# Queue Management and Interference control for Cognitive Radio

**Pål Håland**

Master of Science in Electronics
Submission date: June 2009
Supervisor: Torbjørn Ekman, IET
Co-supervisor: Kimmo Kansanen, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

# Problem Description

Future wireless networks are envisioned to utilize new paradigms of spectrum reuse, i.e. cognitive radio. where radio frequencies can opportunistically be taken into use if no other use is detected. In such systems the mechanisms for interference control are of essential importance. On the other hand, delay and energy efficiency requirements call for intelligent data queue handling for transmission over fading channels.

In this work we study the use of stochastic control methods on the transmitter data queue to optimize the energy useage of an opportunistic user in a cognitive radio system. Radio sensors are used to monitor the interference levels generated by the opportunistic user, and control the opportunistic transmissions in order to not exceed the allowed average interference level.
The student will implement a system model in MATLAB and find its behavior through simulations.

Assignment given: 21. January 2009
Supervisor: Torbjørn Ekman, IET

# Contents

# Chapter 1

# Introduction

Today wireless systems take up more and more of the frequencies that are available. Most of them are licensed to mobile and telephone operators, other for high speed wireless internet. While new wireless systems are developed there seems to be a lack of free frequencies, and most of the unlicensed frequencies is already packed with users.

These high demands on frequencies require the new system to build upon new types of technology. One of them called cognitive radio, which allow the re-use of spectrum. This technology takes in to account other users on the network, the spectrum and channel state. The idea of cognitive radio was first introduced by Joseph Mitola III in 1998, and the term cognitive radio identifies (from [1]) the point at which wireless personal digital assistants (PDA) and the related networks are sufficiently computationally intelligent about radio resources and related computer-to-computer communications to:

a) detect user communications needs as a function of use context, and

b) to provide radio resources and wireless services most appropriate to those needs.

What we are interested in is a spectrum agile radio. We use a more simple approach by re-using the spectrum and implementing a form of interference control too keep the average received level at a constant level.

One problem with the cognitive radio is that the secondary user do not know if any primary user that's in its range are already listening to another primary user outside the listening range. At this point the secondary user would just start transmitting and thus interfering with the primary user. One approach to protect the primary user is to set up a sensor that listens on the secondary user as well as the primary user, and we solve the problem with interference at the sensor. Since the secondary user does not know of the primary user transmitting, he thinks it is okay for him to transmit, thus interfering for the primary user. The sensor should then give feedback to the secondary user that it should lower the transmission power. At this point we need a data queue at the secondary user that control the arriving data, a power allocation method to allocate power based of the channel state between secondary user TX-RX and the interference history received from the sensor. We do this to keep the data rate between the secondary users high while trying to limiting the average received interference at the sensor. The and one interference queue at the sensor that just hold the interference size. If there are no primary users transmitting that the sensor senses, the transmission of the secondary user is okay.

The goal of the work is to maximize the data rate between the secondary transmitter and receivers, while limiting the interference with the primary user, using a sensor network. We will focus on three things, the data queue at the secondary transmitter, the power allocation and a virtual interference queue at the sensor. This virtual queue is modeled on the principles a ordinary queue, so that we can use the same stability criteria. In [2] Michael J. Neely develops a dynamic control strategy for minimizing energy expenditure in time varying wireless networks. The queue and power management were used in Neelys paper, but there he used a virtual power queue to control the average transmit power at each node. Where his strategy were to operate without knowledge of traffic rates or channel statistics, we tries to develop a way to maximize the data throughput using a sensor to rely feedback to the secondary user. Here we will implement an virtual interference queue at a sensor instead of using a local virtual power queue, that will submit feedback to the secondary user about the interference level it receive.

# Chapter 2

# Background theory

There are a couple of assumptions made in this report. The first assumption is that both the sensor and primary user that is receiving are in close proximity to each other. By looking at the power level received at the sensor, we can determine if the secondary user is degrading any signal that the primary user is trying to receive. This is only possible if we assume that the channel $S_S$ is ergodic, and have the same statistics between $TX_{SU_1}$ and $RX_{PU}$ as well as $TX_{SU_1}$ and $RX_S$. This assumption does not hold if the transmitters and receivers are stationary, and the channel is no longer ergodic. If that happens, it is possible to enforce an ergodic channel by using multiple transmit antennas, or if the secondary is moving relative to the sensor and primary user.

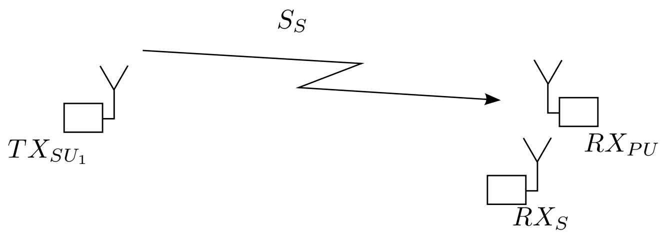

Figure 2.1: Display of sender and transmitter

Another assumption is based on the propagation environment, we assumed that the channel is located in a high density area, where there are a lot of objects creating many paths (multipaths) from the transmitter to receiver. In these environments you typically do not have any form of line of sight (LOS) component, and we assume that the channel in these areas has a Rayleig fading distribution. This channel have an exponential probability density function (PDF) of the squared envelope, because it exist of a multipath that distort the phase and quadrature component. By knowing this we can emulate a channel that have the same statistical probabilities. For further reading of the Rayleigh fading channel you can read [3, Ch:3.2].

Since it is the sensor that give feedback to the secondary user about interference, and we assume that the sensor and primary user is close to each other, then it is necessary to know that the channel statistics. If we cannot assume the same statistics on the propagation environment, then we cannot assume that the sensor and primary user have the same signal strength, and the interference measured might not be the same as at the primary user.

For other parts of the system we use the assumption that the flow of data in to the queue is less then the data capacity through the channel between $TX_{SU_1}$ and $RX_{SU_2}$. If the flow in is larger than the flow out, it is possible to force this assumption by utilizing package dropping. The channel capacity can be calculated using the techniques in[3, 4].

# Chapter 3

# Methods

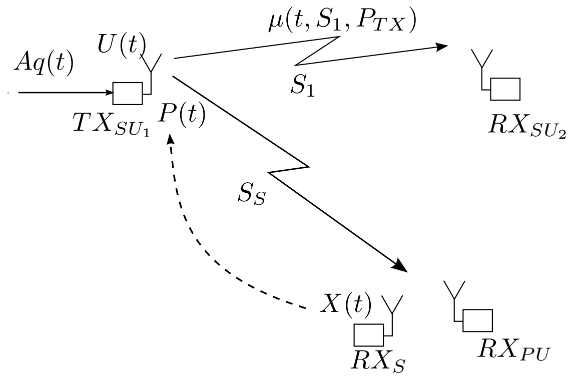

Figure 3.1: Sketch of the system principle

Our system design in figure 3.1 is designed to simulate the effects of using a sensor to limit the interference level created by communication between two secondary users, while trying to achieve a high data throughput between them. For this purpose you need to control the data flow in to the system, and out of the system, thus creating a data queue where the data that arrives should be queued before transmitting. Controlling the interference means that you would need a form of transmit power control, that control the power to a level where it do not interfere with the primary user. Here you need to spend as much power as possible without interfering with a primary user to acheive a high data throughput. Thats when the sensor should do its purpose, creating an interference queue that keeps track of the interference level from the secondary user. By assuming that the channel between the secondary user and the sensor share the same statistics as the channel between the secondary user and primary user, we can use the interference data from the sensor to control the secondary user transmit power.

The thought is that a lot of sensors is placed randomly around a location, and would probably be close to a primary user. If the sensor senses a secondary user transmitting, it would start monitoring that user, and give feedback to it in case it start interfering too much. Since the sensor and primary user are not located at the same place, you have to use the assumption that the channel between $TX_{SU_1} - RX_S$ and $TX_{SU_1} - RX_{PU}$ have the same statistics. With the assumption that the sensor and primary user have the same statistics, we can assume that the primary user have the same average received power as the sensor. Our approach is to create an interference queue at the sensor, and then stabilizing it. If we manage to stabilize the interference queue, we could stabilize the interference that the secondary user cause. This is done by lowering the transmission power at the secondary user, using data from the interference queue at the sensor.

## 3.1 Different functions of the system

The different functions of the system in figure 3.1 are as follows:

- the data queue $U(t)$ - Is the transmit buffer for the transmitter $TX_{SU_1}$, and hold the size of the data queue at a given time. It has the arrival process $Aq(t)$ as input, and depends on the transfer rate function $\mu(t, P, S)$ as output.
- the arrival process $Aq(t)$ - Models the random arrival of data from the upper transportation layers in to the $TX_{SU_1}$.
- the channel state $S(t)$ - is the channel state between the secondary users and sensor.
- rate function $\mu(t, P, S)$ - is the function that calculate how much data is transmitted from the $TX_{SU_1}$ to the $RX_{SU_2}$, depending on the transmit power $P(t)$ and $S_1$.
- interference queue $X(t)$ - is at the receiving sensor $RX_S$ and holds the interference, and it depends of $S_S$ and $P(t)$.
- power allocation function $P(t)$ - Implemented in $TX_{SU_1}$ and uses $X(t)$ from the sensor, but need to operate if there is no received interference from the sensor.

### 3.1.1 How the queue is handled

The queue is handled by subtracting amount transferred from the size of the queue then adding a random arrival $Aq(t)$. By storing the queue state at every time instant, we can find the approximate of how long data is stored before transmitted, by taking the mean of the queue size. The next value of the queue size is calculated with the following equation:

$$U(t+1) = \begin{cases} U(t) - \mu(t, P(t), S_1(t)) + Aq(t) & \text{if } U(t) \geq \mu(t, P(t), S_1(t)) \\ Aq(t) & \text{else} \end{cases} \tag{3.1}$$

The queue would be stable if and only if the mean arrival is less or equal to mean transmission data.

### 3.1.2 The arrival process

The idea of the queue arrival process is to simulate random arrival to be transmitted. The current setup of the queue arrival is:

$$Aq(t) = \begin{cases} n \text{ bits} & \text{if } k < p_R \\ 0 \text{ bits} & \text{if } k \geq p_R \end{cases} \tag{3.2}$$

where $n \in [1, N]$ and $k \in [0, 1]$ is a random number with uniform distribution. The arrival rate cannot be higher than $N \cdot p_R \leq$ instantaneous Shannon capacity for the channel between the two secondary users, where $p_R$ is the probability of receiving data to the queue and $N$ is max amount of bits that can arrive. This limitation is set because of the assumption that the flow in to the system should be less or equal to the flow out of the system.

### 3.1.3 Channel state

The channel state is a description on how good the signal from the transmitter can be "read" at the receiver. To generate the channel state, the Matlab function *random('exp',<dimension>·<mean>)* was used.

### 3.1.4 The transfer function $\mu$

The transfer functions purpose is to calculate how much data we manage to transmit over the channel, given a channel state $S(t)$ and the transmit power $P(t)$ calculated in section 3.1.6. It origins from Shannon's capacity theorem, found in [5, Ch:9.10]

$$\mu(t, P(t), S(t)) = \log_2(1 + P(t) \cdot S_1(t)) \tag{3.3}$$

where $P(t)$ is the transmit power and $S_1(t)$ is the channel state between the secondary $TX$ and $RX$.

### 3.1.5 The virtual interference queue

By using the sensor to update the interference level, and returning the value to the secondary user. In [2] a similar virtual power queue was used to enforce a transmit power constraint. Here we hope to stabilize the virtual interference queue to stabilize the interference. If this is possible the hope is to get as much throughput to the secondary receiver with stable interference to the primary user.

$$X(t+1) = \begin{cases} X(t) - X_{av} + P(t) \cdot S_s(t) & \text{if } X(t) > X_{av} \\ P(t) \cdot S_s(t) & \text{if } P(t) \cdot S_s(t) > X_{min} \text{ and } X(t) < X_{av} \\ X_{min} & \text{if } P(t) = 0 \text{ and } X(t) < X_{av} \end{cases} \tag{3.4}$$

The interference queue's next value is updated every time instant, and depends on the transmit power $P(t)$ from the secondary user, the channel state $S_s(t)$ between $SU_1$ and sensor and the average interference $X_{av}$. The average interference is how much interference is allowed to be over an average. This value can be tuned to the interference level you allow at the sensor, but would also change the data throughput between the $TX_{SU}$ and $RX_{SU}$. A high $X_{av}$ allow for higher data throughput, and a low value for less data throughput. The $X_{min}$ control maximum power level, if set to 0 would make it possible for the transmit power to become infinity. If we manage to stabilize the interference queue, we should be able to stabilize the interference done to the primary user/sensor. The interference queue can only be stable if the average received $P(t) \cdot S(t)$ is less than $X_{av}$.

### 3.1.6 Transmit power allocation

The goal of the transmit power allocation is to get an adaptive system, one that tries to transmit as much data as possible on good channel states. The system depends on the channel state between the secondary TX and RX, the data queue size and the interference registered at the sensor. The interference queue explained in section 3.1.5, transmit the interference level back to the $TX_{SU}$ so that the value can either increase or decrease the power level. The power allocation is done the following way:

$$P(t) = \begin{cases} P_{\text{peak}} & \text{if } P_{\text{peak}} \leq P(t) \\ \frac{2 \cdot U(t)}{X(t)} - \frac{1}{S_1(t)} & \text{if } 0 < P(t) < P_{\text{peak}} \\ 0 & \text{if } P(t) \leq 0 \end{cases} \tag{3.5}$$

As you can see in equation 3.5 it allocate the power to use based on the interference queue $X(t)$ explained in equation 3.4. Both of the equations have a channel state, the power allocation use the current channel state $S_1$, and the interference queue use the channel state $S_S$. The power allocation is also part of the water filling solution $\left(\frac{1}{\gamma_0} - \frac{1}{\gamma_D}\right)$ from [3], but here the threshold $\gamma_0 = \frac{X(t)}{2(U(t)}$ and would change depending on how large the data queue or interference queue changes.

## 3.2 The capacity of the system

To figure out where to limit and start the simulations, we have to determine the capacity of a Rayleigh fading channel with an average signal to noise ratio (SNR) of 1. We assume we have an average power of 1 and that the channel is slowly varying. The capacity of the system were then calculated numerical using Newtons method, to find a unique $x_0$ so that $f(x_0) = 0$ using formula (13):

$$f(x) = \frac{e^{-x}}{x} - E_1(x) - \bar{\gamma} \tag{3.6}$$

from [4], where $x = \frac{\gamma_0}{\bar{\gamma}}$, $E_n(x)$ is the exponential integral of order $n$, $\gamma_0$ is the SNR at the Shannon capacity and $\bar{\gamma} = 1$ is the average carrier-to-noise (CNR). Using Newton's method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.7}$$

where $f'(x) = e^{-x}$ is the derived of $f(t)$. Computing a new $x_{n+1}$ trying to get as close as possible to 0. When we find a value that we are happy with, we use this in equation (16) from [4] which is

$$\frac{C_{opra}}{B} = \log_2(e) E_1(\frac{\gamma_0}{\bar{\gamma}}) \tag{3.8}$$

and we can find the $C_{opra}$ for the specific $\gamma_0$.

# Chapter 4

# Results

By running the Matlab scripts listed in appendix A, written with the help of the formulas in chapter 3. It is possible to get different simulations of how a sensor network would control the transmission of a secondary user, and how much it would interfere with the primary user. Varying one and one of the parameters used in the simulation it is possible to see how it affects the entire system.

## 4.1 The arrival rate

By adjusting the arrival rate to the queue, up to the theoretic capacity of the system, we can see where the system starts being ineffective. The simulations in this section are done with an average interference level $X_{av} = 1$, minimum interference level $X_{min}$ is set to be a tenth of $X_{av}$. The probability of receiving any bit $n$ to the queue $p_R = 0.1$.

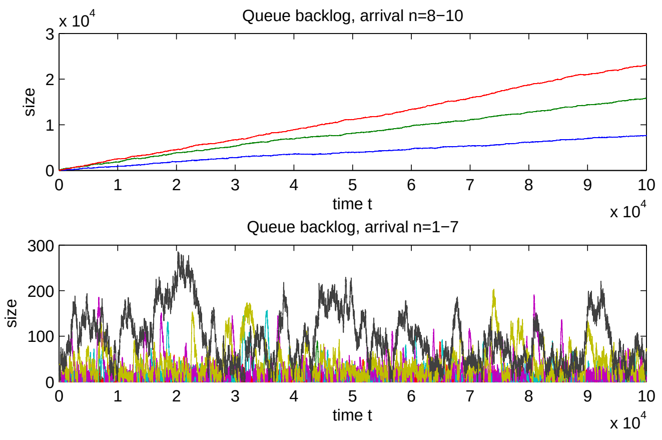

Figure 4.1: Queue backlog, $n = 1 - 10$ bit, $p_R = 0.1$, $X_{av} = 1$, $X_{\min} = 0.1$ and $P_{\text{peak}} = \text{Inf}$

Figure 4.1 show the result for this simulation, where the upper plot is for $8 - 10$ bit arrival, and the lower plot is for $1 - 7$ bit arrival. The blue graph is for 8 bit arrival, the green is for 9 bit arrival, the red i is for 10 bit arrival and the grey on the lower plot is for 7 bit arrival with a 10% chance.

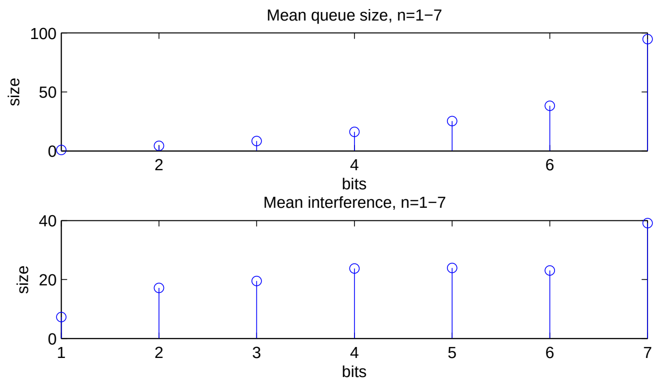

Figure 4.2: Mean queue backlog, $n = 1 - 7$ bit, $p_R = 0.1$, $X_{av} = 1$, $X_{\min} = 0.1$ and $P_{\text{peak}} = \text{Inf}$

By looking at the plot in figure 4.1 we can see that the system is not able to stabilize when the arrival rate is higher than 7 bits. A closer look at the mean size of the data queue and interference can be found in figure 4.2. On the lower arrival rates the $P_{\text{peak}}$ peaks at $150 - 200$, but averages at 1 because of less transmission instances. The plot of the queue size interesting because it show when the transmission rate is less than the arrival rate, at this point the queue just start growing. The mean queue size also tell us how long it is expected for data to stay in the queue before it is transmitted, meaning the delay of the system. One of the reasons that the transmission do not reach the Shannon capacity is that we have a moving threshold at the power allocation. When we have interfered at the sensor, it sends the interference back to the secondary transmitter. If the interference is high compared to the queue size, then the threshold for transmission is high, and some good channel states are wasted. One way to solve this problem is including a peak power limitation $P_{\text{peak}}$, so that the interference queue do not peak as high.

### 4.1.1 Arrival rate below capacity

Figure 4.3 and 4.4 were both simulated with a constant arrival rate between 0 and 1 bit to see the effect on low arrival rates. As you can see from the stem in figure 4.4 the transmission rate does not become steady before an arrival rate around 0.05 bit. This is because of equation 3.5 where the size of $\frac{2 \cdot U(t)}{x(t)}$ has to be larger than $\frac{1}{S_1}$. It is part of the water filling solution, where you would spend less power on bad channels and more power on good channels. The second change in the transmission rate is where the transmission rate become less than the arrival rate. This will be discussed later in the results.

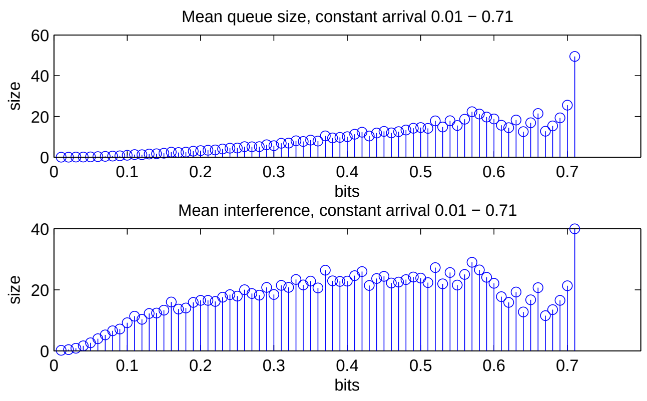

Figure 4.3: Mean of the queue size and interference for increasing constant arrival size up to 0.71 bit.

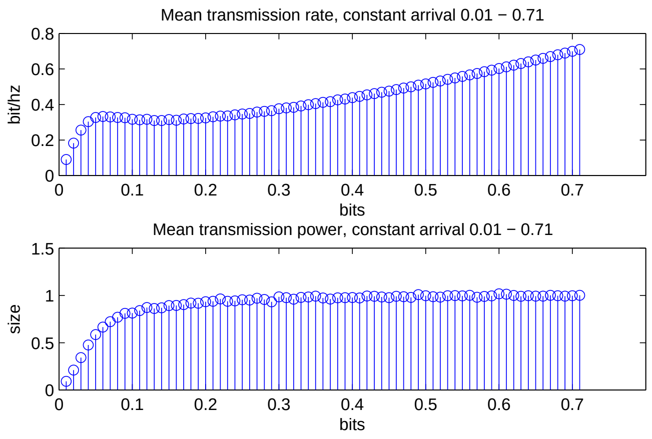

Figure 4.4: Mean of the transmission size and transmission power for increasing constant arrival size up to 0.71 bit.

While looking at a zoomed in version of the queue and interference of the stable part of the system in figure 4.5, where the arrival rate is less than 0.71 bit. We can see in the beginning of the plot that when the interference level reaches a minimum of around 0.1, and when the queue is larger, and there is a good channel. Then the interference suddenly peaks because it uses this good channel to transmit data. While doing this it prevents new transmission trough good channels because it is has changed the threshold because of high interference level and a small queue. This stops the transmissions on good channels, and happens because of the power estimation in equation 3.5.

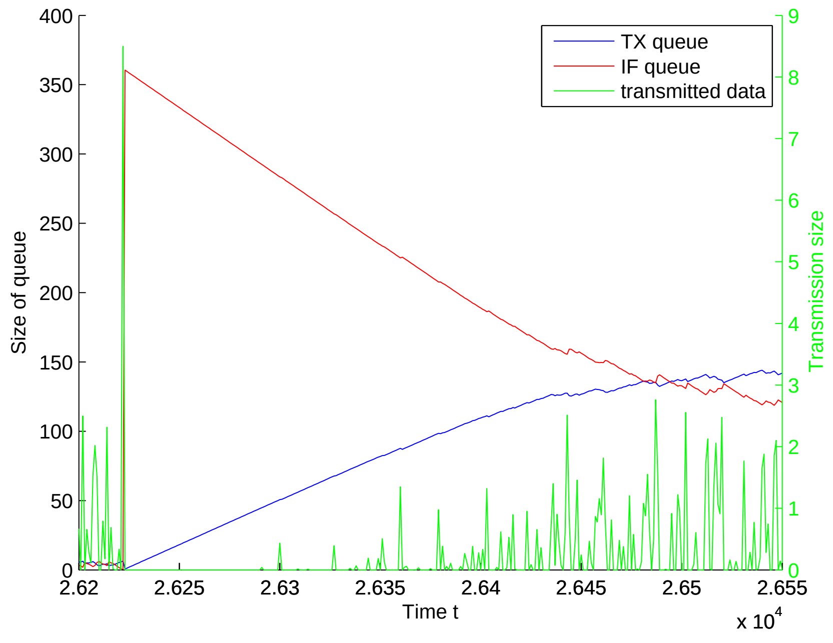

Figure 4.5: A cut of the queue size and interference level, blue is queue size and red is interference level (green is transmission size).

By zooming out a bit we can see that this is a trend in figure 4.6. The interference level will be less than the queue until it hits a level where $\frac{2 \cdot U(t)}{X(t)} > \frac{1}{S_1(t)}$. When this happens the interference level will be very large compared to the queue size, and the transmit power will be low until the interference level get below the queue size. The queue size will continue growing because of an average arrival rate higher than the transmission rate until the interference level is less that the queue size. Then the transmitter will start transmitting more often, and average transmission rate will be higher than the arrival rate again. This continues until the interference queue hit a minimum, and peaks up to a size a lot larger than the queue size. This problem could be solved by using a $P_{\text{peak}}$ limitation, spreading the transmissions over time instead of everything on a good channel.

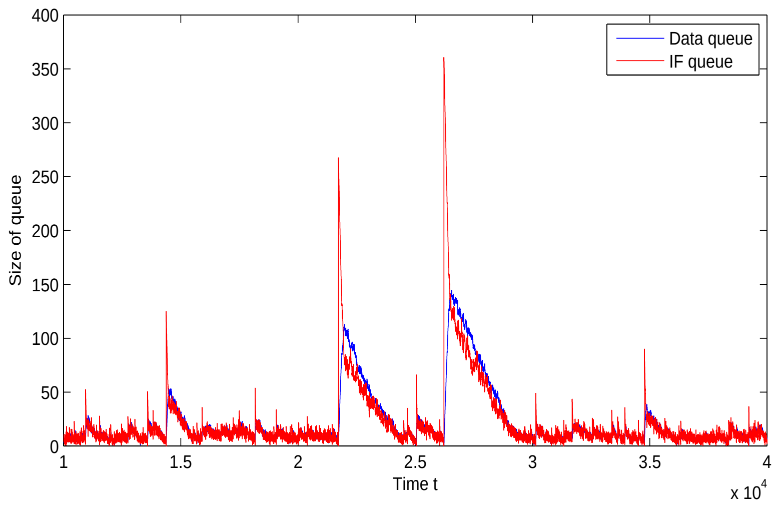

Figure 4.6: A larger cut of the queue size and interference level, blue is queue size and red is interference level.

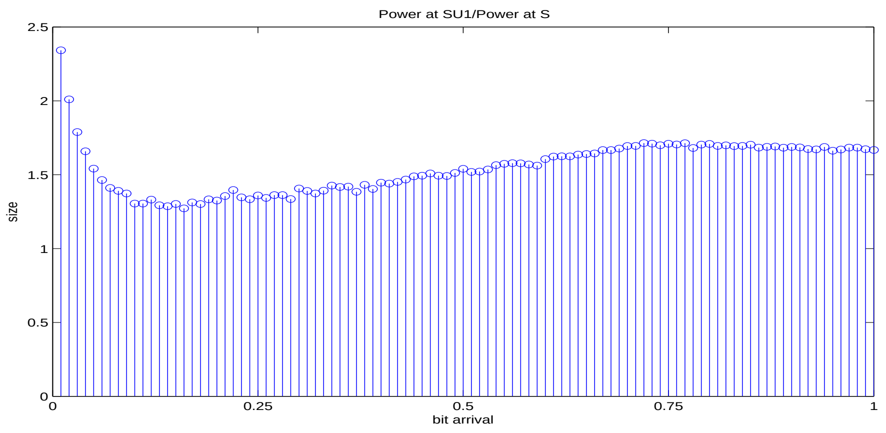

Figure 4.7: Factor between received power at sensor and secondary user - $\frac{mean(P_{SU_1})}{mean(P_{S_1})}$.

From figure 4.7 we see that power received at the secondary user is higher on an average then at the sensor. This show that the sensor network can do it's purpose to keep a higher power at the secondary user compared to at the sensor (and the primary user). One interesting thing in the figure, is that at lower arrival rates, the system give a higher ratio. this could be because the queue has to grow larger before transmitting, and transmits only on good channels.

### 4.1.2 Arrival rate above capacity

The best place to find out where it start going wrong when the arrival rate is higher then the transmission rate is at the beginning of the data and interference queue. Looking at figure 4.8 it show the same thing that happened for an arrival rate less then the transmission rate. The interference queue make a peak after it had gone to a minimum with a good channel. When the interference queue gets below the size of the data queue, the interference queue will start increase again. At this point the arrival rate is higher then the transmission rate, and the data queue will continue to grow, even though the system continue to transmit data. This causes the interference queue to always be less than the data queue.

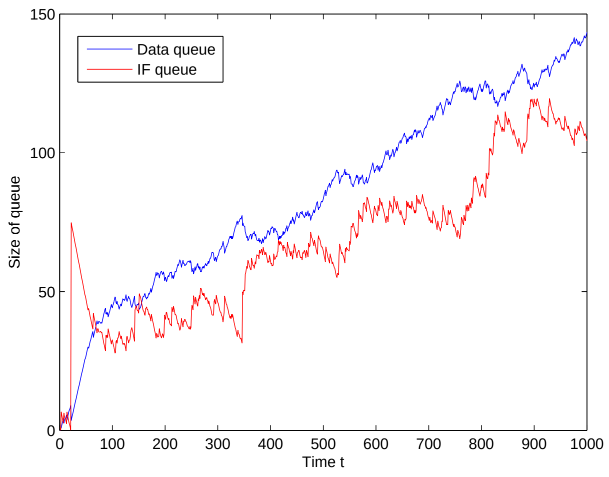

Figure 4.8: Arrival rate higher than the transmission rate, arrival rate is 0.84 bit pr time instance

To see if there is any change in the trend of an ever growing queue, I try to run a simulation increasing the bit arrival from $n = 1$ to $n = 100$ with a $p_R = 0.1$. Figure 4.9 show the mean transmission and power.

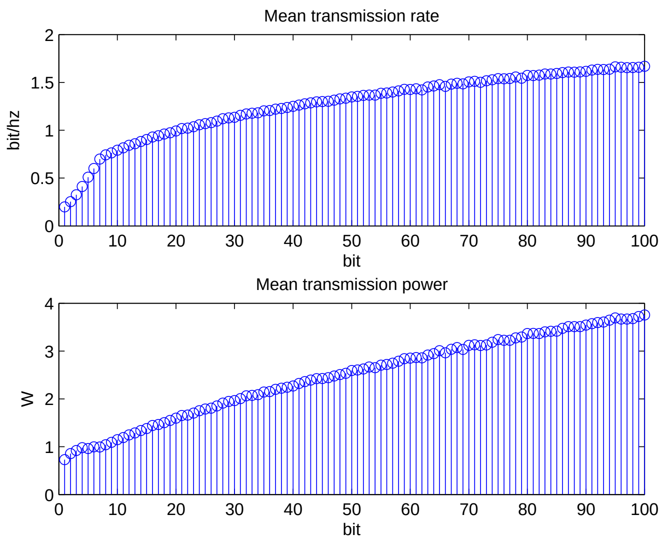

Figure 4.9: Mean of the transmission size and transmission power for increasing arrival size.

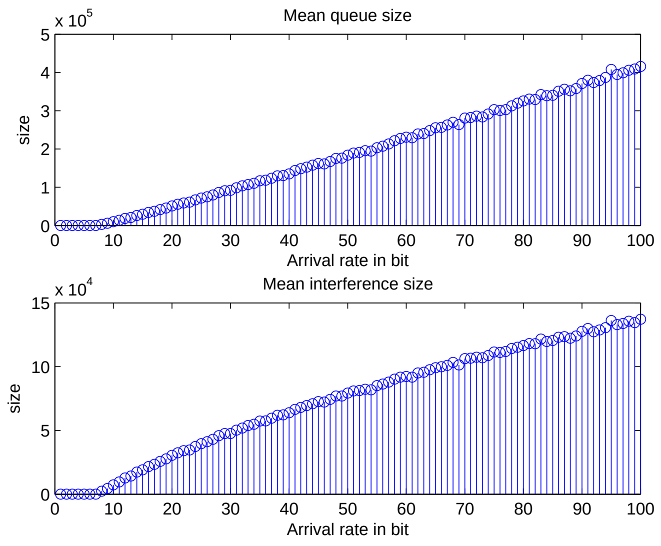

Figure 4.10: Mean of the queue size and interference size for increasing arrival size up to 100 bit and 10% chance of arrival.

Looking figure 4.10 you can see that the queue size is a lot larger than the interference size. If you use a much larger number in to equation 3.5 for the queue size, it would increase the transmission power. As the arrival rate continue to grow, the transmission rate continue to grow as well at the expense of increased average power. By looking at the equation 3.5 for power allocation, we see that this happens because $U(t)$ grows quicker than $X(t)$, but the ratio between $\frac{2U(t)}{X(t)}$ seem to stabilize at 2.5 as seen in figure 4.11. When this happen the primary user

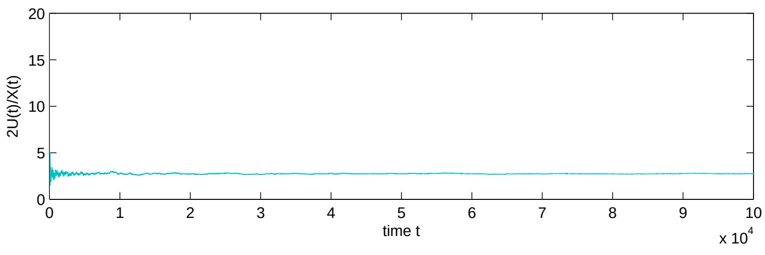

Figure 4.11: The factor $\frac{2U(t)}{X(t)}$ as the time increase. Arrival rate higher than transmission rate.

will transmit when the factor $\frac{1}{S_S}$ is smaller than 2.5, and it depends only on the channel if the secondary user transmit, because the factor between $\frac{2U(t)}{X(t)}$ stays the same.

## 4.2 Average received interference $X_{av}$

The average received interference $A_{av}$ that the sensor operate from adjust how much interference the sensor should tolerate. For the first simulation I set the arrival rate to be the same as the transmission rate when the average power is equal to one. We can see from figure 4.12 that the system will not be stable before $X_{av} = 1$, but the mean queue size decrease for increasing $X_{av}$. Under this simulation the arrival rate were adjusted to the average transmission rate 0.71 with an average power of 1.

By changing the arrival rate to be in the stable region, to an arrival rate of 0.4 bit/s, we can simulate how changing the $X_{av}$ affects the data queue and average transmission for increasing value of $X_{av}$. While increasing it, we can see from figure 4.13 that the system is starts getting stable after passing an average interference above 0.4.

By fixing the average interference $X_{av}$ to be 0.4, and then simulating over different arrival rate, we can see that the factor between $P_{SU_1}$ and $P_S$ from figure 4.14, have increased compared to figure 4.7. This is because the $TX_{SU}$ would wait for a good channel to transmit because of the interference level done to the sensor.

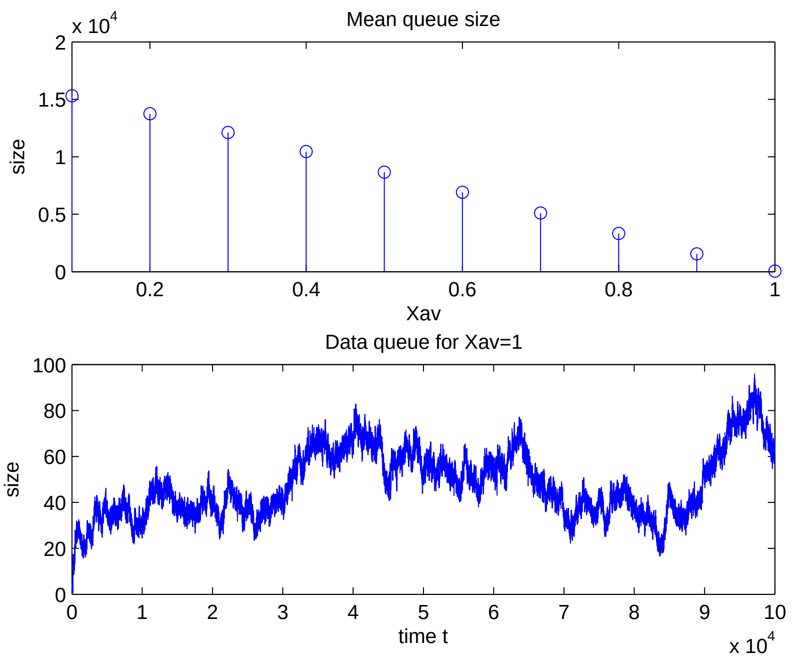

Figure 4.12: The mean queue and interference for increasing $X_{av}$, fixed arrival rate of 0.71

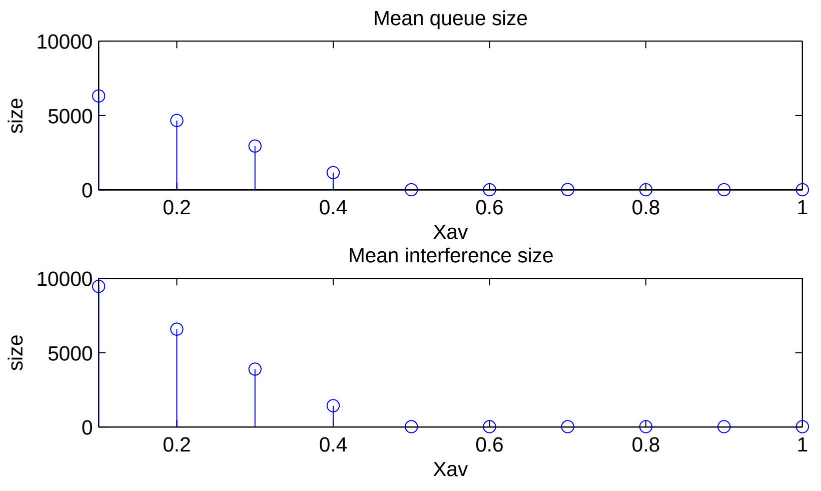

Figure 4.13: The mean queue and interference for increasing $X_{av}$, fixed arrival rate of 0.4.

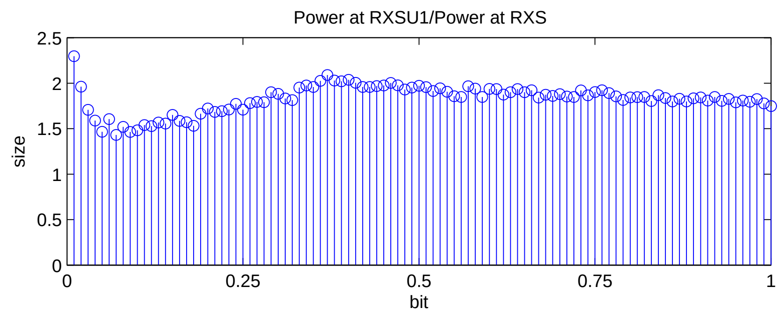

Figure 4.14: Factor between received power at sensor and secondary user - $\frac{mean(P_{SU_1})}{mean(P_{S_1})}$.

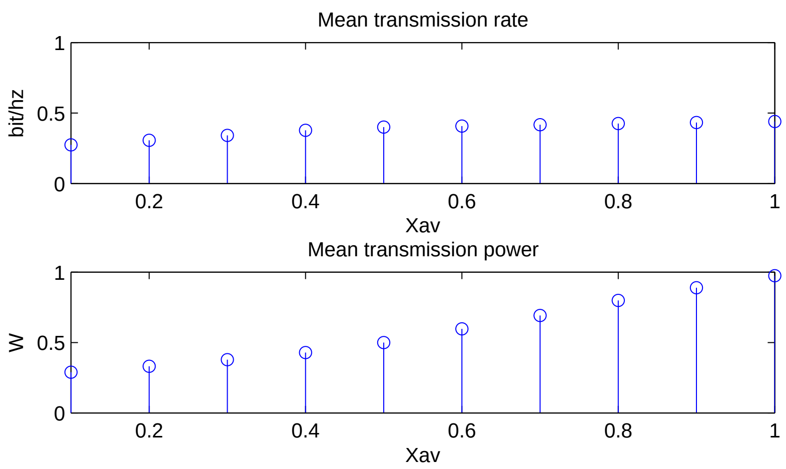

Figure 4.15: The transmission rate and power.

## 4.3 The peak power limit

Changing the power limit $P_{\text{peak}}$ equal to $X_{av} = 1$ makes a huge change to the system. Looking at figure 4.16 we see that the queue size is actually a lot less at an arrival rate of 0.8 compared to no peak power constrain, but this causes the system to transmit with an average power of almost 1, as we can see from figure 4.17, causing the system to have the same signal strength at the primary user/sensor as it has with the $RX_{SU_2}$. Using a $P_{\text{peak}} > X_a v$ will make sure that the link between the secondary users always have a higher strength then the link between $TX_{SU_1}$ and $RX_S$. If the data queue grows a lot larger than the interference queue, the system will transmit with a peak power, no matter what the channel state is.

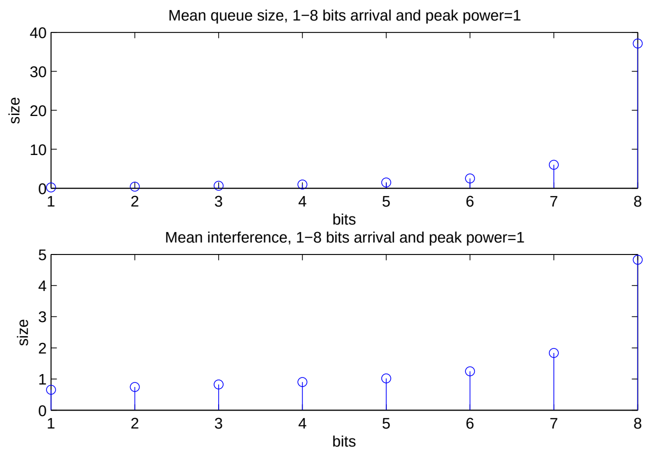

Figure 4.16: The mean data and interference queue with a $P_{\text{peak}} = 1$ and $X_{av} = 1$.

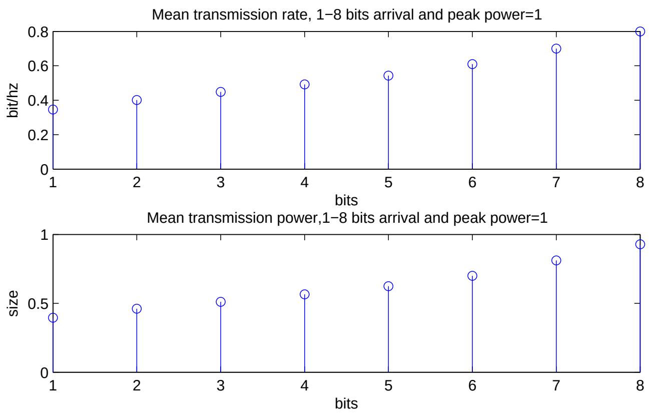

Figure 4.17: The mean transmission size and power with a $P_{\text{peak}} = 1$ and $X_{av} = 1$.

Chapter 5

# Conclusion and Discussion

From the results of the simulations we have that using an opportunistic transmission mode gives a higher received power at the secondary receiver. Using the sensor implementation helped control the power level at the secondary transmitter, with an average received interference of $X_{av} = 1$ at the sensor. One of the problems occurred was that we could not achieve close to the theoretical capacity of the channel. Much of this was because of the interference constrain in the power allocation, caused good channels between the two secondary users go to waste because of a high interference level at the sensor. Setting a peak power limitation would spread the transmissions and keeping a low interference at the sensor. This would stop the virtual interference queue from peaking, and created the interference queue more stable, allowing a higher arrival rate. For higher arrival rate then the capacity of the system would cause the power to go in to constant transmission. For arrival rates higher than the capacity of the system, it is possible to use package dropping at the arrival process. But this would not cause any increase in capacity. Decreasing the $X_{av}$ decreases the transmit power from the secondary user, but would cause the data queue to become unstable as long as arrival rate were higher or equal to $X_{av}$. At lower arrival rates there was a peak in the factor between the power at secondary receiver and the power at the sensor.

For future work on this subject, it would be interesting to study more of the effects adjusting the peak power to achieve better efficiency. Implementing package dropping to keep the data queue stable.

# Appendix A

# Matlab code

# Listings

## A.1 Transfer function $\mu(t, S(t), P(t))$

Listing A.1: The transfer function

```
function retur = uTrans(t,P,S)
  global transm
  % B = bandwith, P = transmit power, S = channel state
  %B = 200000; % Hz
  %transm(t) = B*log2(1+P.*S);
  transm(t) = log2(1+P.*S);
  retur = transm(t);
end
```

## A.2 Arrival function $Aq(t)$

Listing A.2: The arrival function

```
function retur = aQueue(t,varargin)
  global Aq vilkaarlige
  k = 0.01;
  bit = 102;
  if (size(varargin,2) >= 2 )
    bit = varargin{1};
    k = varargin{2};
  elseif (size(varargin,2) == 1)
    bit = varargin{1};
  end
  tmp = vilkaarlige(t);
  if tmp > k;
    Aq(t) = 0;
  else
    Aq(t) = bit;
  end
  retur = Aq(t);
end
```

## A.3 Queue function $U(t)$

Listing A.3: The queue function

```
% The queue management, Utran updates the queue length U every time it is
% run, does not need the t. pTrans is supposed to be the variable power,
% used to minimize the interference at the primary user. s_Trans is the
% state of the link. uTrans is how much is transmitted trough the link. and
% aQueue is how much that has arrived to the queue.

function retur = Utran(j,u_temp,S,varargin)
  if size(varargin,2) == 1
    bit = varargin{1};
    k = 0.1;
  elseif size(varargin,2) >= 2
    bit = varargin{1};
    k = varargin{2};
  else
    k = 0.1;
    bit = 10;
  end
  p = pTrans(j,S(1,j),S(2,j));
  retur = max([u_temp-uTrans(j,p,S(2,j)),0])+aQueue(j,bit,k);
  %retur = max([u_temp-uTrans(j,1,S(2,j)),0])+aQueue(j,bit,k);
end
```

## A.4 Channel state function $S(t)$

Listing A.4: The channel state function

```
function retur = s_Trans(varargin)
  MU = 1;
  hoyde = 1;
  bredde = 1;
  if (size(varargin,2) == 3)
    MU = varargin{1};
    hoyde = varargin{2};
    bredde = varargin{3};
  elseif (size(varargin,2) == 2)
    MU = varargin{1};
    bredde = varargin{2};
    hoyde = 1;
  elseif (size(varargin,2) == 1)
    MU = varargin{1};
    bredde = 1;
    hoyde = 1;
  end

  retur = random('exp',MU*ones(hoyde,bredde));
end
```

## A.5 Power allocation function $P(t)$

Listing A.5: The power function

```
function retur = pTrans(t,varargin)
  global U x xav Power metode P_peak Xmin
  S = [0 0];
  if size(varargin,2) >= 2
    S(1) = varargin{1};
    S(2) = varargin{2};

    prev = x(t);

    P_transm = 2*U(t)/prev-1/S(1);
    Power(t) = min(max(P_transm,0),P_peak);

    if (t == size(x,2))
      if strcmp(metode,'runs')
        x(1)=max(max(prev-xav,0)+Power(t)*S(2),Xmin);
      elseif strcmp(metode,'sample')
        x(1)=1;
      end
    else
      x(t+1)=max(max(prev-xav,0)+Power(t)*S(2),Xmin);
    end
  else
    Power(t) = 1;
  end
  retur = Power(t);
end
```

## A.6 Runtime environment

Listing A.6: The runtime environment

```
%% Variables
global x xav U S Aq Power transm averaging vilkaarlige metode P_peak Xmin
% An empty queue with the lenght of t transmissions;
t = 1000000; % How many runs
maxLength = 100000;
averaging = 100;
tids = zeros(1,ceil(t/maxLength));
l = 1;
%%%%%Changable vars variables%%%%%
k = 1; % The probability of arrival to the queue
bit =0.1; % The ammount arriving to the queue
increase = .1; % How much one of the variables should increase
xav = 1; % Should be mean SNR
Xmin = 0.1; % The minimum itnterference to use in powerallocation (also
  regulate max power)
P_peak = 1.5; % Maximum transmit power.

% runs or sample
metode = 'sample';
bitminav = 1; % 1 for change in bit, 2 for change in Xmin, 3 for change in Xav.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

c = clock;
```

```
if c(4) < 10
  tempTid = ['0' num2str(c(4))];
else
  tempTid = num2str(c(4));
end
if c(5) < 10
  tempTid = [tempTid '0' num2str(c(5))];
else
  tempTid = [tempTid num2str(c(5))];
end

savePath = strcat('simulering/',date,'--',tempTid,'/');



instilling = struct(...
  'savePath',savePath,'t',t,'maxLength',maxLength,...
  'averaging',averaging,'l',l,'k',k,'bit',bit,'P_peak',P_peak,...
  'xav',xav,'Xmin',Xmin,'metode',metode,...
  'bitminav',bitminav,'increase',increase);
if(~exist(savePath,'file'))
  mkdir(savePath)
end
save(strcat(savePath,'instilling.mat'),'-struct','instilling')
%save(strcat(savePath,'instilling.txt'),'instilling','-ASCII')

meanU = zeros(1,size(tids,2));
U = zeros(1,maxLength);
x = zeros(1,maxLength);
x(1) = 1;
S = zeros(2,maxLength);
Aq = zeros(1,maxLength);
vilkaarlige = zeros(1,maxLength);
Power = zeros(1,maxLength);
transm = zeros(1,maxLength);

Gl = 1; % is one when both receiver and transmitter
        % is omnidirectional

%% Runtime
while( t > 0 )
  tic
  if (t > maxLength)
    j = maxLength;
  else
    j = t;
  end
  S(1,:) = s_Trans(1,1,j);
  S(2,:) = s_Trans(1,1,j);
  vilkaarlige = rand(1,maxLength);
  for i=1:j;
    if (i == j)
      if tids(1) == 0
        saveFiles(savePath)
      else
        saveFiles(savePath,'-append')
      end
      if t-i == 0
      elseif strcmp(metode,'runs')
        meanU(l) = mean(U);
        U(l) = Utran(i,U(i),S,bit,k);
      elseif strcmp(metode,'sample')
```

```
        U(1) = 0;
      end
    elseif strcmp(metode,'runs')
      U(i+1) = Utran(i,U(i),S,bit,k);
    elseif strcmp(metode,'sample')
      U(i+1) = Utran(i,U(i),S,bit,k);
    end
    %U(i+1) = Utran(i,U(i),1);
  end
  tids(l) = toc;
  if t-j==0 && j < maxLength
    U(j+1:size(U,2))=0;
    x(j+1:size(x,2))=0;
    S(1,j+1:size(S,2))=0;
    S(2,j+1:size(S,2))=0;
    Aq(j+1:size(U,2))=0;
    transm(j+1:size(U,2))=0;
  end
  if (tids(2) == 0)
    disp(['Estimated time to run the simulation is ' tiden(tids(1)*(t/maxLength
      ))])
  else
    disp(['Esitmated time left is ' tiden(mean(tids(1:l))*(size(meanU,2)-l))])
  end
  t = t-j;
  l = l+1;
  if strcmp(metode,'sample')
    switch bitminav
      case 1;
        bit = bit+increase;
      case 2;
        Xmin = Xmin+increase;
      case 3;
        xav = xav+increase;
    end
  end
end

disp(['The time it took to run the simulation is ' tiden(sum(tids))])
```

## A.7 Capacity calculation of the channel

Listing A.7: Calculating the capacity of the channel

```
% Computing Copt
MU = 1;%0.5:0.001:1.0285;
hoyde = 1;
bredde = 1000000;
temp = 0;
%S = random('exp',MU.*ones(hoyde,bredde));
xnt = 0.5*ones(1,size(MU,2));
fx = 1*ones(1,size(MU,2));
Copra = zeros(1,size(MU,2));
P_out = [];
f_newton = @(xn) xn - (exp(-xn)/xn - expint(xn)-1)/(-exp(-xn)/xn^2);
f_newton2 = @(xn,gm) xn/gm - (exp(-xn/gm)/(xn/gm) - expint(xn/gm)-gm)/(-exp(-xn
  /gm)/(xn^2/gm));

```

```
f = @(gamma0,gammaavg) exp(-gamma0/gammaavg)/(gamma0/gammaavg) - expint(gamma0/
  gammaavg)-gammaavg;
fd = @(gamma0,gammaavg) exp(-gamma0/gammaavg)/(gamma0^2/gammaavg);

for i = 1:size(MU,2)
  n = 0;
  while fx(i) ~= 0 && n<1000;
    xnt(i) = f_newton2(xnt(i),MU(i));
    %xnt(i) = xnt(i)-f(xnt(i),MU(i))/fd(xnt(i),MU(i)); %f_newton(xnt(i));
    fx(i) = f(xnt(i),MU(i));
    n = n+1;
  end
  gamma0 = xnt(i);

  Copra(i) = log2(exp(1))*expint(gamma0/MU(i));
  P_out= [P_out 1-exp(-gamma0/MU(i))];
end


figure(1)
plot(Copra);
```

# Bibliography

[1] Joseph Mitola III. *Cognitive Radio - An Integrated Agent Architecture for Software Defined Radio.* PhD thesis, Royal Institute of Technology (KTH), 8 May, 2000.

[2] M.J. Neely. Energy optimal control for time-varying wireless networks. *Information Theory, IEEE Transactions on*, 52(7):2915–2934, July 2006.

[3] Andrea Goldsmith. *Wireless communications.* Cambridge University Press, Cambridge, 2005.

[4] M.-S. Alouini and A.J. Goldsmith. Capacity of rayleigh fading channels under different adaptive transmission and diversity-combining techniques. *Vehicular Technology, IEEE Transactions on*, 48(4):1165–1181, Jul 1999.

[5] Simon Haykin, S. *Communication systems / Simon Haykin.*

# Acronyms