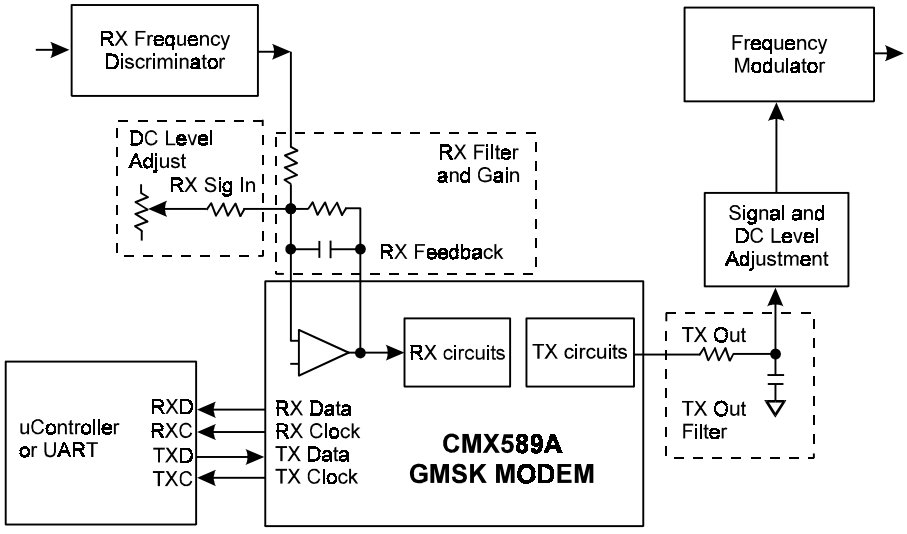


Figure 2: System Block Diagram

2 Signal List

Pin No. E2/D5/ D2/P4	Signal	Type	Description
1	XTALN	O/P	The output of the on-chip clock oscillator.
2	XTAL/CLOCK	I/P	The input to the on-chip Xtal oscillator. A Xtal, or externally derived clock (f_{XTAL}) pulse input should be connected here. If an externally generated clock is to be used, it should be connected to this pin and the XTALN pin left unconnected. Note: Operation without a suitable Xtal or clock input may cause device damage.
3	ClkDivA	I/P	Logic level inputs control the internal clock divider and therefore the transmit and receive data rate. See Table 4.
4	ClkDivB	I/P	Logic level inputs control the internal clock divider and therefore the transmit and receive data rate. See Table 4.
5	RxHOLDN	I/P	A logic ‘0’ applied to this input will freeze the Clock Extraction and Level Measurement circuits unless they are in ‘Acquire’ mode.
6	RxDCacq	I/P	A logic ‘1’ applied to this input will set the Rx Level Measurement circuitry to the ‘Acquire’ mode. See Table 6.
7	PLLacq	I/P	A logic ‘1’ applied to this input will set the Rx Clock Extraction circuitry to the ‘Acquire’ mode. See Table 5.
8	Rx PSAVE	I/P	A logic ‘1’ applied to this input will powersave all receive circuits except for Rx CLK output (which will continue at the set bit-rate) and cause the Rx Data and Rx S/N outputs to go to a logic ‘0’.
9	V_{BIAS}		The internal circuitry bias line, held at $V_{DD}/2$. This pin must be bypassed to V_{SS} by a capacitor mounted close to the pin.
10	Rx FB	O/P	Output of the Rx Input Amplifier.
11	Rx Signal In	I/P	Input to Rx input amplifier.
12	V_{SS}	power	Negative supply (GND).
13	DOC1		Connections to the Rx Level Measurement Circuitry. A capacitor should be connected from this pin to V_{SS}.
14	DOC2		Connections to the Rx Level Measurement Circuitry. A capacitor should be connected from this pin to V_{SS}.
15	BT		A logic level to select the modem BT (the ratio of the Tx Filter's -3dB frequency to the Bit-Rate). A logic ‘1’ = BT of 0.5 and a logic ‘0’ = BT of 0.3.

Pin No. E2/D5/ D2/P4	Signal	Type	Description
16	Tx Out	I/P	Gaussian filtered Tx output signal. In powersave mode the Tx Out pin is high impedance.
17	Tx Enable	I/P	A logic '1' applied to this input, enables the transmit data path, through the Tx Filter to the Tx Out pin. A logic '0' will place the Tx Out pin to V_{BIAS} via a high impedance.
18	Tx PSAVE	I/P	A logic '1' applied to this input will powersave all transmit circuits except for the Tx Clock.
19	Tx Data	I/P	The logic level input for the data to be transmitted. This data should be synchronous with Tx CLK.
20	Rx Data	I/P	A logic level output carrying the received data, synchronous with Rx CLK.
21	Rx CLK	I/P	A logic level clock output at the received data bit-rate.
22	Tx CLK	I/P	A logic level clock output at the transmit data bit-rate.
23	Rx S/N	O/P	A logic level output which may be used as an indication of the quality of the received signal.
24	V_{DD}	power	Positive supply. Levels and voltages within the device are dependent upon this supply. This pin should be bypassed to V_{SS} by a capacitor mounted close to the pin.

Table 1: Signal List

3 External Components

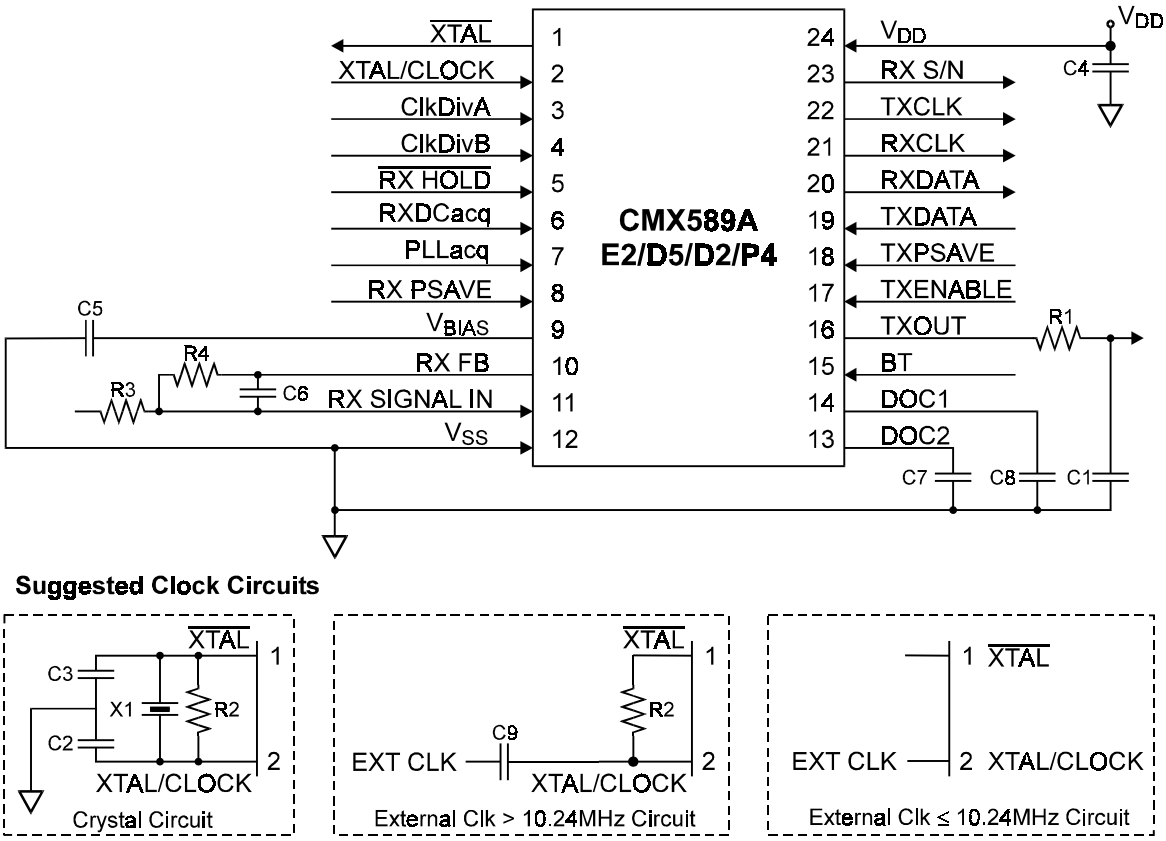


Figure 3: Recommended External Components

Component	Notes	Value	Tolerance
R1	1		±5%
R2		1.0MΩ	±10%
R3	2		±10%
R4	3		±10%
C1	1		±10%
C2	4		
C3	4		

Component	Notes	Value	Tolerance
C4		0.1µF	±20%
C5		1.0µF	±20%
C6	5		±20%
C7	6		
C8	6		
X1	8		

Table 2: Recommended External Components

Recommended External Component Notes:

1. The RC network formed by R1 and C1 is required between the Tx Out pin and the input to the modulator. This network, which can form part of any DC level shifting and gain adjustment circuitry, forms an important part of the transmit signal filtering. The ground connection to the capacitor C1 should be positioned to give maximum attenuation of high-frequency noise into the modulator.
   The component values should be chosen so that the product of the resistance and the capacitance is:

   For a BT of 0.3 R1C1 = 0.34/bit rate (bps)
   For a BT of 0.5 R1C1 = 0.22/bit rate (bps)

Data Rates (kbps)	BT = 0.3		BT = 0.5	
	R1	C1	R1	C1
4	120kΩ	680pF	120kΩ	470pF
4.8	100kΩ	680pF	100kΩ	470pF
8	91kΩ	470pF	120kΩ	220pF
9.6	91kΩ	390pF	47kΩ	470pF
16	47kΩ	470pF	91kΩ	150pF
19.2	100kΩ	180pF	91kΩ	120pF
32	47kΩ	220pF	47kΩ	150pF
38.4 *	47kΩ	180pF	47kΩ	120pF
64 *	56kΩ	100pF	51kΩ	68pF
80 *			39kΩ	68pF
128 *			82kΩ	22pF
144 *			68kΩ	22pF
160 *			62kΩ	22pF
176 *			56kΩ	22pF
192 *			51kΩ	22pF
* $V_{DD} \geq 4.5V$, external clock				

Table 3: Data Rate vs. BT and Selected External Component Values

Note: In all cases, the value of R1 should not be less than 20.0kΩ, and that the calculated value of C1 includes calculated parasitic capacitance.

2. R3, R4 and C6 form the gain components for the Rx Input signal. R3 should be chosen as required by the signal input level.
3. For bit rate ≤ 64kbps, R4 = 100kΩ. For bit rate > 64kbps, R4 = 10kΩ.
4. The values chosen for C2 and C3 (including stray capacitance), should be suitable for the applied V_{DD} and the frequency of X1.
   As a guide: C2 = C3 = 33pF at 1.0MHz falling to 18pF at the maximum frequency.
   At 3.0V, C2 = C3 = 33pF falling to 18pF at 5.0MHz the equivalent series resistance of X1 should be less than 2.0KΩ falling to 150Ω at the maximum frequency. Stray capacitance on the Xtal/Clock circuit pins must be minimized.
5. For bit rate ≤ 64kbps, C6 = 22pF. For bit rate > 64kbps, C6 = $\frac{1}{3 \times \text{bit rate} \times 2\pi \times 10\text{k}\Omega}$
   e.g. for 128kbps, C6 = 41.1pF.
6. C7 and C8 should both be .015µF for a data rate of 8kbps, and inversely proportional to the data rate for other data rates, e.g. 0.030µF at 4kbps, 1800pF at 64kbps, 680pF at 192kbps.
7. The tolerance of C9 is not very critical because it primarily serves as a dc blocking capacitor.
8. The CMX589A can operate correctly with the Xtal frequencies between 1.0MHz and 16.0MHz (V_{DD} = 5.0V) and 1.0MHz to 5.0MHz (V_{DD} = 3.0V). External clock frequencies up to 25.6MHz ($V_{DD} \geq 4.5V$) are also supported (See Table 4 for examples.) For best results, a crystal oscillator design should drive the clock inverter input with signal levels of at least 40% of V_{DD}, peak to peak. Tuning fork crystals generally cannot meet this requirement. To obtain crystal oscillator design assistance, consult your crystal manufacturer. Operation of this device without a Xtal or Clock input may cause device damage.

4 General Description

4.1 Clock Oscillator Divider

The Tx and (nominal) Rx data rates are determined by division of the frequency present at the XTALN pin as generated by the on-chip Xtal oscillator, with external components, or supplied from an external source.

The division ratio is controlled by the logic level inputs on ClkDivA and ClkDivB pins as shown in Table 4, together with an indication of how various standard data rates may be derived from common µP Xtal frequencies.

$$\text{Data Rate} = \frac{\text{Xtal/Clk Frequency}}{\text{Division Ratio (ClkDiv A/B)}}$$

			Xtal/Clock Frequency (MHz)					
			24.576*	8.192	4.9152	4.096	2.4576	2.048
Inputs						12.288/3	12.288/5	6.144/3
ClkDivA	ClkDivB	Xtal/Clk Freq / Data Rate	Data Rate (kbps)					
0	0	128	192*	64*	38.4*	32	19.2	16
0	1	256	96*	32	19.2	16	9.6	8
1	0	512	48*	16	9.6	8	4.8	4
1	1	1024	24*	8	4.8	4		

* $V_{DD} \geq 4.5V$, external clock

Table 4: Example Clock/Data Rates

Note: The device operation is not guaranteed above 200kbps or below 4kbps at the relevant supply voltage.

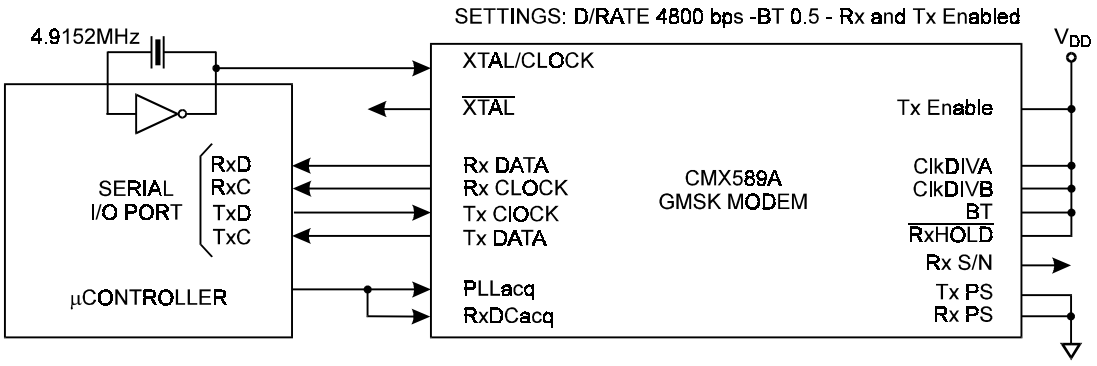


Figure 4: Minimum µController System Connections

4.2 Receive

4.2.1 Rx Signal Path Description

The function of the Rx circuitry is to:

1. Set the incoming signal to a usable level.
2. Clean the signal by filtering.
3. Provide dc level thresholds for clock and data extraction.
4. Provide clock timing information for data extraction and external circuits.
5. Provide Rx data in a binary form.
6. Assess signal quality and provide Signal-to-Noise information.

The output of the radio receiver's Frequency Discriminator should be fed to the CMX589A's Rx Filter by a suitable gain and DC level adjusting circuit. This circuit can be built with external components around the on-chip Rx Input Amplifier. The gain should be set so that the signal level at the Rx Feedback pin is nominally 1V peak to peak (for V_{DD}=5.0V) centered around V_{BIAS} when receiving a continuous 1111000011110000.. data pattern.

Positive going signal excursions at Rx Feedback pin will produce a logic '0' at the Rx Data Output. Negative going excursions will produce a logic '1'.

The received signal is fed through the lowpass Rx Filter, which has a -3dB corner frequency of 0.56 times the data bit-rate, before being applied to the Level Measure and Clock and Data extraction blocks.

The Level Measuring block consists of two voltage detectors, one of which measures the amplitude of the positive parts of the received signal. The other measures the amplitude of the negative portions. (Positive refers to signal levels higher than V_{DD}/2, and negative to levels lower than V_{DD}/2.) External capacitors are used by these detectors, via the Doc1 and Doc2 pins, to form voltage 'hold' or 'integrator' circuits. These two levels are then used to establish the optimum DC level decision-thresholds for the Clock and Data extraction, depending upon the Rx signal amplitude and any DC offset.

4.2.2 Rx Circuit Control Modes

The operating characteristics of the Rx Level Measurement and Clock Extraction circuits are controlled, as shown in Table 6, by logic level inputs applied to the PLLacq, Rx HOLDN and RxDCacq pins to suit a particular application, or to cope with changing reception conditions, reference Figure 5.

In general, a data transmission will begin with a preamble, for example, 1100110011001100, to allow the receive modem to establish timing and level-lock as quickly as possible. After the Rx carrier has been detected, and during the time that the preamble is expected, the RxDCacq and PLLacq Inputs should be switched from a logic '0' to a logic '1' so that the Level Measuring and Clock Extraction modes are operated and sequenced as shown.

The Rx HOLDN input should normally be held at a logic '1' while data is being received, but may be driven to a logic '0' to freeze the Level Measuring Clock Extraction circuits during a fade. If a fade lasts for less than 200 bit periods, normal operation can be resumed by returning the Rx HOLDN input to a logic '1' at the end of the fade. For longer fades, it may be better to reset the Level Measuring circuits by placing the RxDCacq to a logic '1' for 10 to 20 bit periods.

Rx HOLDN has no effect on the Level Measuring circuits while RxDCacq is at a logic '1', and has no effect on the PLL while PLLacq is at a logic '1'.

A logic '0' on Rx HOLDN does not disable the Rx Clock output, and the Rx Data Extraction and S/N Detector circuits will continue to operate.

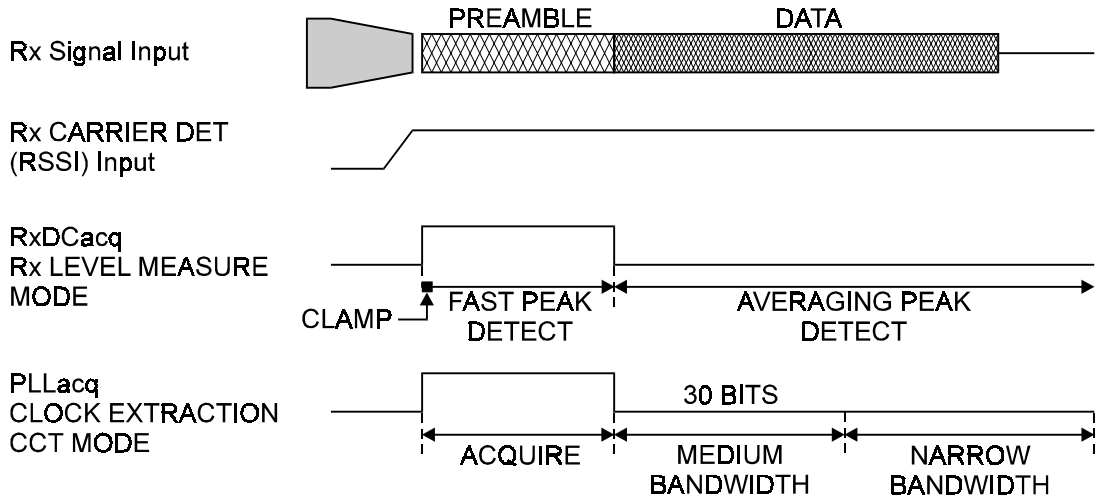


Figure 5: Rx Mode Control Diagram

PLLacq	Rx HOLDN	PLL Action	
1	1	Acquire	Sets the PLL bandwidth wide enough to allow a lock to the received signal in less than 8 zero crossings. This mode will operate as long as PLLacq is a logic “1”.
1 to 0	1	Medium Bandwidth	The correction applied to the extracted clock is limited to a maximum of ±1/16th bit-period for every two received zero-crossings. The PLL operates in this mode for a period of about 30 bits immediately following a 1 to 0 transition of the PLLacq input, provided that the Rx HOLDN input is a logic ‘1’.
0	1	Narrow Bandwidth	The correction applied to the extracted clock is limited to a maximum of ±1/64th bit-period for every two received zero-crossings. The PLL operates in this mode whenever the Rx HOLDN Input is a logic ‘1’ and PLLacq has been a logic ‘0’ for at least 30 bit periods (after Medium Bandwidth operation for instance).
0	0	Hold	The PLL feedback loop is broken, allowing the Rx Clock to freewheel during signal fade periods.

Table 5: PLL Action Measurement Operational Modes

RxDCacq	Rx HOLDN	Rx Level Measure Action	
0 to 1	X	Clamp	Operates for one bit-time after a 0 to 1 transition of the RXDCacq input. The external capacitors are rapidly charged towards a voltage mid-way between the received signal input level and V_{BIAS}, with the charge time-constant being of the order of 0.5 bit-time.
1	X	Fast Peak Detect	The voltage detectors act as peak-detectors, one capacitor is used to capture the positive-going signal peaks of the Rx Filter output signal and the other capturing the negative-going peaks. The detectors operate in this mode whenever the RXDCacq input is at a logic ‘1’, except for the initial 1-bit Clamp-mode time.
0	1	Averaging Peak Detect	Provides a slower but more accurate measurement of the signal peak amplitudes.
0	0	Hold	The capacitor charging circuits are disabled so that the outputs of the voltage detectors remain substantially at the last readings (discharging very slowly [time-constant approx. 2,000 bits] towards V_{BIAS}).

X = Do not care

Table 6: Rx Level Measurement Operational Modes

4.2.3 Rx Clock Extraction

Synchronized by a PLL circuit to zero-crossings of the incoming data, the Rx Clock Extraction circuitry controls the Rx Clock output. The Rx Clock is also used internally by the Data Extraction circuitry. The PLL parameters can be varied by the Rx Circuit Control inputs PLLacq and Rx HOLDN to operate in one of four PLL modes as described in Table 5 and Table 6.

4.2.4 Rx Data Extraction

The Rx Data Extraction circuit decides whether each received bit is a 1 or 0 by sampling the received signal, after filtering, and comparing the sample values to an adaptive threshold derived from the Level Measuring circuit. This threshold is adapted from bit to bit to compensate for intersymbol interference caused by the and limiting of the overall transmission path and the Gaussian premodulation filter. Extracted data is output from the Rx Data pin, and should be sampled externally on the rising edge of the Rx CLK.

4.2.5 Rx S/N Detection

The Rx S/N Detector system classifies the incoming zero-crossings as GOOD or BAD depending upon the time when each crossing actually occurs with respect to its expected time as determined by the Clock Extraction PLL. This information is then processed to provide a logic level output at the Rx S/N pin. A high level indicates a series of GOOD crossings; a low level indicates a BAD crossing.

By averaging this output, it is possible to derive a measure of the Signal-to-Noise-Ratio and hence the Bit-Error-Rate of the received signal.

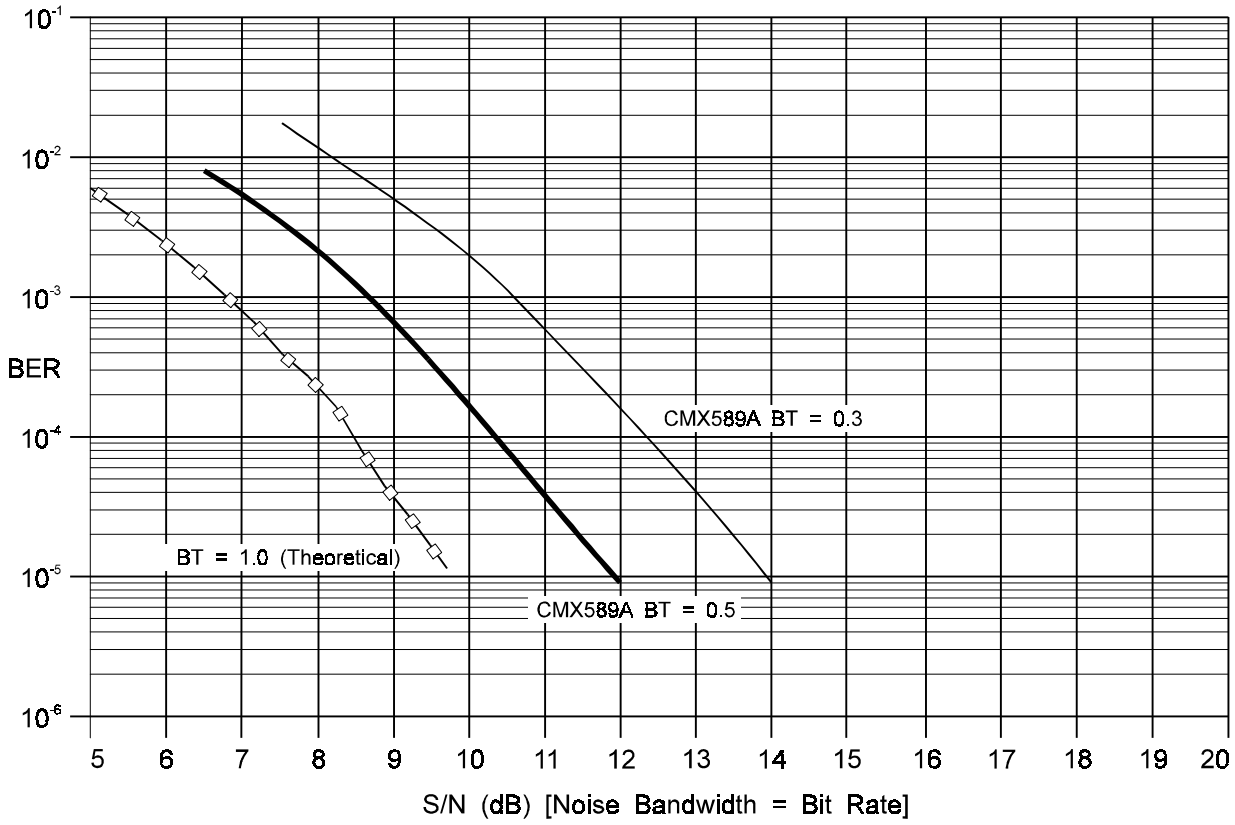


Figure 6: Typical Bit-Error-Rate Performance at V_{DD} = 5.0V

Note: Figure 6 indicates typical performance, independent of bit rate (although the applied noise bandwidth is considered to match the bit rate used), radio performance (e.g. IF filter distortion), supply voltage (higher bit rates require $V_{DD} \geq 4.5V$), and other 'real world' factors.

4.2.6 Rx Signal Quality

The effect of input Rx Signal quality on the Rx S/N output is shown in Figure 7.

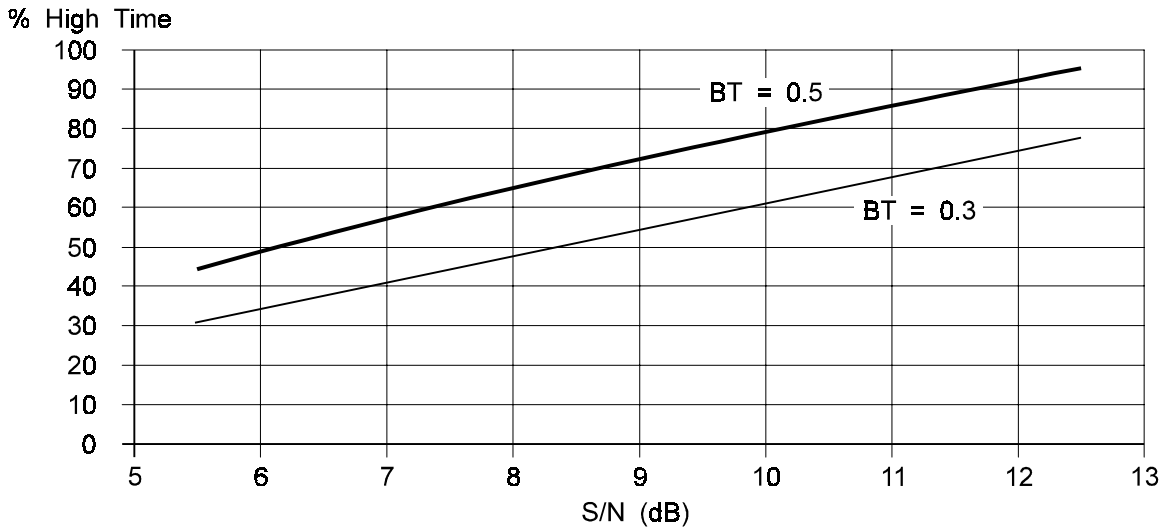


Figure 7: Typical Rx S/N Output High Time (%) vs. Input S/N

4.3 Transmit

4.3.1 Tx Signal Path Description

The binary data applied to the Tx Data input is retimed within the chip on each rising edge of the Tx Clock and then converted to a 1-volt peak-to-peak binary signal centred at V_{BIAS} (for V_{DD}= 5.0V)

If the Tx Enable input is high, then this internal binary signal will be connected to the input of the Lowpass Tx Filter, and the output of the filter connected to the Tx Out pin.

Tx Enable	Tx Filter Input	Tx Out Pin
1	Data @ $\frac{V_{DD}}{5}$ V_{P-P} e.g. 1V_{P-P} for V_{DD}=5V	Filtered 'Tx Filter Input'
0	V_{BIAS}	V_{BIAS} via 500kΩ

A 'low' input to the Tx Enable will connect the input of the Tx Filter to V_{BIAS}, and disconnect the Tx Out pin from the filter, connecting it instead to V_{BIAS} through a high resistance (nominally 500kΩ).

The Tx Filter has a lowpass frequency response, which is approximately gaussian in shape as shown in Figure 9, to minimize amplitude and phase distortion of the binary signal while providing sufficient attenuation of the high frequency-components which would otherwise cause interference into adjacent radio channels. The actual filter bandwidth to be used in any particular application will be determined by the overall system requirements. The attenuation-vs.-frequency response of the transmit filtering provided by the CMX589A has been designed to meet the specifications for most GMSK modem systems that are -3dB bandwidth switchable between 0.3 and 0.5 times the data bit-rate (BT).

Note: An external RC network is required between the Tx Out pin and the input to the Frequency Modulator (see Figure 2 and Figure 3). This network, which can form part of any DC level shifting and gain adjustment circuitry, forms an important part of the transmit signal filtering. The ground connection to capacitor C1 should be positioned to give maximum attenuation of high-frequency noise into the modulator.

The signal at Tx Out is centered around V_{BIAS}, going positive for logic '1' (high) level inputs to the Tx Data input and negative for logic '0' (low) inputs.

When the transmit circuits are put into a powersave mode (by a logic '1' to the Tx PS pin) the output voltage of the Tx Filter will go to high impedance. When power is subsequently restored to the Tx filter, its output will take several bit-times to settle. The Tx Enable input can be used to prevent these abnormal voltages from appearing at the Tx Out pin.

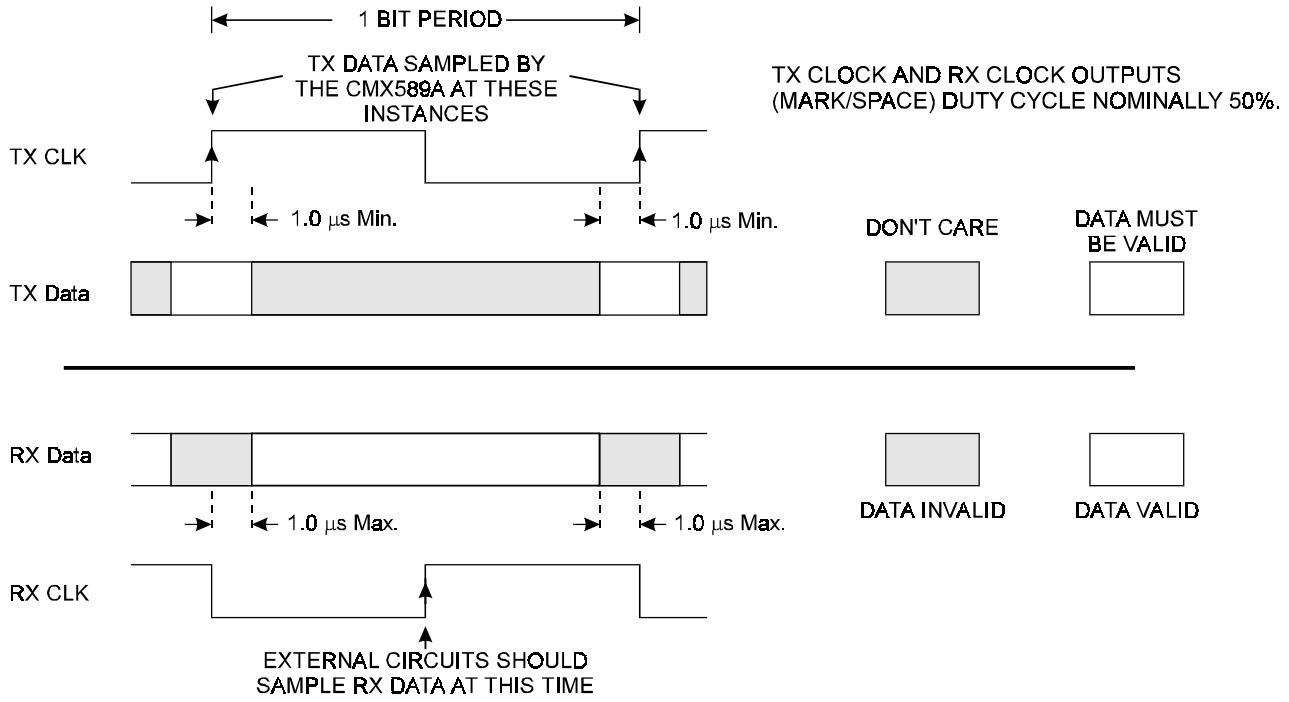


Figure 8: Rx and Tx Clock Data Timings

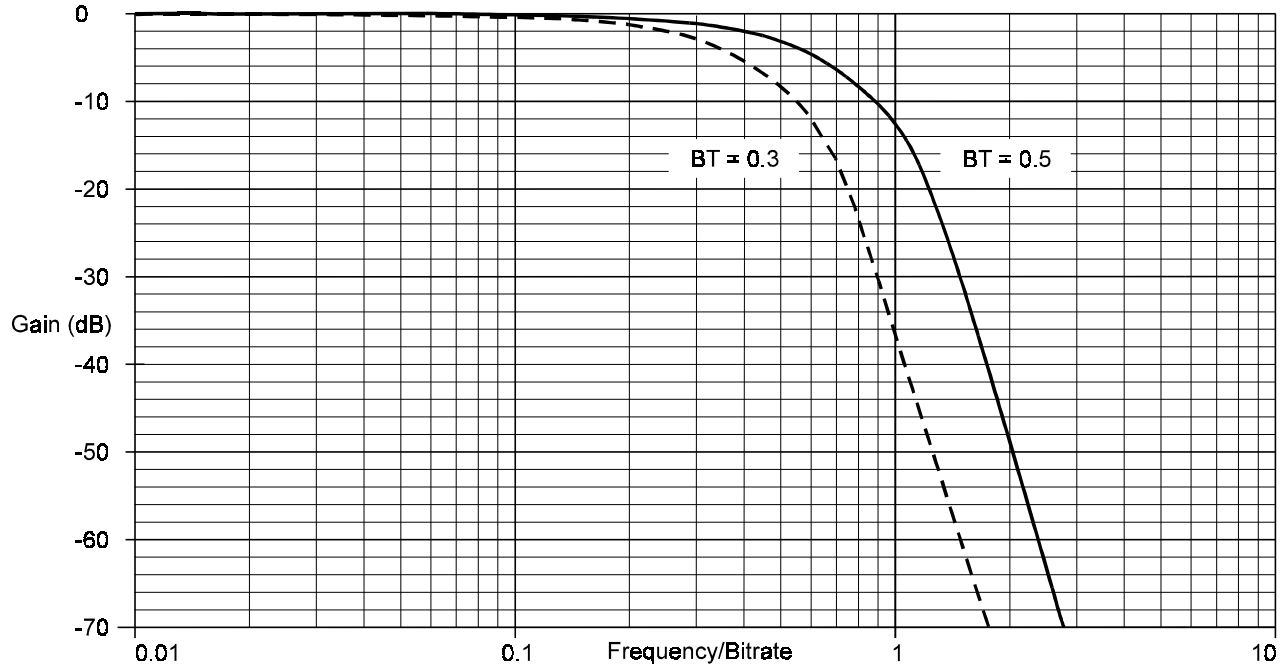


Figure 9: Tx Filter Response

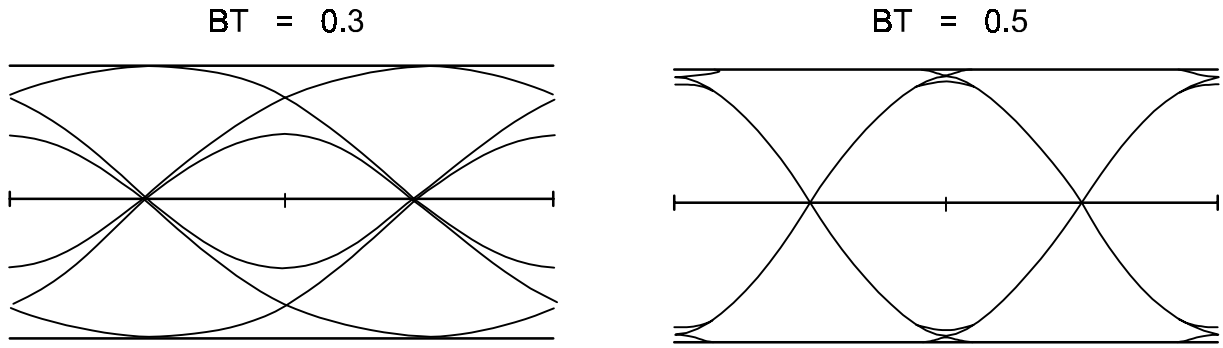


Figure 10: Typical Transmit Eye Patterns

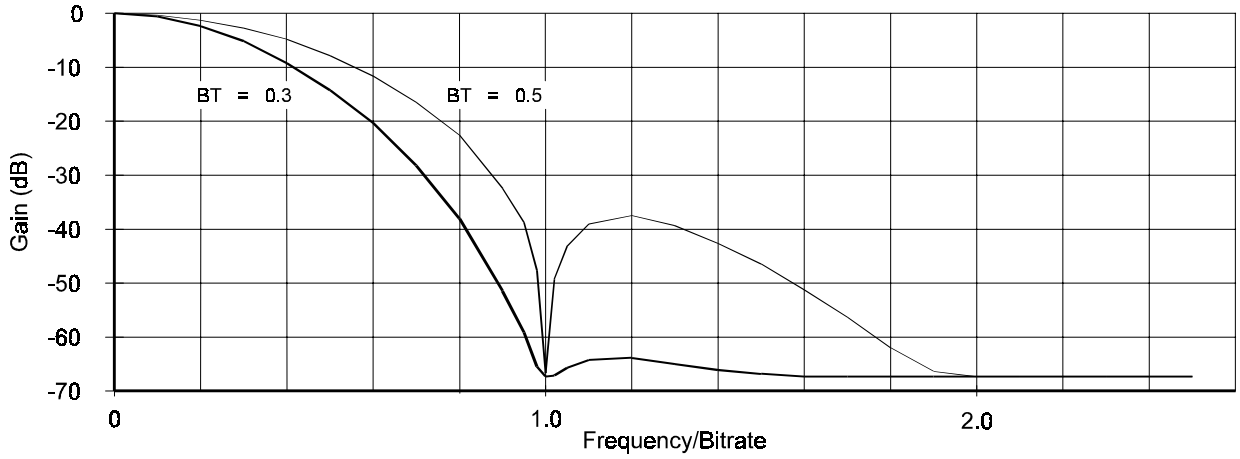


Figure 11: Tx Output Spectrum (Random Data)

4.4 Data Formats

The receive section of the CMX589A works best with data which has a reasonably random structure --the data should contain approximately the same number of 'ones' as 'zeroes' with no long sequences (>100 bits) of consecutive ones or zeroes. Also, long sequences (>100 bits) of 10101010 ... patterns should be avoided.

For this reason, it is recommended that data be made random in some manner before transmission, for example by exclusive-ORing it with the output of a binary pseudo-random pattern generator.

Where data is transmitted in bursts, each burst should be preceded by a preamble designed to allow the receive modem to establish timing and level lock as quickly as possible. This preamble for BT=0.3 should be at least 16 bits long, and should preferably consist of alternating pairs of ones and zeros i.e. 110011001100....; the eye of pattern 10101010 .... has the most gradual slope and will yield poor peak levels for the Rx circuits. For BT=0.5 the eye pattern of 10101010... has reduced intersymbol interference and may be used as the preamble (DC Acq pin should be held high during preamble). See Figure 5.

4.5 Acquisition and Hold Modes

The RXDCacq and PLLacq inputs must be pulsed High for about 16 bits at the start of reception to ensure that the DC measurement and timing extraction circuits lock-on to the received signal correctly. Once lock has been achieved, the above inputs should be taken Low again.

In most applications, there will be a DC step in the output voltage from the receiver FM discriminator due to carrier frequency offsets as channels are changed or when the remote transmitter is turned on.

The CMX589A can tolerate DC offsets in the received signal of at least ±10% of V_{DD} with respect to V_{BIAS}, (measured at the Rx Feedback pin). However, to ensure that the DC offset compensation circuit operates correctly and with minimum delay, the Low to High transition of the RXDCacq and PLLacq inputs should occur after the mean input voltage to the CMX589A has settled to within about 0.1V of its final value.

Note: This can place restrictions on the value of any series signal coupling capacitor.

As well as using the Rx Hold input to freeze the Level Measuring and Clock Extraction circuits during a signal fade, it may also be used in systems which use a continuously transmitting control channel to freeze the Rx circuitry during transmission of a data packet, allowing reception to resume afterwards without losing bit synchronization. To achieve this, the CMX589A Xtal clock needs to be accurate enough that the derived RxClock output does not drift by more than about 0.1 bit time from the actual received data-rate during the time that the RxHold input is 'Low'.

However; the RXDCacq input may need to be pulsed High for 2 bit durations to re-establish the level measurements if the RxHold input is Low for more that a few hundred bit-times (exact number depends on system crystal tolerances).

The voltages on the Doc1 and Doc2 pins reflect the average peak positive and negative excursions of the (filtered) receive signal, and could therefore be used to derive a measure of the data signal amplitude.

Note: These pins are driven from very high-impedance circuits, so that the DC load presented by any external circuitry should exceed 10MΩ to V_{BIAS}.

5 Application

5.1 Radio Channel Requirements

To achieve legal adjacent channel performance at high bit-rates, a radio with an accurate carrier frequency and an accurate modulation index is required. For optimum channel utilization, (e.g. low BER and high data-rates) attention must be paid to the phase and frequency response of both the IF and baseband circuitry.

5.1.1 Bit Rate, BT, and Bandwidth

The maximum data rate that can be transmitted over a radio channel depends on the following:

Channel spacing
Allowable adjacent channel interference
Tx filter bandwidth
Peak carrier deviation (Modulation Index)
Tx and Rx carrier frequency accuracies
Modulator and Demodulator linearity
Rx IF filter frequency and phase characteristics
Use of error correction techniques
Acceptable error-rate

As a guide to MOBITEX operation, a raw data-rate of 8kbps at 12.5kHz channel spacing may be achievable - depending on local regulatory requirements- using a ±2kHz maximum deviation, a BT of 0.3, and no more than 1.5kHz discrepancy between Tx and Rx carrier frequencies. Forward error correction (FEC) could then be used with interleaving to reduce the effect of burst errors.

Reducing the data-rate to 4.8kbps would allow the BT to be increased to 0.5, improving the error-rate performance.

5.1.2 FM Modulator, Demodulator and IF

For optimum performance, the eye pattern of the received signal (when receiving random data) applied to the CMX589A should be as close as possible to the Transmit eye pattern examples shown in Figure 10.

Of particular importance are general symmetry, cleanliness of the zero-crossings, and for a BT of 0.3, the relative amplitude of the inner eye opening.

To achieve this, attention must be paid to:

Linearity and frequency/phase response of the Tx frequency modulator. Unless the transmit data is especially encoded to remove low frequency components, the modulator frequency response should extend down to a few hertz. This is because two-point modulation is necessary for synthesized radios.

Bandwidth and phase response of the Rx IF filters.

Accuracy of the Tx and Rx carrier frequencies -any difference will shift the received signal towards one of the skirts of the IF filter response.

Ideally, the Rx demodulator should be DC coupled to the CMX589A Rx Signal In pin (with a DC bias added to center the signal at the Rx Feedback pin at $V_{DD}/2$ [V_{BIAS}]). However, AC coupling can be used provided that:

The 3dB cut-off frequency is 20Hz or below (i.e. a 0.1µF capacitor in series with 100kΩ).

The data does not contain long sequences of consecutive ones or zeroes.

Sufficient time is allowed after a step change at the discriminator output (resulting from channel changing or the appearance of a RF carrier) for the voltage into the CMX589A to settle before the RXDCacq line is strobed.

5.1.3 Two-Point Modulation

When designing the CMX589A into a radio that uses a frequency synthesizer, a two-point modulation technique is recommended. This is both to prevent the radio's PLL circuitry from counteracting the modulation process, and to provide a clean flat modulation response down to DC.

Figure 12 shows a suggested basic configuration to provide a two-point modulation drive from the CMX589A Tx Output using MX-COM's MX019 Digitally Controlled Quad Amplifier Array. The MX019 elements provide individual set-up, calibration and dynamic control of modulation levels. Level setting control of the amplifiers/attenuators of the MX019 is via an 8-bit data word. Note that the MX019 frequency response only supports data rates as high as 8kbps.

With reference to Figure 12:

The buffer amplifier is required to prevent loading of the CMX589A external RC circuit.

Stage B, with R1/R2, provides suitable signal and DC levels for the VCO varactor; C1 is RF decoupling. The drive level should be adjusted (digitally) to provide the desired deviation.

Stage C, with R3/R4, provides the Reference Oscillator drive (application dependent). This parameter is set by adjusting for minimum AC signal on the PLL control voltage with a low-frequency modulating signal (inside the PLL bandwidth) applied.

Stage D could be used with the components shown if a negative reference drive is required.

Stage A provides buffering and overall level control.

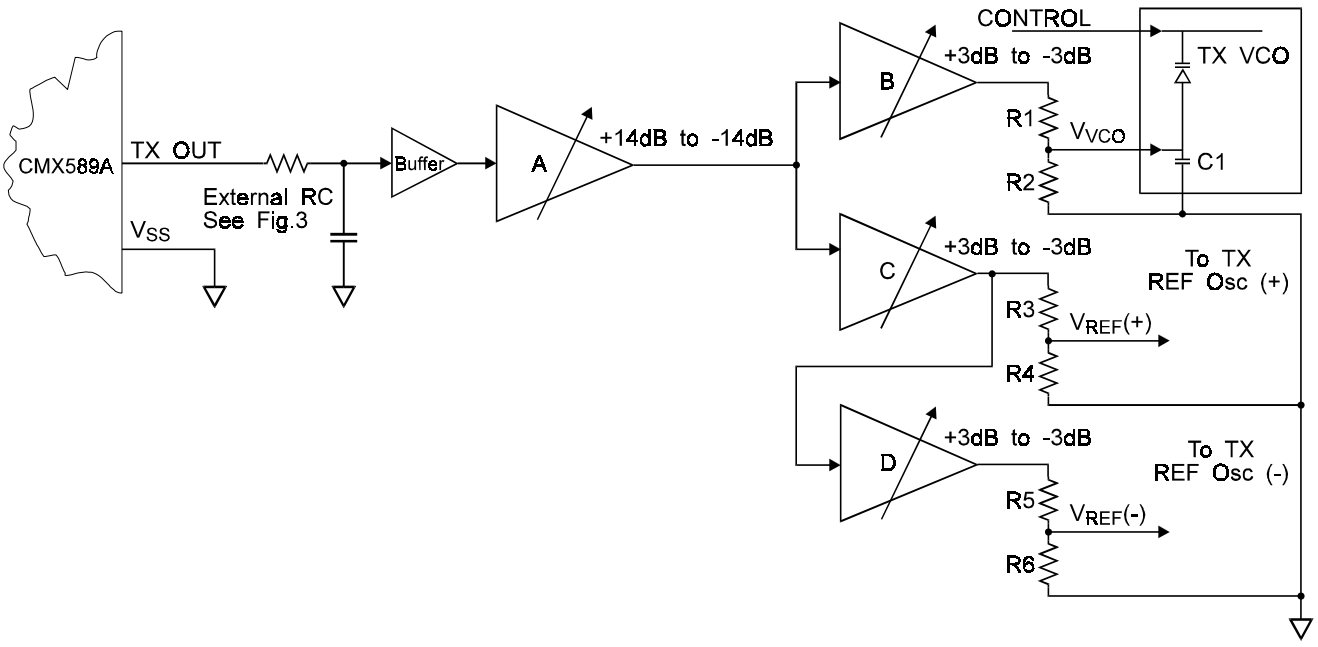


Figure 12: An Example of Two-Point Modulation Drive with Individual Adjustment Using the FX019

5.2 AC Coupling of Tx and Rx Signals

In practical applications, it is possible to arrange AC coupling between the CMX589A Tx Output and the frequency modulator to cut-off at a very low frequency, such as 5.0Hz. AC coupling between the receive discriminator and the input of the CMX589A may need a shorter time-constant to avoid problems from voltage steps at the output of the discriminator when changing channels or when the distant transmitter turns on.

For these reasons, as well as to maintain reasonable BER, the optimum –3dB cut-off frequencies are around 5.0Hz in the Tx path and 20.0Hz in the Rx path.

Figure 13 shows the typical static Bit-Error-Rate performance of the CMX589A operating under nominal conditions for various degrees of AC coupling at the Rx input and the Tx output.

Data Rate = 8kbps V_{DD} = 5.0V T_{AMB} = 25°C Tx BT = 0.3

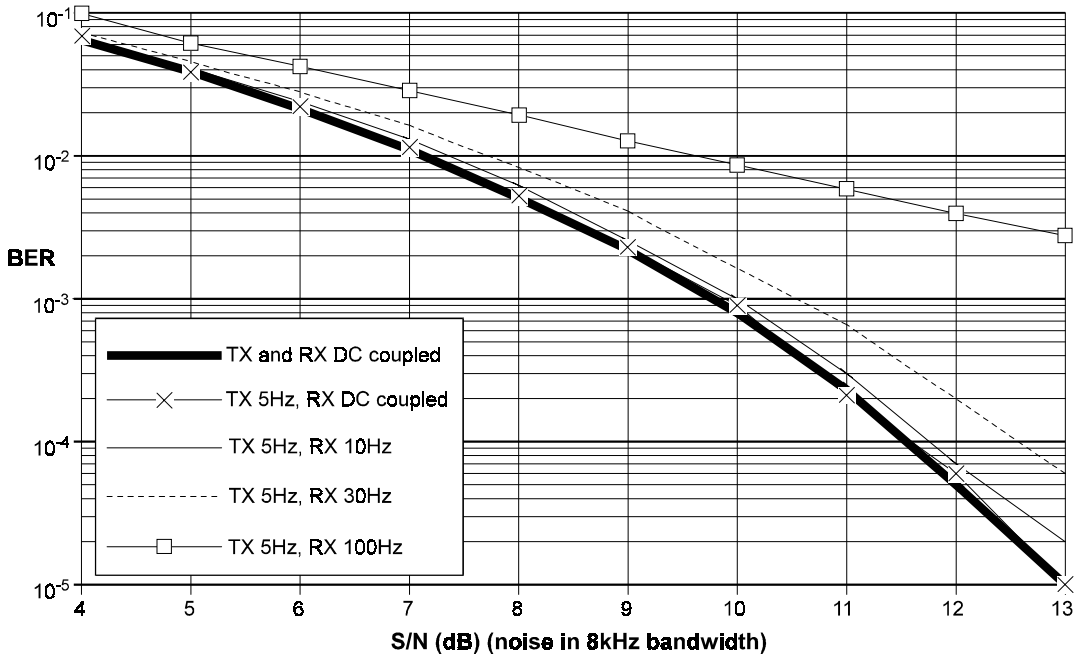


Figure 13: Effect of AC Coupling on Typical Bit-Error Rate

Any AC Coupling at the receive input will transform any step in the voltage at the discriminator output to a slowly decaying pulse which can confuse the modem's level measuring circuits. As illustrated in Figure 14, the time for this step to decay to 37% of its original value is 'RC' where:

$$RC = \frac{1}{2\pi(\text{the 3dB cutoff frequency of the RC network})}$$

which is 32ms, or 256 bit times at 8kbps, for a 5Hz network.

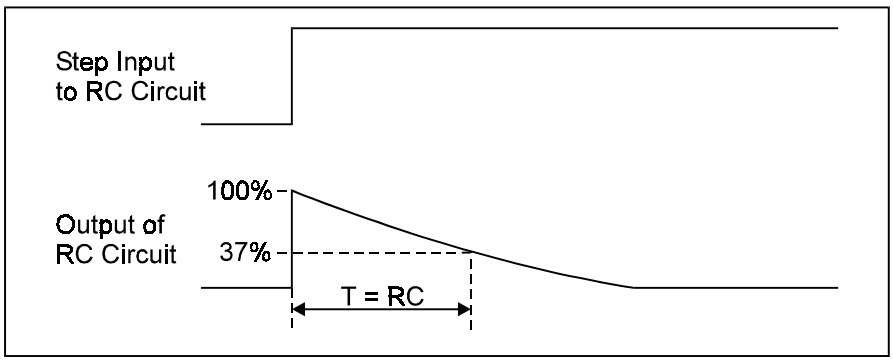


Figure 14: Decay time-AC Coupling

6 Performance Specifications

6.1 Electrical Specifications

6.1.1 Absolute Maximum Limits

Exceeding these maximum ratings can result in damage to the device.

General	Notes	Min.	Typ.	Max.	Units
Supply (V_{DD}-V_{SS})		-0.3		7.0	V
Voltage on any pin to V_{SS}		-0.3		V_{DD} + 0.3	V
Current					
V_{DD}		-30		30	mA
V_{SS}		-30		30	mA
Any other pin		-20		20	mA
D2 / P4 Packages					
Total allowable Power dissipation at T_{AMB} = 25°C				800	mW
Derating above 25°C			13		mW/°C
Operating Temperature		-40		85	°C
Storage Temperature		-55		125	°C
D5 Package					
Total allowable Power dissipation at T_{AMB} = 25°C				550	mW
Derating above 25°C			9		mW/°C
Operating Temperature		-40		85	°C
Storage Temperature		-55		125	°C
E2 Package					
Total allowable Power dissipation at T_{AMB} = 25°C				320	mW
Derating above 25°C			5.3		mW/°C
Operating Temperature		-40		85	°C
Storage Temperature		-55		125	°C

Table 7: Absolute Maximum Ratings

6.1.2 Operating Limits

Correct Operation of the device outside these limits is not implied.

	Notes	Min.	Typ.	Max.	Units
Supply (V_{DD}-V_{SS})		3.0	3.3/5.0	5.5	V
Operating Temperature		-40		85	°C
Rx and Tx Data Rate					
$V_{DD} \geq 3.0V$		4		32	kbps
$V_{DD} \geq 4.5V$		4		200	kbps
Xtal Frequency					
$V_{DD} \geq 3.0V$		1.0		5.0	MHz
$V_{DD} \geq 4.5V$		1.0		16.0	MHz
External Clock Frequency					
$V_{DD} \geq 3.0V$	1	1.0		5.0	MHz
$V_{DD} \geq 4.5V$	1	1.0		25.6	MHz
High Pulse Width	1	15			ns
Low Pulse Width	1	15			ns

Table 8: Operating Limits

Operating Limits Notes

1. Timing for an external clock input to the Xtal/Clock pin.

6.1.3 Operating Characteristics

For the following conditions unless otherwise specified.
V_{DD} = 5.0V @ T_{AMB} = 25°C
Xtal/Clock Frequency = 4.096MHz, Data Rate = 8kbps, Noise Bandwidth = Bit Rate

Static Values			**Notes**	**Min.**	**Typ.**	**Max.**	**Units**
Supply Current	Tx PS	Rx PS	1				
I_{DD} (V_{DD} = 3.0V)							
	1	1		-	0.5	-	mA
	0	1		-	1.0	-	mA
	1	0		-	1.0	-	mA
	0	0		-	1.5	-	mA
I_{DD} (V_{DD} = 5.0V)							
	1	1		-	1.0	-	mA
	0	1		-	2.0	-	mA
	1	0		-	3.0	-	mA
	0	0		-	4.0	-	mA
Input Logic Level							
Logic 1 Input Level				3.5	-	-	V
Logic 0 Input Level				-	-	1.5	V
Logic Input Current			2	-5.0	-	5.0	µA
Output Logic Level							
Logic 1 Output Level (I_{OH} = 120µA)				4.6	-	-	V
Logic 0 Output Level (I_{OL} = -120µA)				-	-	0.4	V
Transmit Parameters				-			
Tx OUT pin DC bias shift caused by change from Tx Enable = 0 to Tx Enable = 1 while Tx PSAVE = 0 at 25°C				-85.0	-	85.0	mV
Tx OUT, Output Impedance			3	-	1.0	-	kΩ
Tx Out, Level			4, 10	0.8	1.0	1.2	V_{P-P}
Output DC Offset			12	-0.125	-	0.125	V
Tx Data Delay							
BT = 0.3			5	-	2.0	2.5	bit-periods
BT = 0.5			5	-	1.5	2.0	bit-periods
Tx PS to Output-Stable time			6	-	4.0	-	bit-periods
Receive Parameters							
Rx Amplifier							
Input Impedance				1.0	-	-	MΩ
Output Impedance			7	-	10.0	-	KΩ
Voltage Gain				-	50.0	-	dB
Rx Filter Signal Input Level			8, 10	0.7	1.0	1.3	V_{P-P}
Rx Time Delay			9	-	-	3.0	bit-

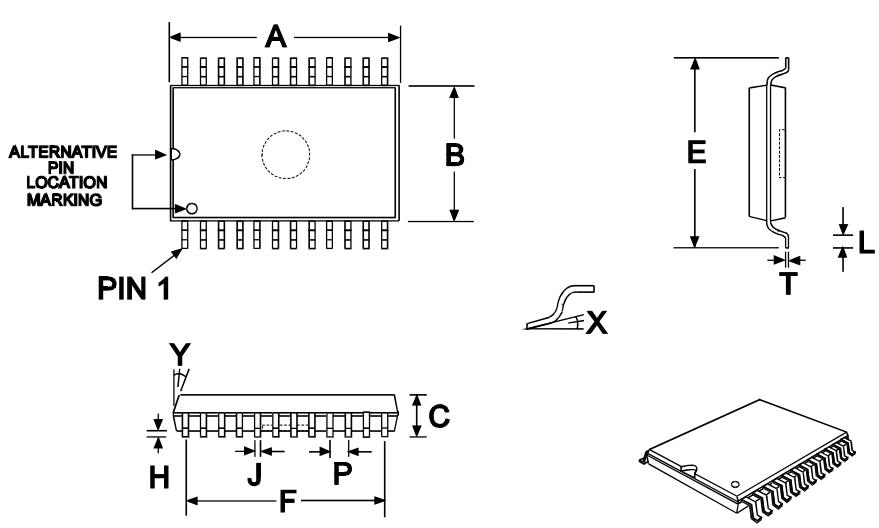
Static Values	Notes	Min.	Typ.	Max.	Units
					periods
On-Chip Xtal Oscillator					
R_{IN}		10.0	-	-	MΩ
R_{OUT}	11	-	50.0	-	kΩ
Voltage Gain	11	-	25.0	-	dB

Table 9: Operating Characteristics

Operating Characteristics Notes:

1. Not including current drawn from the CMX589A pins by external circuitry. See Absolute Maximum Ratings.
2. For V_{IN} in the range V_{SS} to V_{DD}.
3. For a load of 10KΩ or greater. Tx PS input at logic '0'; Tx Enable = '1'.
4. Data pattern of 1111000011110000…
5. Measured between the rising edge of Tx Clock and the centre of the corresponding bit at Tx Out.
6. Time between the falling edge of the Tx PS and the Tx Out voltage stabilizing to normal output levels.
7. For a load of 10kΩ or greater. Rx PS input at logic '0'.
8. For optimum performance, Measured at the Rx Feedback pin for an 1111000011110000… pattern.
9. Measured between the center of bit at Rx Signal In and corresponding rising edge of the Rx Clock.
10. Levels are proportional to applied V_{DD}
11. Small signal measurement at 1.0kHz with no load on Xtal output.
12. (Tx OUT enabled DC level) – (Tx Out disabled DC level) when transmitting a repeating 11110000 bit pattern.

6.2 Packages



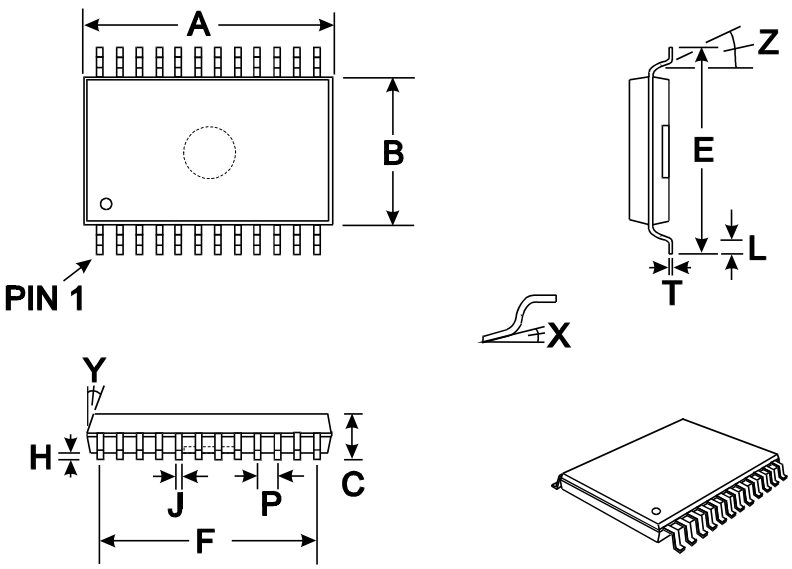
DIM.	MIN.	TYP.	MAX.
* A	0.303 (7.70)		0.311 (7.90)
* B	0.169 (4.30)		0.177 (4.50)
C	------		0.047 (1.20)
E	0.248 (6.30)		0.256 (6.50)
F		0.286 (7.15)	
H	0.002 (0.05)		0.006 (0.15)
J	0.007 (0.17)		0.012 (0.30)
L	0.020 (0.50)		0.030 (0.75)
P		0.026 (0.65)	
T	0.003 (0.08)		0.008 (0.20)
X	0°		7°
Y		12°	

NOTE :

* A & B are reference datum's and do not include mold deflash or protrusions.

All dimensions in inches (mm.)
Angles are in degrees

Figure 15: 24-pin TSSOP Mechanical Outline: ***Order as part no. CMX589AE2***



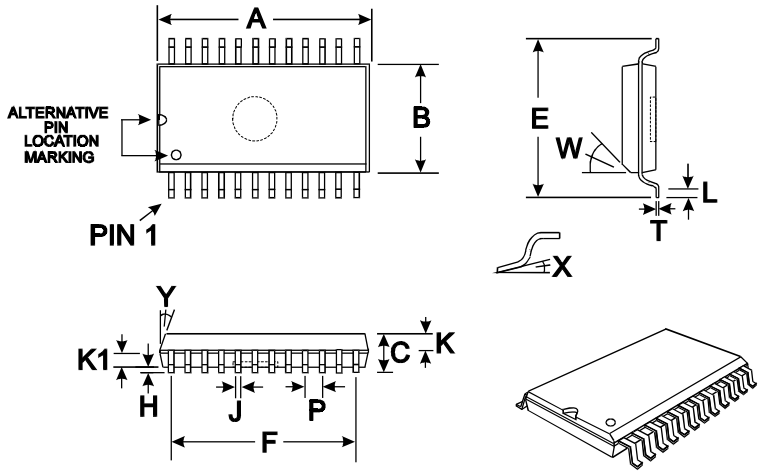
DIM.	MIN.	TYP.	MAX.
* A	0.318 (8.08)		0.328 (8.33)
* B	0.205 (5.21)		0.213 (5.39)
C	0.066 (1.67)		0.079 (2.00)
E	0.301 (7.65)		0.312 (7.90)
F		0.281 (7.15)	
H	0.002 (0.05)		0.008 (0.21)
J	0.010 (0.25)		0.015 (0.38)
L	0.022 (0.56)		0.037 (0.94)
P		0.026 (0.65)	
T	0.005 (0.13)		0.009 (0.23)
X	0°		8°
Y	7°		9°
Z	4°		10°

NOTE :

* A & B are reference datum's and do not include mold deflash or protrusions.

All dimensions in inches (mm.)
Angles are in degrees

Figure 16: 24-pin SSOP Mechanical Outline: ***Order as part no. CMX589AD5***



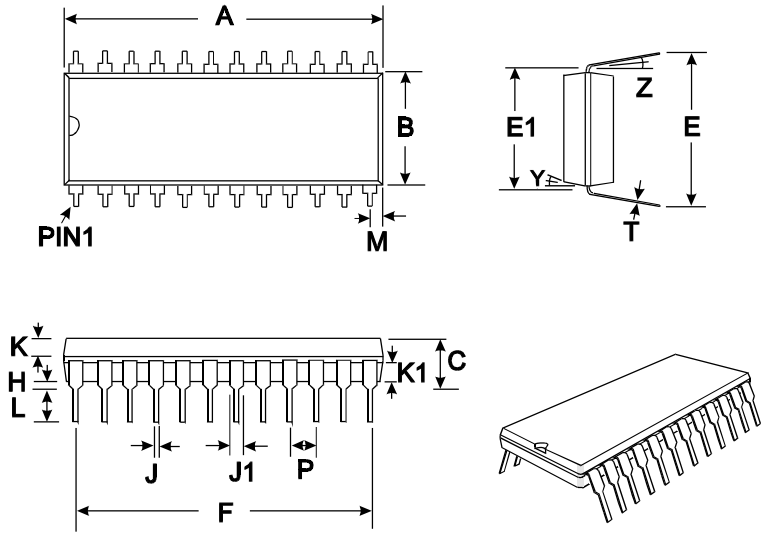
DIM.	MIN.	TYP.	MAX.
* A	0.597 (15.16)		0.613 (15.57)
* B	0.286 (7.26)		0.299 (7.59)
C	0.093 (2.36)		0.105 (2.67)
E	0.390 (9.90)		0.419 (10.64)
F		0.550 (14.1)	
H	0.003 (0.08)		0.020 (0.51)
J	0.013 (0.33)		0.020 (0.51)
K		0.041 (1.04)	
K1		0.041 (1.04)	
L	0.016 (0.41)		0.050 (1.27)
P		0.050 (1.27)	
T	0.009 (0.23)		0.0125 (0.32)
W		45°	
X	0°		10°
Y		7°	

NOTE :

* A & B are reference datum's and do not include mold deflash or protrusions.

All dimensions in inches (mm.)
Angles are in degrees

Figure 17: 24-pin SOIC Mechanical Outline: *Order as part no. CMX589AD2*



DIM.	MIN.	TYP.	MAX.
* A	1.200 (30.48)		1.270 (32.26)
* B	0.500 (12.70)		0.555 (14.10)
C	0.151 (3.84)		0.220 (5.59)
E	0.600 (15.24)		0.670 (17.02)
E1	0.590 (14.99)		0.625 (15.88)
F		1.10 (27.94)	
H	0.015 (0.38)		0.045 (1.14)
J	0.015 (0.38)		0.023 (0.58)
J1	0.040 (1.02)		0.065 (1.65)
K	0.066 (1.68)		0.074 (1.88)
K1	0.060 (1.52)		0.074 (1.88)
L	0.121 (3.07)		0.160 (4.06)
M		0.180 (4.58)	
P		0.100 (2.54)	
T	0.008 (0.20)		0.015 (0.38)
Y		7°	
Z		4°	

NOTE :

* A & B are reference datum's and do not include mold deflash or protrusions.

All dimensions in inches (mm.)
Angles are in degrees

Figure 18: 24-pin PDIP Mechanical Outline: *Order as part no. CMX589AP4*

Handling precautions: This product includes input protection, however, precautions should be taken to prevent device damage from electro-static discharge. CML does not assume any responsibility for the use of any circuitry described. No IPR or circuit patent licences are implied. CML reserves the right at any time without notice to change the said circuitry and this product specification. CML has a policy of testing every product shipped using calibrated test equipment to ensure compliance with this product specification. Specific testing of all circuit parameters is not necessarily performed.

Consumer Microcircuits Limited
M-I-X-E-D S-I-G-N-A-L S-O-L-U-T-I-O-N-S

1 WHEATON ROAD
WITHAM - ESSEX
CM8 3TD - ENGLAND

Telephone: +44 1376 513833
Telefax: +44 1376 518247
e-mail: sales@cmlmicro.co.uk
website: http://www.cmlmicro.co.uk

CML Microcircuits

COMMUNICATION SEMICONDUCTORS

CML Product Data

In the process of creating a more global image, the three standard product semiconductor companies of CML Microsystems Plc *(Consumer Microcircuits Limited (UK), MX-COM, Inc (USA) and CML Microcircuits (Singapore) Pte Ltd)* have undergone name changes and, whilst maintaining their separate new names *(CML Microcircuits (UK) Ltd, CML Microcircuits (USA) Inc and CML Microcircuits (Singapore) Pte Ltd*), now operate under the single title **CML Microcircuits**.
These companies are all 100% owned operating companies of the CML Microsystems Plc Group and these changes are purely changes of name and do not change any underlying legal entities and hence will have no effect on any agreements or contacts currently in force.

CML Microcircuits Product Prefix Codes

Until the latter part of 1996, the differentiator between products manufactured and sold from MXCOM, Inc. and Consumer Microcircuits Limited were denoted by the prefixes MX and FX respectively. These products use the same silicon etc. and today still carry the same prefixes. In the latter part of 1996, both companies adopted the common prefix: CMX.

This notification is relevant product information to which it is attached.

Company contact information is as below:

CML Microcircuits (UK) Ltd
COMMUNICATION SEMICONDUCTORS

Oval Park, Langford, Maldon,
Essex, CM9 6WG, England
Tel: +44 (0)1621 875500
Fax: +44 (0)1621 875600
uk.sales@cmlmicro.com
www.cmlmicro.com

CML Microcircuits (USA) Inc.
COMMUNICATION SEMICONDUCTORS

4800 Bethania Station Road,
Winston-Salem, NC 27105, USA
Tel: +1 336 744 5050,
0800 638 5577
Fax: +1 336 744 5054
us.sales@cmlmicro.com
www.cmlmicro.com

CML Microcircuits (Singapore) Pte Ltd
COMMUNICATION SEMICONDUCTORS

No 2 Kallang Pudding Road, 09-05/
06 Mactech Industrial Building,
Singapore 349307
Tel: +65 7450426
Fax: +65 7452917
sg.sales@cmlmicro.com
www.cmlmicro.com

D/CML (D)/1 February 2002

Appendix C

TM8115 Data sheet

This appendix presents properties of the TAIT voice-and-data radio discussed in Section 2.7.1.

Rugged • High Speed Data • Integrator Friendly

TM8115 mobile

The Tait 8000 Series of innovative and high-performing products sets a standard of excellence for analogue radio communications technology. With advanced software-flexible features, 8000 Series products lead their class.

25W, 24-Channel Conventional Radio •
Intuitive Interface with 2-Digit Display •
Digital Controller Design •
Optional Internal High Speed Data Modem •
Rugged Construction •
Robust RF Performance •
Ultimate Flexibility for System Integration •
Internal AVL Software Support •
Windows-based Programming Application •

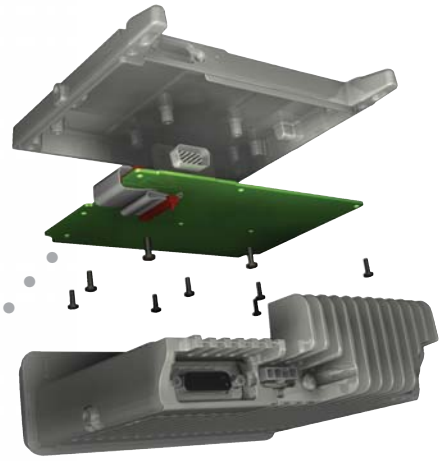
TAIT radio communications

TM8115 mobile

Leading its class, the new TM8115 is a robust, software-flexible radio, ideal for a wide range of voice and data applications.

Based on Digital Signal Processor (DSP) technology and capable of transmitting high speed data through an internal software modem, these new radios offer everything you need for reliable voice and data communications — and have been designed with ultimate flexibility for system integration.

The TM8115 offers high performing, innovative and proven technology at a competitive price.



large options board area

multiple D-range and serial ports

Features

Digital Controller Design
The TM8115 features a state-of-the-art DSP, providing exceptionally fast and reliable data processing. This optimises performance of IF filtering, FM demodulation, and transceiver and receiver audio processing.

Data Ready
Suited to the latest mobile data applications, the TM8115 supports 1200 baud FFSK data out-of-the-box. System integrators can also connect external high speed modems via a robust D-range. A transmitter rise time under 10mS and very low group delay make the TM8115 the perfect platform for data applications.

Internal High Speed Data Modem — Software Option
This optional software-based modem supports over 9600 bits per second and is activated with a software key. Digital processing optimises RF performance gains for both built-in and external modems, improving the integrity of your data.

Rugged Construction
Engineered with a strong diecast metal chassis and almost entirely constructed using Surface Mount Technology (SMT), the TM8115 meets stringent specifications for reliability including the MIL-STD 810 C, D, E & F specs and IP54.

Robust RF Performance
The TM8115 has been built using innovative RF design. When the radio detects unusually high transmitter temperatures it reduces power output so it can continue to operate effectively.

Advanced System Integration Capabilities
Featuring industry-leading integration capabilities, the TM8115 has been designed with customisation in mind. The built-in modems, serial control, AVL engine support and high speed data make the TM8115 a market leader. The integrator has maximum design flexibility with multiple ports for auxiliary connectors and a large options board area. The comprehensive 3DK Hardware Developer's Kit provides hardware and software tools for customisation.

Application-Ready Software
The TM8115 features many formats and hooks for application development. An AVL application can be easily integrated with the supported polling vehicle location format and direct connect GPS receiver port. Mobile alarm monitoring, asset tracking, and public address features are also simple additions with the TM8115 software interfaces.

Standard Features:
- **Full Selcall Functionality**
- **DTMF Encoder**
- **Built-in CTCSS/DCS**
- **Multiple Scanning and Voting Formats Supported**
- **Emergency Mode, Stun and Revive**
- **Low Stand-by Power Consumption**
- **Three Control Signal Outputs**
- **Four RF Power Levels**
- **Three Mute Settings**
- **Handset Audio**
- **Remote Volume Control**
- **AVL Software Support**
- **Third Party Control Head Capable**

Optional Features:
- **Public Address System**
- **Comprehensive 3DK Hardware Developer's Kit with Target Board and Development Software Examples**
- **Blank Control Head**

TAIT radio communications

TM8105 data radio also available

TM8115 Specifications

General

Frequency Range	136-174MHz
	400-470MHz
	450-530MHz
	216-266MHz
Frequency Stability	+/-1.5ppm
Channel Capacity	24 channels (simplex or semi-duplex)
Power Supply	10.8-16VDC
Channel Spacing	12.5/20/25kHz
Dimensions (LxWxH)	175 x 160 x 50mm
	6.88 x 6.29 x 1.97 inches
Weight	1.43kg
	50.44oz
Operational Temperature	-30˚C to +60˚C (-22˚F to +140˚F)
Sealing	Passes dust and rain testing to IP54
Vibration	Meets IEC 60571
Low Pressure	MIL-STD 810C,D,E & F
High Temperature	MIL-STD 810C,D,E & F
Low Temperature	MIL-STD 810C,D,E & F
Temperature Shock	MIL-STD 810C,D,E & F
Solar Radiation	MIL-STD 810C,D,E & F
Rain	MIL-STD 810C,D,E & F
Humidity	MIL-STD 810C,D,E & F
Salt Fog	MIL-STD 810C,D,E & F
Dust	MIL-STD 810C,D,E & F
Vibration	MIL-STD 810C,D,E & F
Shock	MIL-STD 810C,D,E & F
RF Connector	50 Ohm BNC/mini UHF
Interface Connectors	3 Interface Connectors with Serial Ports
Programmable Digital I/O and Audio Tap Points	

Transmitter

Output Power	1,5,10,25W
Modulation Limiting	<+/-2.5kHz 12.5kHz
	<+/-4kHz 20kHz
	<+/-5kHz 25kHz
FM Hum and Noise	>38dB 12.5kHz
	>41dB 20kHz
	>43dB 25kHz
Conducted/Radiated Emissions	<-36dBm to 1GHz
	<-30dBm 1-4GHz below 500MHz
	<-30dBm 1-12.75GHz above 500MHz
Audio Response	300-3kHz
	Flat or with pre-emphasis
Audio Distortion	< 3% at 1kHz 60% mod
Transmit Rise Time	<10mS

Receiver

Sensitivity	< -118dBm for 12dB SINAD
Intermodulation	>66dB
Selectivity	>65dB 12.5kHz
	>70dB 20kHz
	>75dB 25kHz
Spurious Responses	>72dB
Audio Distortion	<3%
Hum and Noise	>40dB 12.5kHz
	>41dB 20kHz
	>43dB 25kHz
Audio Response	300-3kHz
	Flat or de-emphasis
Receive Detect Time	< 3mS

Authorised Dealer

Specifications are subject to change without notice and shall not form part of any contract. They are issued for guidance purposes only. For further information please check with your nearest Tait office or authorised dealer.

FCC Rule 2.803 (c)(e)(iii) applies: This device has not been authorised as required by the rules of the Federal Communications Commission. This device is not, and may not be offered for sale or lease, or sold or leased, until authorisation is obtained.

www.taitworld.com