Power and Bandwidth Allocation in Multibeam Satellite Systems

Major Project

Bachelor of Technology

in

Computer Science and Engineering

by

Shubham Kumar Singh

Dipali Mishra

Aman Pasi

Under the supervision of

Dr. Santosh kumar Mahto

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RANCHI

Acknowledgement

Firstly, we would like to thank *IIIT Ranchi* for giving us an opportunity to perform this project. We surely have learnt a lot from here. We hope for everything that we have learnt here will be helpful us in future.

With great pleasure we would like to express our deep sense of gratitude to Dr. Santosh kumar Mahto (Dept. of Electronics & Communication Engg) for guiding us throughout the journey of our project. We would like to thank them for having immense patience while teaching us about the project and sparing his previous time for us. We would also like to thank our Director Prof. Vishnu Priye for providing facilities and infrastructure in the department and would also like to thank other faculties for providing us a great opportunity to work on such an important project.

Abstract

Communications satellites are becoming more flexible and complex. New generations feature on-board signal processing and provide hundreds of Gbit/s of throughput. Therefore, *dynamic resource management* (DRM) is increasingly becoming one of the challenges faced today by satellite operators.

In recent years there has been considerable interest in DRM for communications satellites. Previous studies have proposed power allocation to beams using genetic algorithms and joint power and bandwidth allocation using analytical approaches as well as the Lagrange multipliers technique. Nevertheless, there is a research gap in *metaheuristics/artificial intelligence algorithms* in communications satellites to allocate both power and bandwidth while considering important factors such as the interference between beams. Therefore, this thesis aims to develop a new methodology of power and bandwidth allocation in multi-beam satellite systems taking into account interference's and to investigate the improvement of dynamically allocating bandwidth in addition to power.

First, the model is explained and results on simple case studies are shown in order to analyze how power and bandwidth influence the data rate in communications satellites. Then, the proposed algorithm is explained and the outcomes from simulations are presented. Results indicate that apart from allocating power, the *unmet system capacity* (USC) can be further reduced by allocating bandwidths per beam. The genetic algorithm implemented in this thesis can diminish the USC in 10% - 15%. Furthermore, the demand variation has an effect on that improvement: the higher this variation is, the more important it is to allocate also bandwidths and to allow them a higher flexibility in order to obtain the best solution (and vice versa).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The communications satellites market has experienced disruptive technological advancements in the last 10 years driven by the need to satisfy the increasing demand for connectivity services in remote locations not served by ground infrastructure, and the expansion of the mobility sector (airplanes and ships)
Spot beams have been one of the most significant advancements leading to the increase in capacity in modern satellites. In a spot beam the signal power is focused on a specific area of the Earth's surface, being the beam's footprint on the order of several hundreds of kilometers. Current-generation high-throughput satellites (HTS) – such as ViaSat-2 and EchoStar 24 – provide 300 - 600 Gbps of capacity and have hundreds of beams. Future generations such as ViaSat-3 or SES's mPower MEO constellation will have thousands of beams and capacities in the Tbps range. Finally, phased-array antenna technology is being introduced into next-generation satellites, enhancing the capabilities of spot beams by allowing for reconfigurable numbers of beams, boresight pointing, and beamforming.

To further expand the capabilities of their satellites and serve the rising demand, operators are transitioning from rigid bent-pipe architectures towards flexible and reconfigurable satellite architectures [2], as shown in Fig. 1; in newer designs analog payloads are replaced with digital payloads, which allow for dynamic on-demand resource allocation [3]. In particular, whereas older generations of satellites had a bent-pipe architecture (the uplink signal was relayed back to Earth through a particular beam after being amplified and shifted in frequency), new satellites provide advanced adaptability such as dynamic power and bandwidth allocations, beam shaping and beam steering, signal demodulation and remodulation, routing capabilities, etc.

However, with greater flexibility comes increased complexity. The larger number of beams and multiple configurable variables for each of them (power, bandwidth, boresight pointing, etc.) requires the use of advanced techniques for dynamic resource management (DRM), which has become a popular topic of research both in industry and academia due

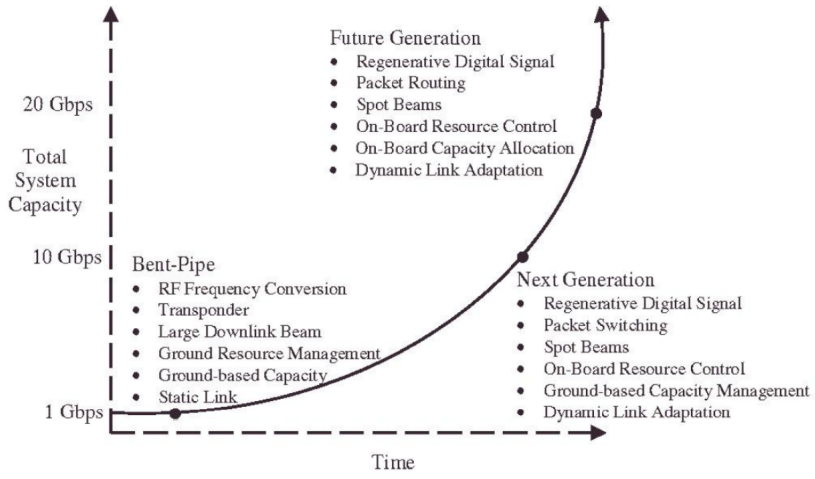

Figure 1.1: **Technological advancements in successive generations of communications satellites. Adapted from**

to the increasing number of companies planning to launch satellites with flexible architectures. Among others, SES (mPower, SES-17), Telesat, Intelsat, and SpaceX (Starlink) have scheduled launches of such systems within the next 5 years.

1.2 Literature Review

Recent literature has examined the issue of resource allocation in communications satellites and optimization [4]. A number of studies focus on **beam hopping techniques** in which only a portion of the satellite beams are active at a given time. In particular, the authors in [5] proposed a genetic algorithm to optimize the active beams' time plan, whereas [6] analyzed the advantages of applying beam hopping to conventional satellite systems.

Another area of research is **cognitive satellite communications**, which consists in monitoring the spectrum dedicated to other systems and exploiting this spectrum if it is not being used. Authors in [7], [8] and [9] propose beam-forming [10] and bandwidth allocation in a spectral coexistence scenario of satellites and ground users.

Finally, with regard to algorithms for *power and/or bandwidth allocation*, Aravanis et al. [11] developed a hybrid genetic algorithm and simulated annealing method to allocate power to each beam so that the **unmet system capacity (USC) and the total power used were minimized**. Based on the duality theory, the authors in [12] developed an iterative power and bandwidth allocation algorithm that penalized delays, but it ignored interference between beams.

This literature review reveals that there is a gap in existing research, in that multibeam-satellite algorithms that allocate both power and bandwidth (while considering important factors such as the interference between beams) have not yet been addressed.

1.3 Research Goals

This goal of this project is:

- Analyze power and bandwidth influence on the performance of a communications satellite with and without interference and atmospheric attenuation
- To investigate the performance improvement obtained by dynamically allocating bandwidth, as compared to previous studies where only power was allocated.

Chapter 2

System Models

2.1 Link Model

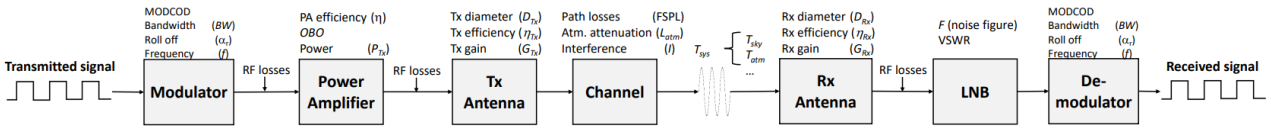

Figure 2.1: **Link model block diagram**

Figure 2.1 shows an overview of the link-model, as well as the parameters considered by each of the blocks. These parameters are the inputs for the link-budget equation, which is used to **compute the achievable data rate** for each of the beams. As can be seen in the Figure, in our link-budget we consider the effects of the full RF chain, from digital signal modulation to demodulation, including power amplifier and LNB considerations.

Space communications are established through wireless links. To design them, engineers use an equation commonly termed the "link budget equation" that accounts for the gains and losses in power from the transmitter, through the different components of the medium and to the receiver. In other words, the link budget equation determines the received power after subtracting all the power losses to the transmit power.

```
import math
from scipy import constants

def getSignalNoiseRatio(PTx, OBO, GTx, GRx, Tsys, L):

    '''
    C/N0 = PTx - OBO + GTx + GRx - L - 10*log10(k*Tsys)
```

```
    L = FSPL + Latm + LRFTx + LRFRx
    PTx : transmitted power (dB)
    OBO : power-amplifier output back-off
    GTx : Transmitting Antenna Gains (dB)
    GRx : Receiving Antenna Gains (dB)
    Tsys : System Temperature (K)
    L : sum of losses (dB)
    k: Boltzmann constant 1.380649    10-23
    '''
    k = constants.k
    signalNoiseRatio = PTx - OBO + GTx + GRx - L - (10*math.log(k*Tsys,10))

    return signalNoiseRatio


def getSystemTemperature(Tant, LRF, Latm, Tatm, Tw):

    '''
    where Tant is the antenna temperature (K),
    Tatm is the atmospheric temperature (K),
    and Tw is the waveguide temperature (K).
    Latm are the total atmospheric losses (dB), and
    LRF are the RF losses in reception (dB).
    '''

    Tsys = Tant*10**(-(LRF/10)) + Tatm*10**(-(Latm + LRF)/10) +
           Tw*(1 - 10**(-LRF/10))

    return Tsys


def getSumOfLosses(FSPL, Latm, LRFTx, LRFRx):

    '''
    L is the sum of the losses considered, (dB).
    In particular, we consider free-space path losses (FSPL),
    atmospheric losses (Latm), and losses in the
    transmitting and receiving RF chains (LRFTx and LRFRx, respectively).
    '''
```

```
    L = FSPL + Latm + LRFTx + LRFRx

    return L

def getLink(SNR, BW, Rb, CABI=1, CASI=1, CXPI=1, C3IM=1):

    '''
    x -> C/(N0 + I)= (1/CABI + 1/CASI + 1/CXPI + 1/C3IM + 1/SNR)**-1
    y -> Eb/(N + I)= C/(N0 + I)*(BW/Rb)
    where Rb is the link data rate (bps) , and BW
    is the bandwidth allocated to that beam (Hz). Notice how
    our link budget equation considers four different types of
    interference (CABI, CASI, CXPI, and C3IM). In above Eqs all terms
    are in linear scale.
    SNR -> Signal to Noise Ratio = C/N0
    BW -> Bandwidth allocated to Beam (Hz)
    Rb -> link data rate (bps)
    '''
    # use try catch block here
    x = 1/(1/CABI + 1/CASI + 1/CXPI + 1/C3IM + 1/SNR)
    y = x*BW/Rb

    return y

def getBeamDataRate(BW, Ar, y):

    '''
    Finally, the beam data rate is computed as
    Rb = BW/(1 +  r )*T(Eb/(N + I)) [bps],
    where  r  is the roll-off factor, and    is the spectral efficiency
    of the modulation and coding scheme (MODCOD) (bps/Hz),
    which depends on the Eb/(N + I)
    As shown, the spectral efficiency (T) is a function
    of the power: higher powers yield higher ratios of energy
    per bit to noise power spectral density
    '''


    Rb = BW/(1+Ar)*spectralEfficiency(y)

    return Rb
```

Note that in order to carry out the link-budget computations, one needs to assume *a-priori* that a given MODCOD scheme is used, compute Eqs. 1 - 6, and then verify whether condition in Eq. 7 is satisfied.

In this paper it is assumed that the satellites use the MODCOD schemes defined in the standard DVB-S2X [19], which corresponds to the second generation standard developed by the Digital Video Broadcast Project. This standard is the most popular for broadcasting, interactive services, and broadband services for space-based communications, and it defines the channel coding, framing structure, and modulation schemes to be employed. As part of the standard, a set of more than 60 MODCODs are provided, with modulation orders ranging from BPSK to 256-APSK, and coding rates ranging from 1/4 to 9/10.

Finally, to estimate the output back-off (OBO) for each of the MODCODs, we generate a synthetic sequence of 100,000 symbols and assume the OBO equals the peak-to-average power ratio of such a sequence (computed as the ratio between the 99.9th percentile power and the average power). Notice that this is an over-estimation of the required OBO, as in a real scenario, one could optimize the OBO value by simulating the channel and the RF chains in transmission and reception, as well as further reduce it by using pre-distortion techniques to push the amplifier closer to saturation.

Chapter 3

Problem Statement

3.1 Problem Statement

We consider a satellite with N fixed-pointing beams, whose power and bandwidth can be dynamically allocated to satisfy the long-term estimated demand of each beam. The objective is to assign an average power and bandwidth to each of the N beams, such that the **unmet system capacity (USC)** is minimized, while satisfying a set of constraints imposed by the satellite.

Our figure of merit, the USC, represents the fraction of the demand that is not satisfied by the satellite, and is computed as indicated in Eq. 10 (note that there are no extra gains for offering a data rate that exceeds the demand). This metric is more suitable than its complement – the **met system capacity** (MSC) – because of its economic significance: ***communications satellite companies have to pay penalty fees when they fail to meet their customer service-level agreements (SLAs), and thus their interest in minimizing the USC.*** Moreover, this metric was used in this paper's baseline reference, [12], where it was chosen based on the comparison of several metrics carried out in [27].

```
def getUnmetSystemCapacity(N, Db = [], Rb = []):

    '''
    If the demand Db higher than the data rate Rb,
    the sum of the MSC and the USC equals the demand.
    On the other hand, if the demand is lower than the data rate,
    the MSC equals the demand and the USC is 0.
    '''
    usc = 0
    for i in range(N):
        usc = usc + max(Db[i] - Rb[i], 0)
```

```
return usc
```

Conceptually, the relationship between the MSC and the USC is shown in Figure 2.2. As it can be seen, the sum of the MSC and the USC equals the demand, and if the demand is lower than the data rate, the USC is 0 (the MSC equals the demand).

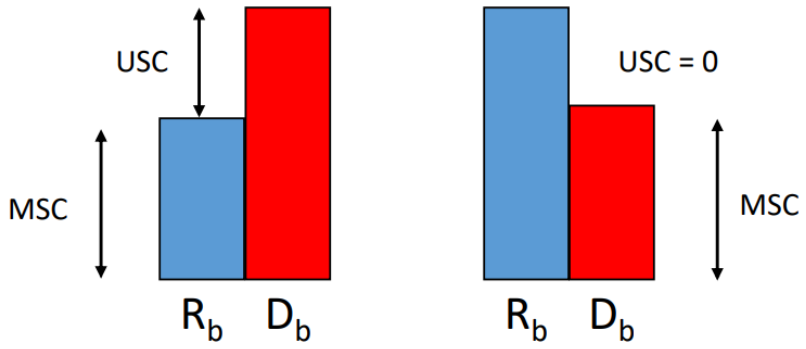

Figure 3.1: **Conceptual depiction of the MSC and USC metrics**

In terms of constraints, we assume that the satellite has a total bandwidth Btot, a total available power Ptot, and that the maximum transmit power per beam is Pmax. As mentioned before, our objective is to minimize the USC (Eq. 10) by allocating power and bandwidth to beams in a satellite subject to the constraints.

$$\begin{aligned}
&\underset{P_b,B_b}{\text{minimize}} && \text{USC}(P_b, B_b) && && (11)\\
&\text{subject to} && B_b \leq B_{tot} && \forall b \in \mathcal{B} && (12)\\
& && P_b \leq P_b^{max} && \forall b \in \mathcal{B} && (13)\\
& && \sum_{b=1}^{N} P_b \leq P_{tot} && && (14)\\
& && B_a + B_b \leq B_{tot} && \forall (a,b)_{adj,p} && (15)
\end{aligned}$$

- Pb and Bb are, respectively, the transmit power and bandwidth of **beam b**
- B is the set of beams in the satellite
- N is the total number of beams
- Equation 12 imposes the constraint that the bandwidth in any beam cannot exceed the satellite's total bandwidth.

- Equation 13 is needed to capture the saturation value of power amplifiers
- Eq. 14 ensures that beams do not use more power than what is available.
- Equation 15 is required to ensure that adjacent beams do not interfere.

3.2 Genetic Algorithm

```
# install python packages
!pip install deap

# import all required packages
import random
from deap import base
from deap import creator
from deap import tools
from deap import gp


# get randomly generated power and beam
def random_value():
    return (random.random(),random.uniform(500, 10000))


def evaluate(ind):
    ind = constraint_handling(ind)
    sat = Satellite(ind)
    sat.run_link_budget()
    return USC(sat)


Nind = 400 # Population Size
creator.create("FitnessMin", base.Fitness, weights=(-1.0, -1.0))
creator.create("Individual", list, fitness=creator.FitnessMin)
toolbox = base.Toolbox()
# (p, b) allocations of power and bandwidth to every beam in the satellite)
toolbox.register("attribute", random_value)
toolbox.register("individual", tools.initRepeat, creator.Individual,
                 toolbox.attribute, n=Nind)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("evaluate", evaluate)
```

```
toolbox.register("crossover", tools.cxBlend)
toolbox.register("mutate", gp.mutUniform) # change me
toolbox.register("select", tools.selTournament, tournsize=5)


def crossover_blend(offspring):
    for child1, child2 in zip(offspring[::2], offspring[1::2]):
    if random.random() < CXPB:
        toolbox.crossover(child1[0], child2[0], Axvr)
        del child1.fitness.values
        del child2.fitness.values


def mutation(offspring):
    for mutant in offspring:
        if random.random() < MUTPB:
            toolbox.mutate(mutant[0])
            del mutant.fitness.values


Ngen = 750
Nmin = 75
thresh = 0.05 # threshold in percentage
Axvr = 0.2 # Blend alpha
def main():
    pop = toolbox.population(n=300)

    # Evaluate the entire population
    fitnesses = list(map(toolbox.evaluate, pop))
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit

    # CXPB  is the probability with which two individuals are crossed
    # MUTPB is the probability for mutating an individual

    CXPB, MUTPB = 0.95, 0.15

    # Extracting all the fitnesses of
    fits = [ind.fitness.values[0] for ind in pop]
```

```
# Variable keeping track of the number of generations
gen = 0

while gen < Ngen:
    # A new generation
    g = g + 1
    print("— Generation %i —" % g)

    # Select the next generation individuals
    offspring = toolbox.select(pop, len(pop))
    # Clone the selected individuals
    offspring = list(map(toolbox.clone, offspring))

    # Apply crossover and mutation on the offspring
    crossover_blend(offspring)
    mutation(offspring)

    # Evaluate the individuals with an invalid fitness
    invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
    fitnesses = map(toolbox.evaluate, invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    pop[:] = offspring

    if gen > Nmin then:
        min_1 = min_USC(gen)
        min_2 = min_USC(gen - 1)
        min_3 = min_USC(gen - 2)
        improv_1 = max(min_2 - min_1)/min_2
        improv_2 = max(min_3 - min_2)/min_3

        if improv_1 < thresh and improv_2 < thresh:
            break

best = pop[np.argmin([toolbox.evaluate(x) for x in pop])]
return best
```

Table 3: Genetic algorithm parameters

Parameter	Symbol	Value
Selection operator		Tournament
Crossover operator		BLX-α
Mutation operator		Uniform
Max. number generations	N_{gen}	750
Min. number of generations	N_{min}	75
Threshold	thresh	0.05%
Population size	N_{ind}	400
Tournament size	tournsize	5
Blend α	α_{xvr}	0.2
Crossover prob.	p_{xvr}	0.95
Mutation prob.	p_{mut}	0.05
Mutation ind. prob.	p^{i}_{mut}	0.15

Chapter 4

Simple Case Study

In this section, we provide the parameters used for the two case studies analyzed in this paper: a notional GEO satellite located at (0∘, 25.8∘E) with 37 spot beams (similar to the scenario analyzed in [12]), and a realistic case based on Viasat1, a 72-spot-beam high-throughput satellite (HTS) covering North America.

4.1 37-beam GEO Satellite

Table 1 provides a summary of the parameters required for Eqs. 1-6. Atmospheric attenuation was not considered in this scenario to better reproduce the results in [12]. Figure 6 displays the footprint created by the satellite's beams. A four-color frequency reuse scheme (two bandwidth bands + dual polarization) is assumed. In the figure, beams assigned colors red and green use left-hand circular polarization (LHCP), while beams assigned colors yellow and blue use right-hand circular polarization (RHCP). In terms of bandwidth assignment, beams with different polarization's are independent, whereas adjacent beams that share the same polarization can trade bandwidths. Figure 7 shows an example: a red beam (e.g., beam 24) can increase its bandwidth if the green beams that are adjacent (beams 23 and 25) have their bandwidths reduced (as otherwise strong interference would occur, since adjacent beams use overlapping frequency intervals).

Table 1: **Link-budget and problem parameters for the 37-beam satellite case study.**

Parameter	Symbol	Value	Unit
Satellite altitude	h	35,786	km
Number of beams	n_{beams}	37	-
Payload power	P_{tot}	2350	W
Maximum power per beam	P_b^{max}	100	W
Central frequency	f	20	GHz
Total bandwidth	B_{tot}	375($\times$2 pol)	MHz
Tx antenna gain	G_{T_x}	52.2	dB
Tx antenna diameter	D_{T_x}	2.4	m
Output back-off	OBO	5	dB
Roll-off factor	α_r	0	-
Satellite EIRP	EIRP	63	dbW
Free-space path losses	FSPL	212	dB
Rx antenna gain	G_{R_x}	41.5	dB
Rx antenna diameter	D_{R_x}	0.7	m
System temperature	T_{sys}	211	K
LNB's noise figure	F	2.34	-
LNB's voltage standing wave ratio	VSWR	1.2	-
Carrier to adjacent satellite interference	CASI	28	dB
Carrier to cross polarization interference	CXPI	30	dB
Carrier to 3rd order intermodulation interference	C3IM	27	dB

```
>>> import math
>>> from scipy import constants
>>> def getSignalNoiseRatio(PTx, OBO, GTx, GRx, Tsys, L):
...     '''
...     linkBudget - C/N0 = PTx - OBO + GTx + GRx - L - 10*log10(k*Tsys)
...     L = FSPL + Latm + LRFTx + LRFRx
...     PTx - transmitted power (dB)
...     OBO - power-amplifier output back-off
...     GTx - Transmitting Antenna Gains (dB)
...     GRx - Receiving Antenna Gains (dB)
...     Tsys - System Temperature (K)
...      L - sum of losses (dB)
...     '''
...     k = constants.k
...     linkBudget = PTx - OBO + GTx + GRx - L - (10*math.log(k*Tsys,10))
...     return linkBudget
...
>>> getSignalNoiseRatio(2350, 5, 52.2, 41.5, 211, 212)
2432.0563426202407
>>>
```

Figure 4.1: Signal To Noise Ratio Calculation

```
>>> def getBeamDataRate(BW, Ar, spectralEfficiency):
...     '''
...     Finally, the beam data rate is computed as
...     Rb = BW/(1 + αr)*T(Eb/(N + I)) [bps],
...     where αr is the roll-off factor, and Γ is the spectral efficiency
...     of the modulation and coding scheme (MODCOD) (bps/Hz),
...     which depends on the Eb/(N + I)
...     As shown, the spectral efficiency (T) is a function
...     of the power: higher powers yield higher ratios of energy
...     per bit to noise power spectral density
...     '''
...     Rb = BW/(1+Ar)*spectralEfficiency
...     return Rb
...
>>> getBeamDataRate(750, 0, 1)
750.0
>>> getBeamDataRate(750, 0, 2)
1500.0
>>> getBeamDataRate(750, 0, 3)
2250.0
```

Figure 4.2: Beam Data Rate Calculation, Assuming spectral efficiency 1-3 bps/HZ

```
>>> def getLink(SNR, BW, Rb, CASI = 1, CXPI = 1, C3IM = 1):
...     '''
...     x -> C/(N0 + I)= (1/CABI + 1/CASI + 1/CXPI + 1/C3IM + 1/SNR)**-1
...     y -> Eb/(N + I)= C/(N0 + I)*(BW/Rb)
...     where Rb is the link data rate (bps) , and BW
...     is the bandwidth allocated to that beam (Hz). Notice how
...     our link budget equation considers four different types of
...     interference (CABI, CASI, CXPI, and C3IM). In above Eqs all terms
...     are in linear scale.
...
...     SNR -> Signal to Noise Ratio = C/N0
...     BW -> Bandwidth allocated to Beam (Hz)
...     Rb -> link data rate (bps)
...     '''
...     x = 1/(1/CASI + 1/CXPI + 1/C3IM + 1/SNR)
...     y = x*BW/Rb
...     return y
...
>>> getLink(2432.056, 750, 750, 28, 30, 27)
9.390038954870041
>>> getLink(2432.056, 750, 1500, 28, 30, 27)
4.695019477435021
>>> getLink(2432.056, 750, 2250, 28, 30, 27)
3.1300129849566805
```

Figure 4.3: Signal To Noise Ratio Calculation Considering Effect Due to Interference