MURI Network Layer Simulation and Optimization

Hua Wang

Department of Electrical Engineering and Computer Science

University of Michigan

Ann Arbor, MI 48109

September 22, 1999

Contents

1 Overview of Simulation-Based Optimization **2**

2 Network Layer Simulation **2**

2.1 Network Scenario 3

2.1.1 Drift Model 4

2.1.2 Broadcast Model 5

2.1.3 Propagation Model 5

2.1.4 Estimation Model 16

2.1.5 Energy Constraint 17

2.1.6 Optimization Problem Formulation 17

2.2 Modeling using *OPNET* software 19

3 Network Layer Optimization **19**

3.1 Simulated Annealing 20

3.2 Integration of Simulation and Optimization 21

4 Interaction Between Network Layer and Processing Layer **22**

5 Interaction Between Network Layer and Device Layer **24**

5.1 Antenna Model 24

6 Simulation Results **25**

7 Analysis of Simplified Network Scenario **26**

8 Proposed Power Adjustification **26**

9 Conclusions and Future Work **28**

List of Figures

1 Overview of Simulation-Based Optimization . . . 3

2 MURI Network Simulation Scenario . . . 4

3 Multiple Path From the transmitter to the Receiver . . . 5

4 Bit Error Rate as a Function of Distance . . . 8

5 Packet Error Rate as a Function of Distance . . . 8

6 Bit Error Rate as a Function of Distance . . . 9

7 Packet Error Rate as a Function of Distance . . . 9

8 Bit Error Rate as a Function of Distance . . . 10

9 Packet Error Rate as a Function of Distance . . . 10

10 Bit Error Rate as a Function of Distance . . . 11

11 Packet Error Rate as a Function of Distance . . . 11

12 Bit Error Rate as a Function of Distance . . . 12

13 Packet Error Rate as a Function of Distance . . . 12

14 E_b/N_0 as a Function of Distance . . . 13

15 E_b/N_0 as a Function of Distance . . . 13

16 E_b/N_0 as a Function of Distance . . . 14

17 E_b/N_0 as a Function of Distance . . . 14

18 E_b/N_0 as a Function of Distance . . . 15

19 Binary Phase Shift Keying (bpsk) BER . . . 15

20 Network Level Modeling . . . 19

21 Node Level Modeling . . . 20

22 Process Level Modeling . . . 20

23 MURI Network Simulation and Optimization Setup . . . 21

24 MURI Network Simulation and Optimization Using Parallel Structure . . . 22

25 Network Layer and Processing Layer Interaction . . . 23

26 Average Position Estimation Error versus Energy Constraint 26

List of Tables

1 **Symbols Used in Packet Error Rate Calculation** 6

2 **Statistics for Typical Network Simulation** 25

MURI Network Layer Simulation and Optimization

Abstract

The Multidisciplinary University Research Initiative (MURI) is a simulation and optimization study of future army communication system from a low energy perspective. The overall communication system can be partitioned into three different layers, namely, network layer, processing layer, and device layer. The ultimate goal of MURI is to achieve global optimization of these three layers and demonstrate the methodology of approaching problems of such large scale.

Local simulation and optimization is first applied at each layer individually and global optimization of all layers is achieved through the interaction of different layers by means of tables and functions.

At network layer in particular, we chose *OPNET* as our network simulation tool and *simulated annealing* as our optimizer. A very simple scenario has been implemented for the initial approach, where several mobile nodes, initially deployed in a drop zone, travel towards the same destination and broadcast position awareness packets under certain energy constraint. Each node gives estimates of all other nodes' positions periodically. The performance criteria of the system is chosen to be the root mean squared error between a node's estimated position and its real position, and is minimized over the parameter set (T, q, Ect, Ecr), where T is the interval between successive transmissions, q is the retransmission probability, Ect is the energy constraint for a packet at the transmitter, and Ecr is the energy constraint for a packet at the receiver.

We have got some preliminary results of performance versus energy constraint tradeoff, which agree with what we have expected.

Keywords: MURI, OPNET, Simulated Annealing

1 Overview of Simulation-Based Optimization

The Multidisciplinary University Research Initiative (MURI) undertaken at the University of Michigan is to investigate the various algorithms and protocols of future army communication systems from a low energy perspective. The system we are investigating may be partitioned into three layers, namely network layer, processing layer, and device layer. The ultimate goal of MURI is the global simulation and optimization of the overall communication system to achieve a performance versus energy consumption tradeoff that involves different layers altogether at the same time and the demonstration of methodology of approaching problems of such large scale.

To find a mathematical expression of performance measure as a function of parameters for the overall communication system is generally not possible, we therefore have to rely on the integration of simulation and optimization. Even so, it is still extremely hard to integrate simulation and optimization over all these three layers at the same time since it usually involves too huge dimension of freedom on the parameter space and it takes much too long to run simulation. As a consequence, an integration of simulation and optimization has to be taken at each individual layer first before the global optimization can be applied to the three layers to achieve the performance versus energy consumption tradeoff.

The overview of the integration of network layer simulation and optimization, and the interaction among network layer, processing layer and device layer is illustrated in Figure 1.

In the coming sections, more details of simulation-based optimization at the network layer will be given, along with its interaction with processing layer and device layer.

2 Network Layer Simulation

As with any simulation problem, we face the choice of simulation tool that we should use. We use *OPNET* package as our simulation tool mainly because *OPNET* is becoming dominant in industry and military. A very simple network scenario has been modeled in OPNET for optimization.

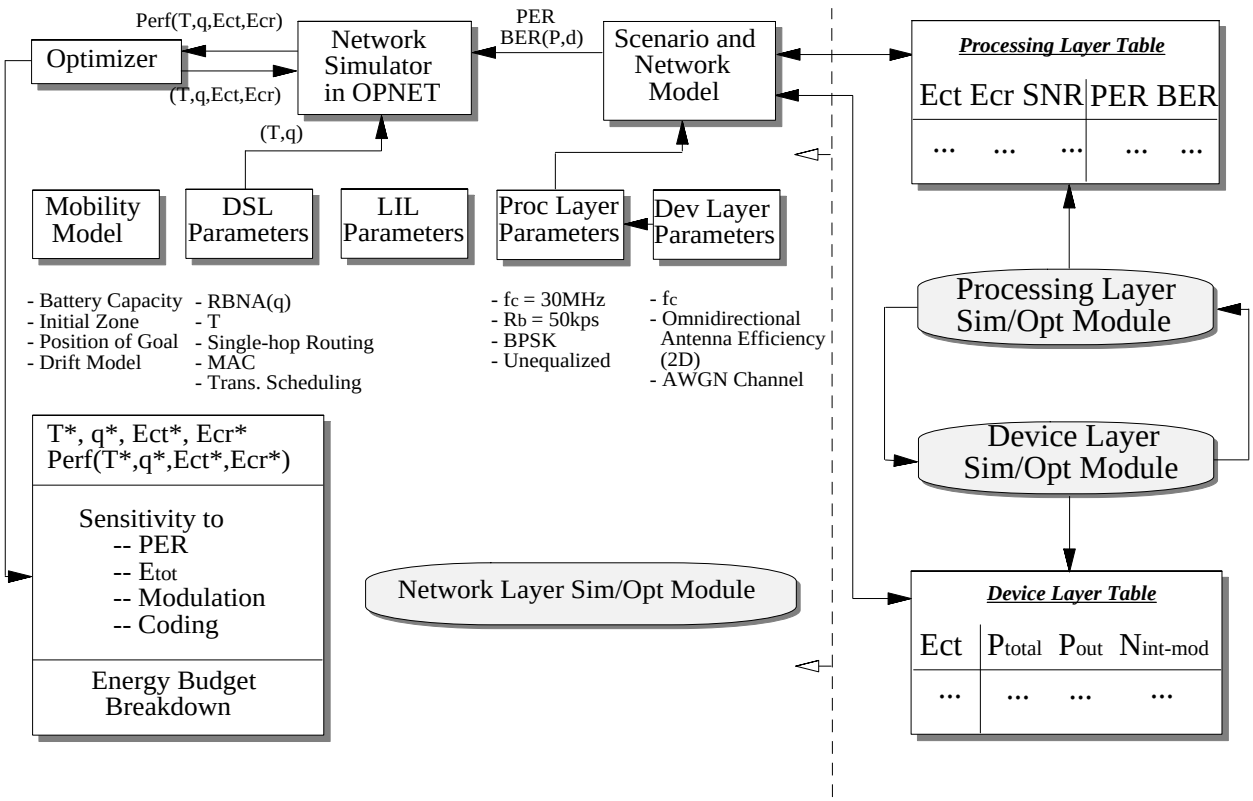

Figure 1: Overview of Simulation-Based Optimization

2.1 Network Scenario

At the network layer, we started with a simple scenario which can be defined as a position awareness task as shown in Figure 2. Multiple mobile nodes are initially deployed in a drop zone of area $1km \times 1km$ and they move toward the same destination $\underline{d} = [d_x, d_y]^T = [6000m, 6000m]^T$ every $T_m = 1s$ at speed $v = 1m/s$ according to a drift model.

While moving toward the destination, each node broadcasts its position awareness packets every T_c, and rebroadcasts the same packet according to retransmission probability q. Each packet, undergoing a propagation model, may or may not reach all other mobile nodes. Each node estimates the positions of all other nodes every $T_e = 2s$ utilizing the packets received from other nodes.

The average root mean squared error between a node's estimate (made by other nodes) and its actual position is obtained as the performance measure of the network layer. In the following subsections, the network model is further divided into drift model, broadcast model, propagation model, estimation model, and energy constraint for detailed explanation.

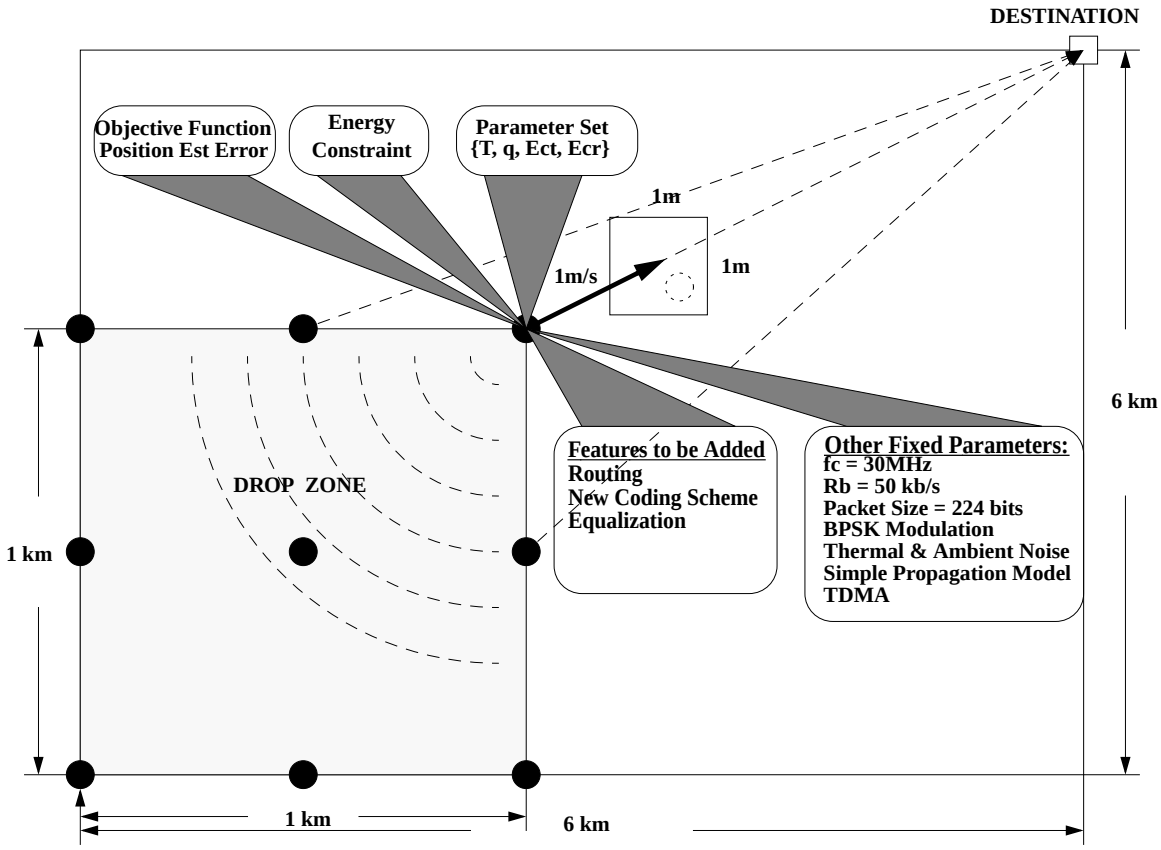

Figure 2: MURI Network Simulation Scenario

2.1.1 Drift Model

All nodes are initially deployed in a drop zone and each node moves toward the goal according to a drift model.

Let $\underline{p}_k^{(i)} = \left[p_{x,k}^{(i)}, p_{y,k}^{(i)}\right]^T$ be the position of node i at time kT_m. Then the mobility model is given by:

$$\underline{p}_k^{(i)} = vT_m \frac{\underline{d} - \underline{p}_{k-1}^{(i)}}{\left\|\underline{d} - \underline{p}_{k-1}^{(i)}\right\|} + \underline{\lambda}_k^{(i)} + \underline{p}_{k-1}^{(i)} \tag{1}$$

where

$$\left\|\underline{d} - \underline{p}_{k-1}^{(i)}\right\| = \sqrt{(d_x - p_{x,k}^i)^2 + (d_y - p_{y,k}^i)^2} \tag{2}$$

and $\underline{\lambda}_k^{(i)} = \left[\lambda_{x,k}^{(i)}, \lambda_{y,k}^{(i)}\right]^T$ represents the uncertainty of the mobility. $\lambda_{x,k}^{(i)}$ and $\lambda_{y,k}^{(i)}$ are uniform random variables on the interval $(-0.5, 0.5)$.

In our current network model, each node jumps from one position to another position every T_m instead of moving continuously from its current position to its next position, even though *OPNET* has the capability of modeling continuous movement.

2.1.2 Broadcast Model

The current communication network model uses time division multiple access (TDMA) and Binary Phase Shift Keying (BPSK) modulation, where each node is assigned a fixed time slot offset to broadcast its position awareness packets every T_c. The transmission data rate for each packet is $R_b = 50kb/s$ and each position awareness packet has length $L = 224bits$.

When it comes to a node's transmission slot, the node makes its first transmission using E_{ct} amount of energy per packet if there is still enough energy left in the battery for it to make the transmission.

After a transmission, the node retransmits the same packet with a predetermined retransmission probability q if there is still enough energy left in the battery and enough time left in its transmission time slot for one packet transmission. The node generates a random number r uniformly distributed on the interval $(0, 1)$. If $r < q$, it retransmits the same packet, increases its consumed energy by E_{ct}, and does the above process again. Otherwise, it stops the retransmission for the current transmission time slot.

2.1.3 Propagation Model

We assume that we have a propagation model that takes into account a direct path from transmitter to receiver, and a reflection path with 180 degree phase change at the reflection point from transmitter to receiver as shown in Figure 3. The cumulative effect of both paths gives us an relation between received power and transmitted power, which is given in Eqn (5). The meaning of all the parameters are given in Table 1.

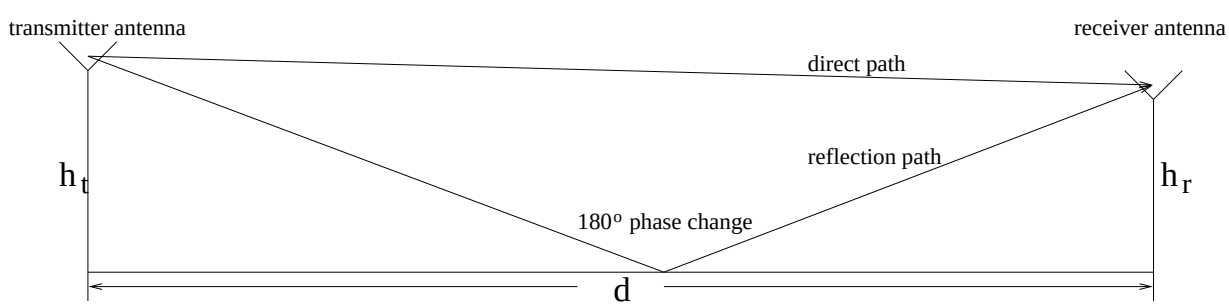

Figure 3: Multiple Path From the transmitter to the Receiver

We need to get a rough idea of how packet error rate is related to the transmitted power before we can set up reasonable energy constraint on each mobile node. Based on our assumptions of the propagation model, we

can calculate bit error rate (BER) and packet error rate (PER) as a function of the transmitted power (the power radiated from antenna) and the propagation distance between the transmitter and the receiver for the uncoded cases.

Since we are only considering thermal noise at the moment without counting any other noise source and thermal noise level is very low, we don't actually need too much transmitted power to get a very low packet error rate.

Table 1: **Symbols Used in Packet Error Rate Calculation**

Symbol	Meaning	Value	Unit
k	Boltzmann constant	$1.379 \cdot 10^{-23}$	$W/Hz/K$
C	propagation speed	$3.0 \cdot 10^{8}$	m/s
T	background temperature	290	K
N_0	one-sided power spectral density	Eqn (3)	W/Hz
f_c	carrier frequency	30	MHz
λ_c	carrier wavelength	Eqn (4)	m
bw	bandwidth	10	kHz
h_t	height of transmitter antenna	1	m
h_r	height of receiver antenna	1	m
G_t	gain of transmitter antenna	1	$none$
G_r	gain of receiver antenna	1	$none$
R_b	data rate	20	kb/s
T_b	data symbol duration	Eqn (6)	s
L	packet length	224	bit
P_t	transmitted power	$[10^{-7}, 1]$	W
d	propagation distance	$(0, 9000)$	m
P_r	received power	Eqn (5)	W
BER	bit error rate	Eqn (8)	$none$
PER	packet error rate	Eqn (9)	$none$

$$N_0 = kT = 1.379 \cdot 10^{-23} \cdot 290 = 3.9991 \cdot 10^{-21} \quad W/Hz \tag{3}$$

$$\lambda_c = \frac{C}{f_c} = \frac{3.0 \cdot 10^8}{30 \cdot 10^6} = 10 \quad m \tag{4}$$

$$P_r = P_t 4 \left(\frac{\lambda_c}{4\pi d}\right)^2 G_t G_r sin^2 \left(\frac{2\pi h_t h_r}{\lambda_c d}\right) = P_t 4 \left(\frac{10}{4\pi d}\right)^2 sin^2 \left(\frac{2\pi}{10d}\right) \tag{5}$$

$$T_b = \frac{1}{R_b} = \frac{1}{20 \cdot 10^3} = 0.00005 \quad s \tag{6}$$

$$E_b = P_r T_b = 0.00005 Pr \quad J \tag{7}$$

$$BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) = Q\left(\sqrt{\frac{2 \cdot 0.00005 Pr}{3.9991 \cdot 10^{-21}}}\right) = Q\left(\sqrt{2.5006 \cdot 10^{16} Pr}\right) \tag{8}$$

Since the communication system is uncoded, we have

$$PER = 1 - (1 - BER)^L \tag{9}$$

Based on the above formulae, we could plot the following set of figures indicating the relationship of PER (or BER) as a function of transmitted power and propagation distance. We could use these theoretical results as a guide to set reasonable energy constraint of our interest region on each node and to design our future power allocation strategy.

Since BPSK modulation is used in our current network layer, Figure 19 shows BER versus E_b/N_0 of a BPSK system. From Figure 14 to Figure 18, which shows the relationship between E_b/N_0 at the receiver and propagation distance and Figure 19, we can derive the relationship of BER and PER versus propagation distance as shown from Figure 4 to Figure 12.

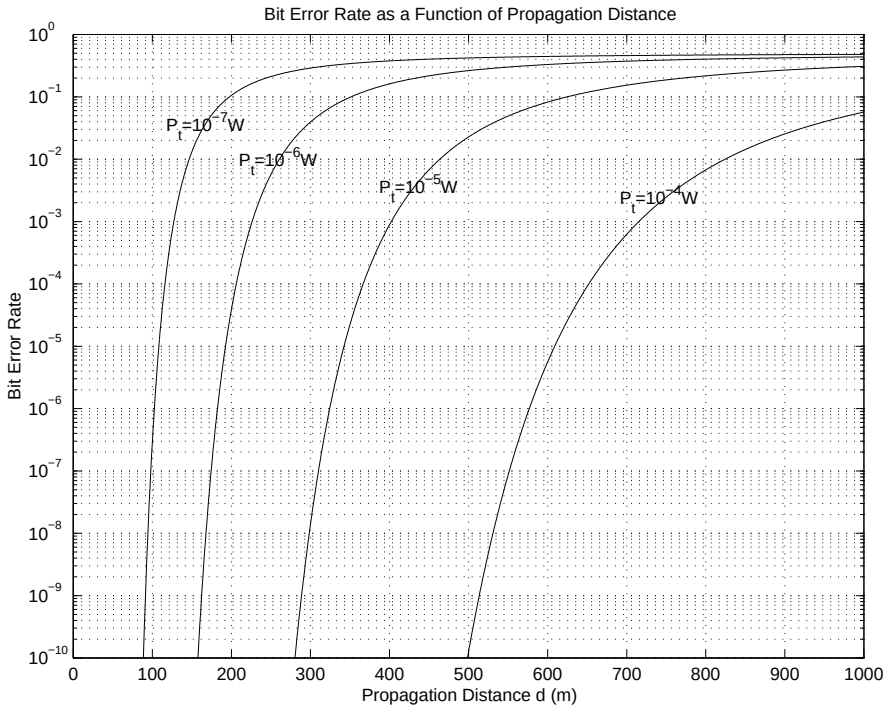

Figure 4: Bit Error Rate as a Function of Distance

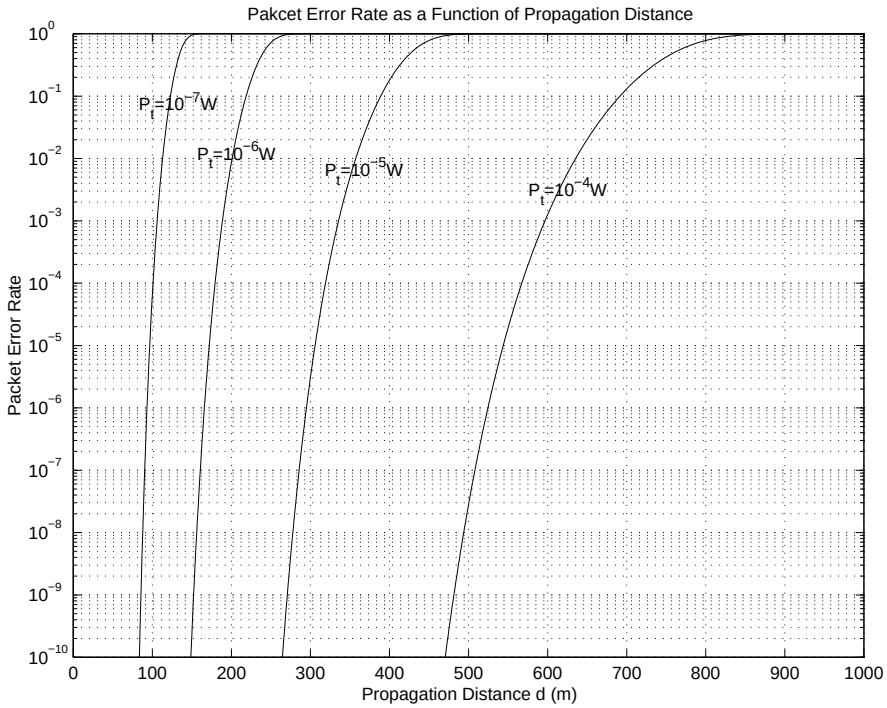

Figure 5: Packet Error Rate as a Function of Distance

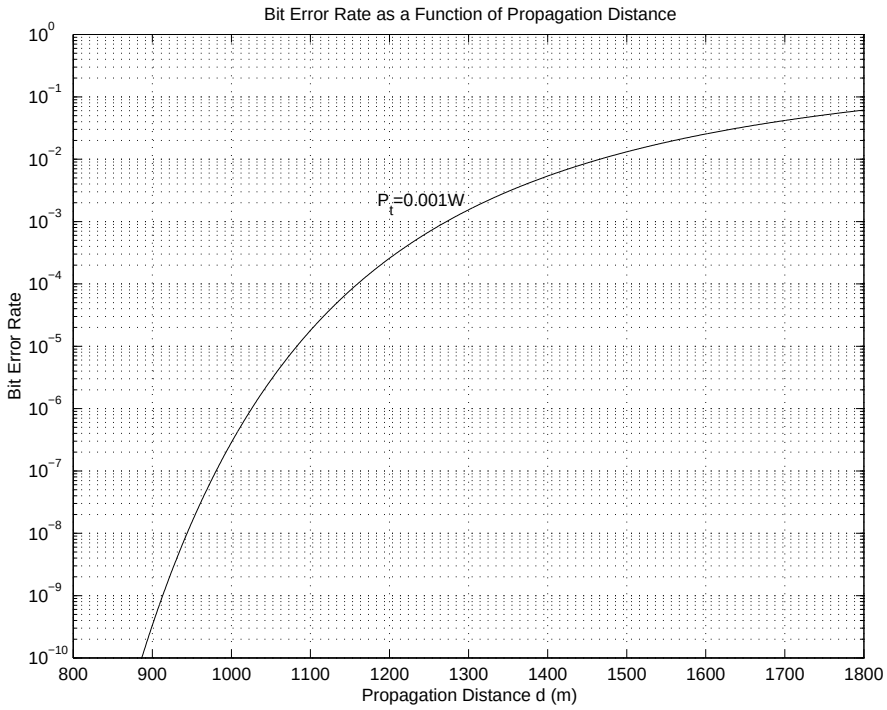

Figure 6: Bit Error Rate as a Function of Distance

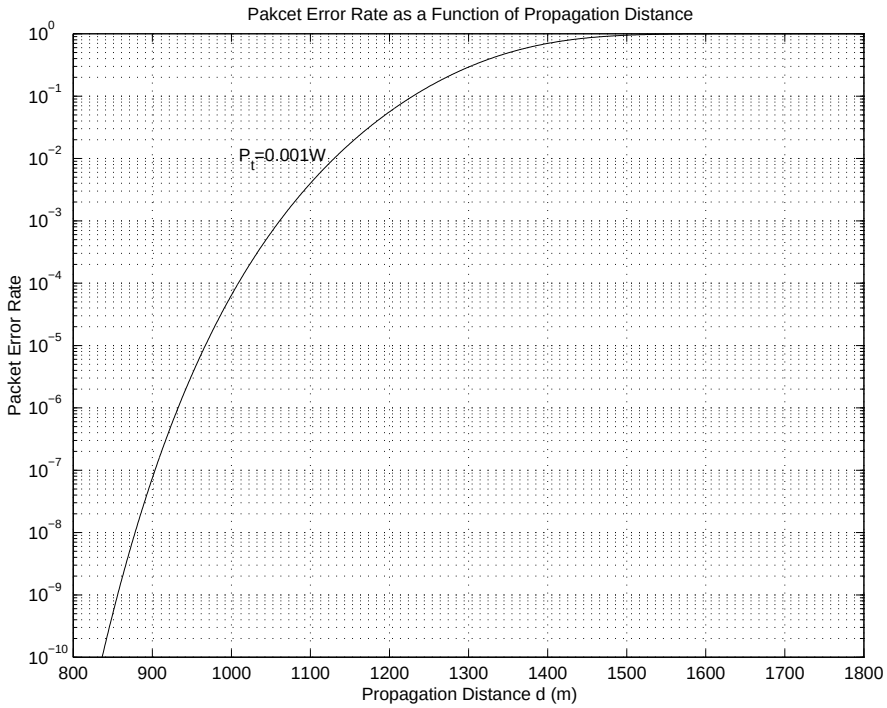

Figure 7: Packet Error Rate as a Function of Distance

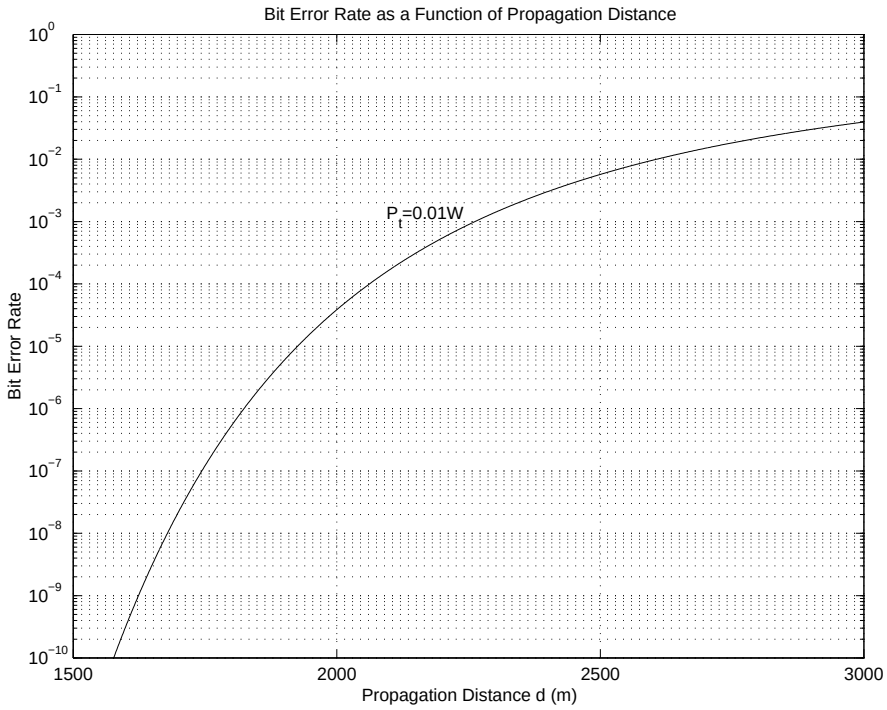

Figure 8: Bit Error Rate as a Function of Distance

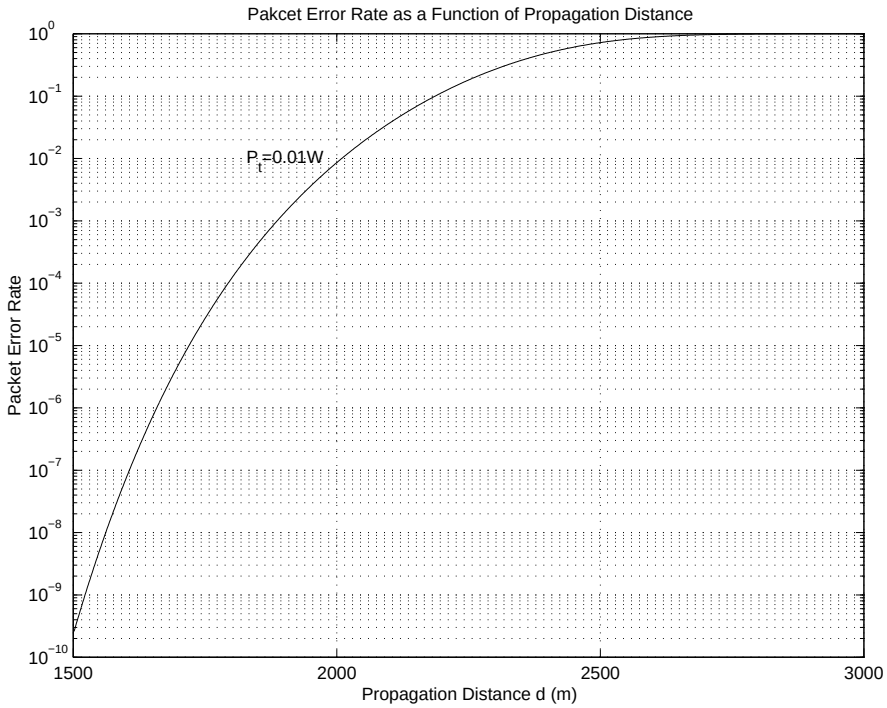

Figure 9: Packet Error Rate as a Function of Distance

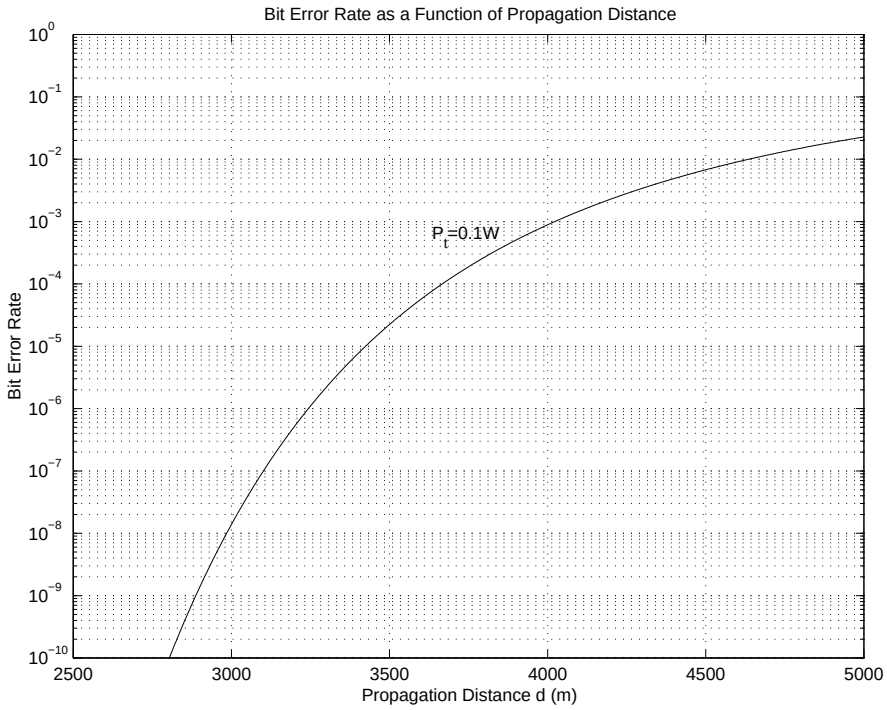

Figure 10: Bit Error Rate as a Function of Distance

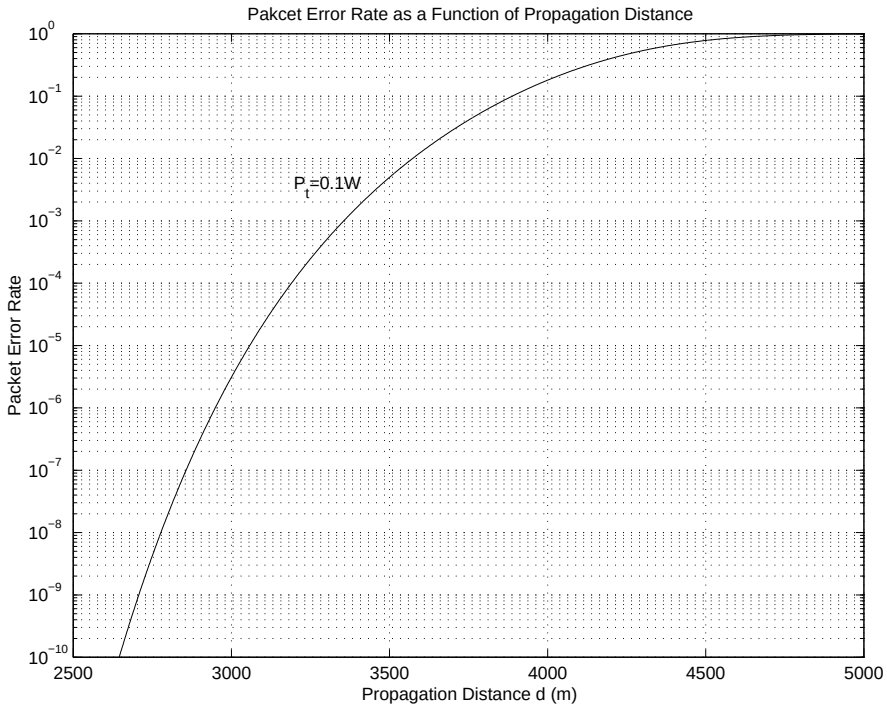

Figure 11: Packet Error Rate as a Function of Distance

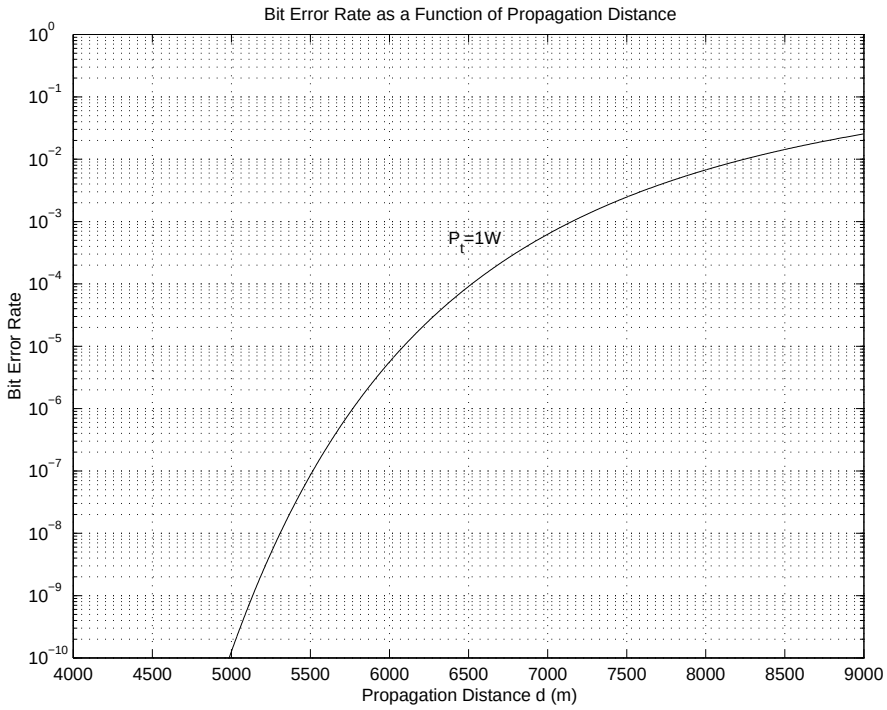

Figure 12: Bit Error Rate as a Function of Distance

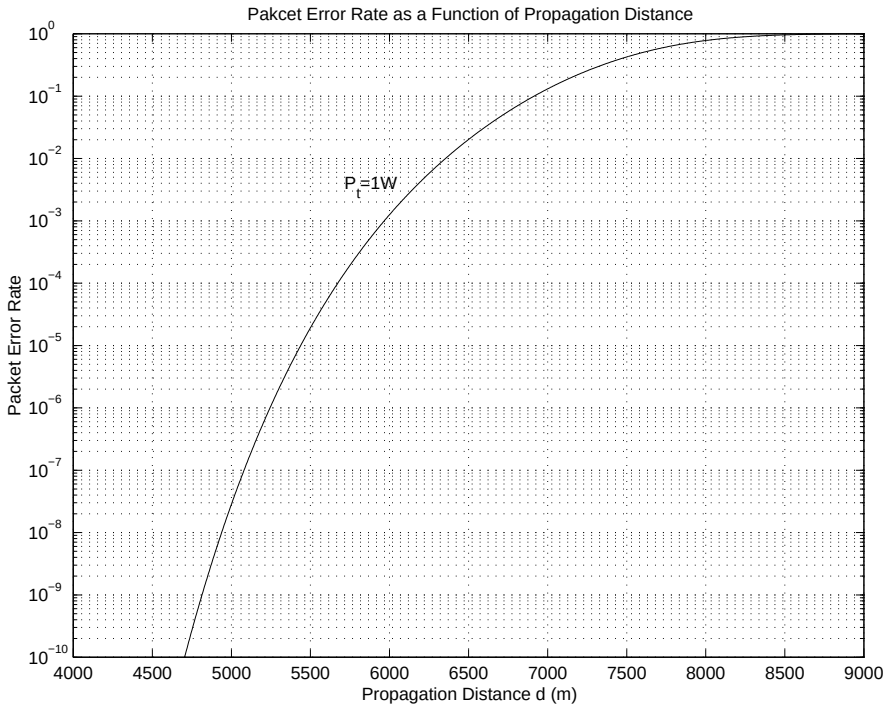

Figure 13: Packet Error Rate as a Function of Distance

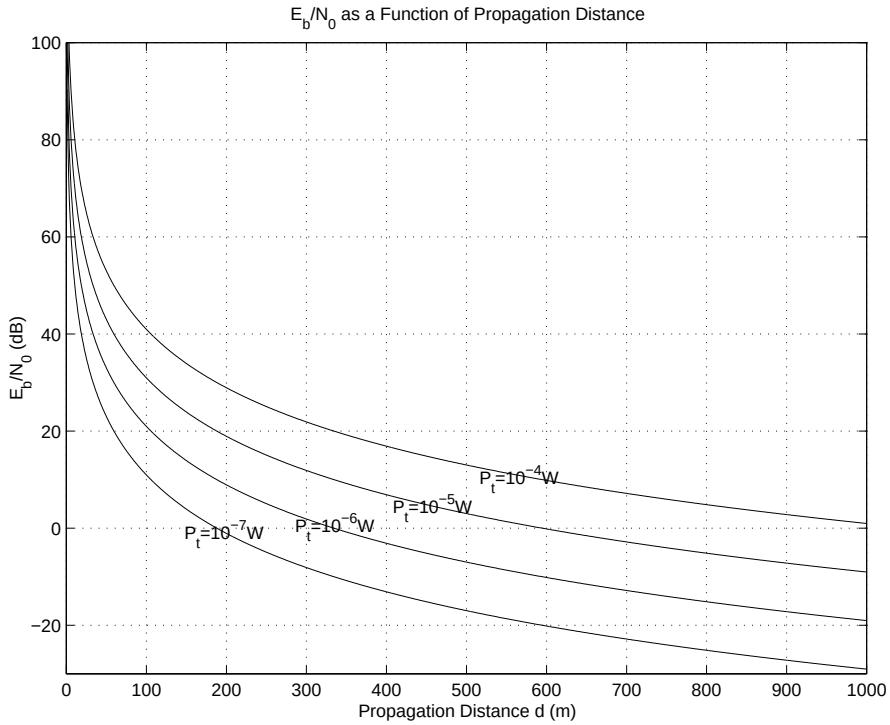

Figure 14: E_b/N_0 as a Function of Distance

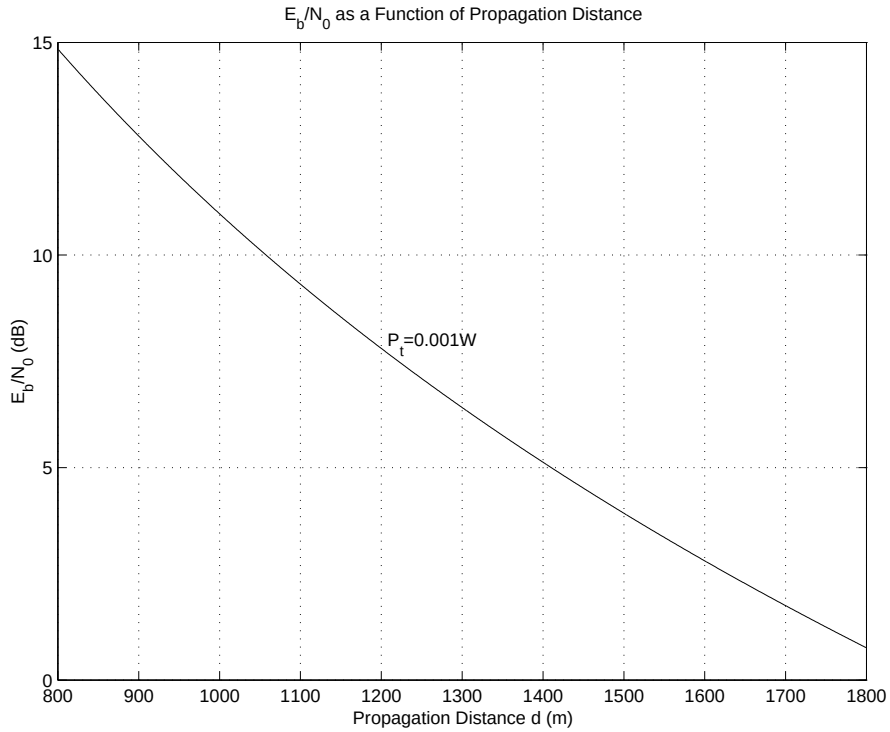

Figure 15: E_b/N_0 as a Function of Distance

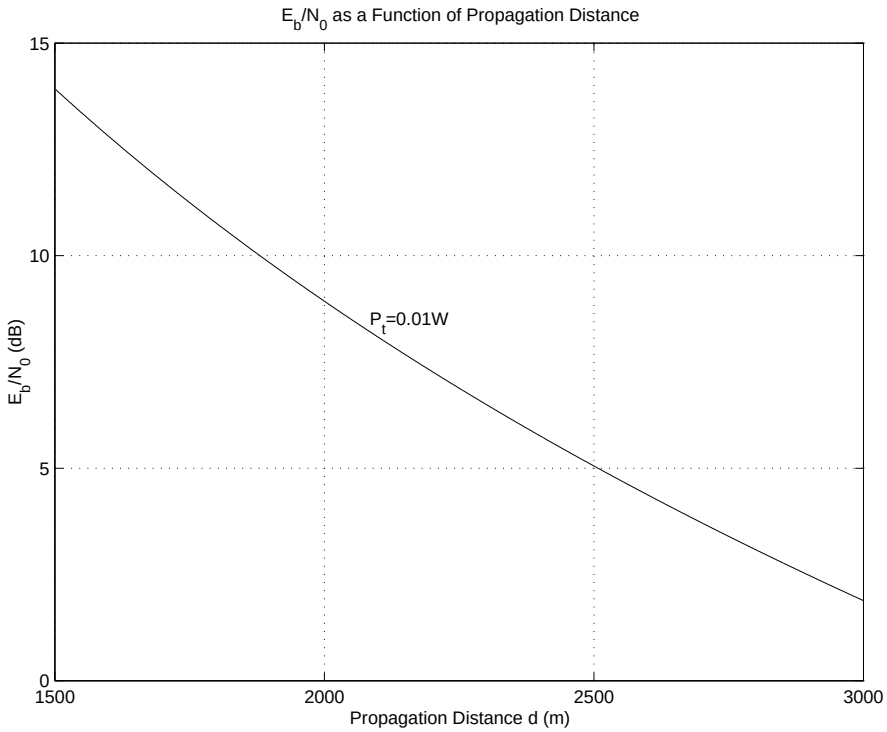

Figure 16: E_b/N_0 as a Function of Distance

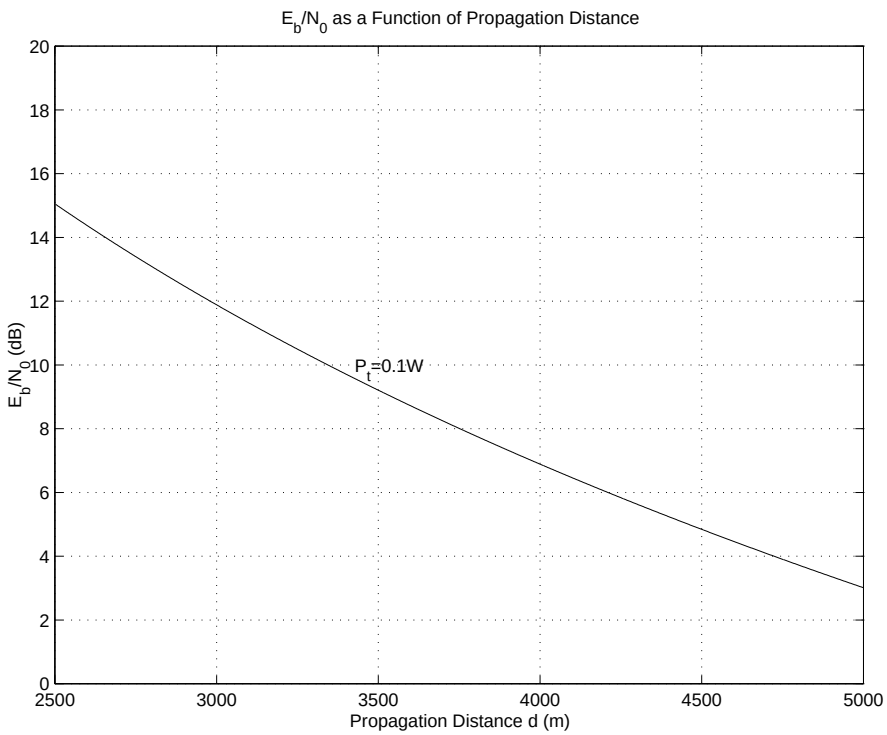

Figure 17: E_b/N_0 as a Function of Distance

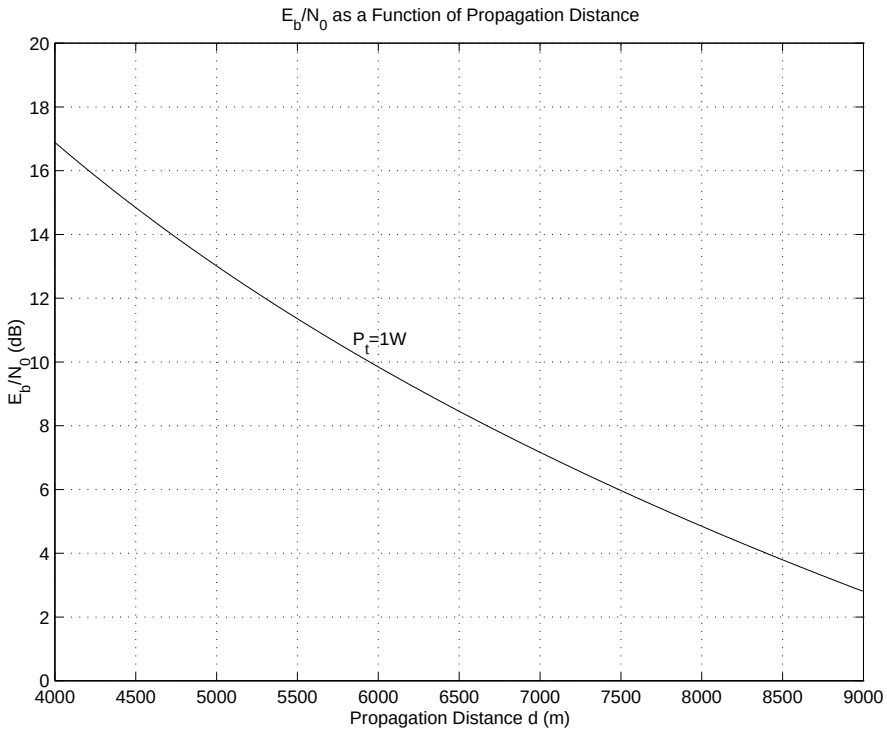

Figure 18: E_b/N_0 as a Function of Distance

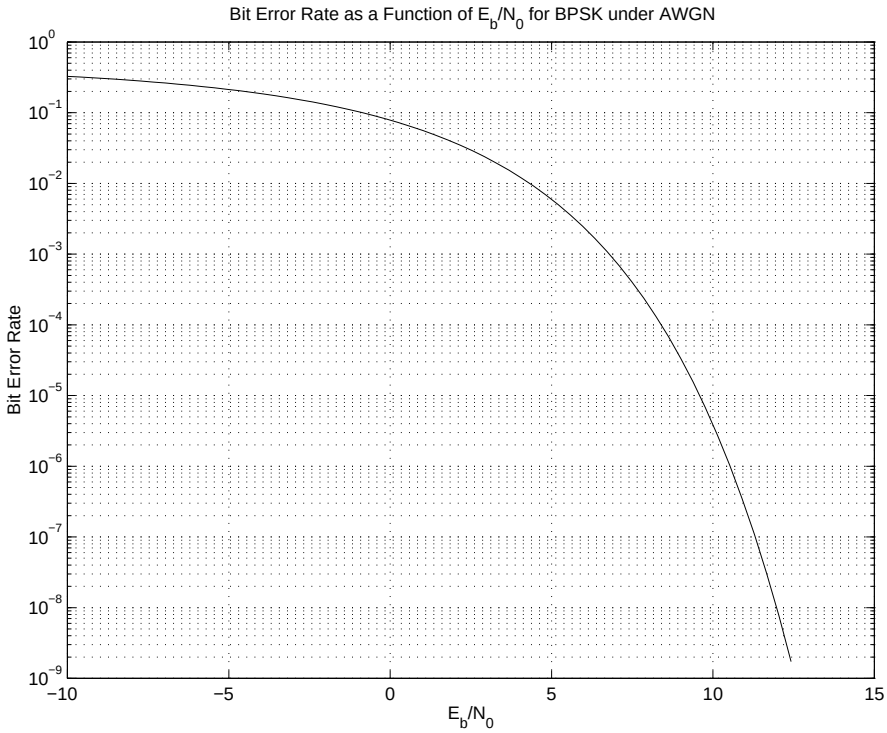

Figure 19: Binary Phase Shift Keying (bpsk) BER

2.1.4 Estimation Model

Each node uses E_{cr} amount of energy to process a received packet no matter whether it can successfully receives the packet or not.

Using the antenna model (will be discussed later in 5.1) and the propagation model, the simulation kernel calculates the signal-to-noise ratio E_b/N_0 at the receiver. Together with the tables/functions provided by the processing and device layers, the packet error rate PER associated with a transmission can be determined, which is then used as a threshold to determine whether the packet can be successfully received or not.

Each node updates its estimate on each other node's position every $T_e = 2s$. Suppose it is time for node 1 to update its estimate on node 2's position, node 1 first gets its last estimate on node 2's position from node 1's own database, suppose this estimate is $\underline{\hat{p}}_{k-1}$. This estimate may be recorded either at time t_{k-1}, where t_{k-1} is the time node 1 last receives a packet from node 2 between $(k-1)T_e$ and kT_e, or at time $(k-1)T_e$, where $(k-1)T_e$ is the time node 1 last gives an estimate on the position of node 2.

Let the estimate made on the position of node 2 as observed by node 1 at time kT_e be $\underline{\hat{p}}_k^{(2\to1)}$. Then the estimation model is given by:

$$\underline{\hat{p}}_k^{(2\to1)} = \underline{\hat{p}}_{k-1}^{(2\to1)} + vT_e \frac{\underline{d} - \underline{\hat{p}}_{k-1}^{(2)}}{\left\| \underline{d} - \underline{\hat{p}}_{k-1}^{(2)} \right\|} \tag{10}$$

where

$$\left\| \underline{d} - \underline{\hat{p}}_{k-1}^{(2\to1)} \right\| = \sqrt{(d_x - \hat{p}_{x,k}^{(2\to1)})^2 + (d_y - \hat{p}_{y,k}^{(2\to1)})^2} \tag{11}$$

If node 1 ever receives a packet from node 2 between time $(k-1)T_e$ and kT_e, then $\underline{\hat{p}}_{k-1}^{(2\to1)} = \underline{p}_{k-1}^{(2)}$. Otherwise $\underline{\hat{p}}_{k-1}^{(2\to1)}$ is the previous estimate of node 1 made on the position of node 2.

The position estimation error at time kT_e made by node 1 on node 2 is defined as

$$e_k^{(2\to)} = \left\| \underline{\hat{p}}_k^{(2\to1)} - \underline{p}_k^{(2)} \right\| \tag{12}$$

Node 1 also gives estimates to the positions of all other nodes $3, \cdots, M$ in the same fashion at time kT_e, where M is the number of nodes in the network model. At the end of simulation, node 1 calculates the average of the position estimation error it made on all other nodes by

$$e^{(1)} = \frac{1}{(M-1)N} \sum_{k=2}^{N} \sum_{i=1}^{M} e_k^{(i\to1)} \tag{13}$$

where N is the total number of estimates that node 1 made on each of other nodes.

Node $2, \cdots, M$ in the network make the estimates on the positions of all other $(M-1)$ nodes in the same way as does node 1. At the end of simulation, node $2, \cdots, M$ also calculate their average position estimation error $e^{(2)}, \cdots, e^{(M)}$ respectively.

The overall network performance measure is given by the average of the position estimation error made by all the nodes in the network:

$$e = \frac{1}{M} \sum_{k=1}^{M} e^{(k)} \tag{14}$$

2.1.5 Energy Constraint

As having been stated in the above sections, each mobile node has an imposed energy constraint, which is realized by means of battery capacity carried by each node. If its consumed energy is more than its carried battery capacity, the node can no longer transmit or receive any packet.

During simulation, caution had to be taken to set this energy constraint to a reasonable value in order to achieve more interesting results.

2.1.6 Optimization Problem Formulation

The goal is to achieve the maximum performance (lowest average position estimation error) for a given energy constraint (battery capacity).

Equivalently, we want to achieve a given performance using the minimum amount of energy.

We first fixed several parameters to reduce the dimension of the problem as shown in Figure 2, i.e., in current network model, the carrier frequency is set to $30MHz$, data rate is $50kb/s$, each packet has size $224bits$, the communication system uses BPSK modulation with convolutional codes and no equalization, and a simple propagation model is assumed as given in Eqn (5), etc.

Since it is very hard to find an analytical expression for average position estimation error e as a function of our chosen parameter set (T, q, Ect, Ecr) with the restriction of the battery capacity E, we have to rely on simulation to get e as the value of the objective function for each parameter set (T, q, Ect, Ecr) generated by the optimizer.

With the concern that each network simulation with different random seeds may give us different position esti-

mation error e, we need to run simulation over the same parameter set (T, q, Ect, Ecr) many times to get an average value of e.

Let $e(i)$ be the position estimation error of the network when $i-th$ random seed is used for simulation, then

$$\overline{e} = \frac{1}{30}\sum_{i=1}^{30} e(i) \tag{15}$$

is used as the value of the objective function for optimization.

The sample variance of $e(1), \cdots, e(30)$ can also be calculated by

$$Var(e) = \frac{1}{29}\sum_{i=1}^{30}(e(i) - \overline{e})^2 \tag{16}$$

As a summary, the problem of network layer simulation and optimization can be defined as follows:

- Energy constraint: E

  Each node has a energy constraint E. It can no longer transmit or receive any packet if its consumed energy is over energy constraint E.

- Interval between successive transmissions: T $(= T_c)$

  Each node broadcasts its position awareness packet every T.

- Retransmission strategy: q

  Each node rebroadcasts the same packet after the first transmission based on q.

- Energy constraint for a packet at the transmitter: Ect

  Each node broadcasts or rebroadcasts the packet with energy constraint Ect for each packet.

- Energy constraint for a packet at the receiver: Ecr

  Each node receives a packet with processing energy constraint Ecr for each received packet.

- Optimization problem:

  Given E, optimize over (T, q, Ect, Ecr) to get the lowest $\overline{e}$.

After defining our problem, we are ready to use *OPNET* to model our network protocols.

2.2 Modeling using *OPNET* software

OPNET software is a very complicated, yet very powerful modeling tool. The basic *OPNET* modeling can be partitioned into three levels, namely, network level, node level, and process level. Please note that these three levels have nothing to do with MURI layered structure. Network modelers can build models bottom up or top down, normally we have to go back and forth among different levels during our design.

The network level modeling, node level modeling, and process level modeling for MURI are shown in Figure 20, Figure 21, and Figure 22 respectively.

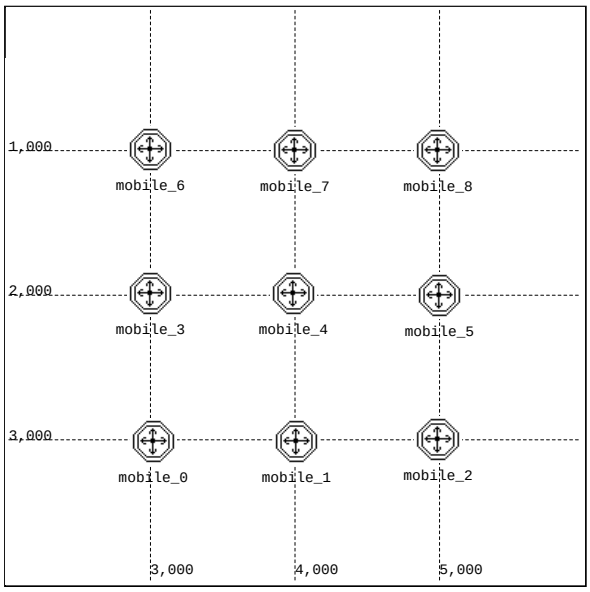

Figure 20: Network Level Modeling

3 Network Layer Optimization

We chose *simulated annealing* program as our optimizer because simulated annealing has great potential in finding global minimum. The integration of *OPNET* simulation and the *simulated annealing* has been carried out to measure the performance of the communication protocols at the network layer.

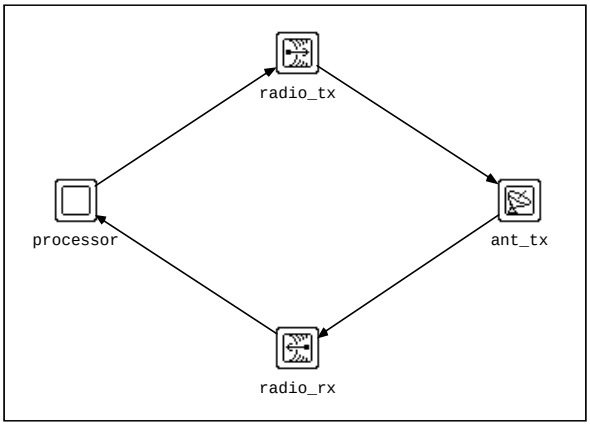

Figure 21: Node Level Modeling

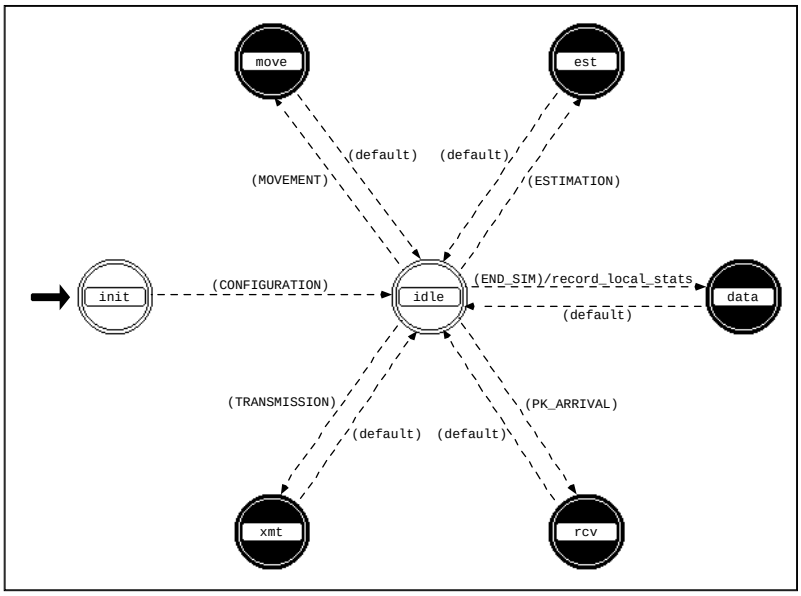

Figure 22: Process Level Modeling

3.1 Simulated Annealing

The method of *simulated annealing* is a technique that has attracted significant attention as suitable for optimization problems of large scale, especially ones where a desired global extremum is hidden among many, poorer, local extrema. This is one of the reasons *simulated annealing* was used as the optimization tool in MURI.

The *simulated annealing* program used in MURI is called *hidenseek* and was developed by the students of Professor Robert Smith's at the IOE department of University of Michigan. *hidenseek* requires a well-defined objective function

on a parameter set and can only deal with parameters that are continuous instead of discrete at the moment.

3.2 Integration of Simulation and Optimization

Since optimization program requires a well defined objective function, we want to provide the optimization program with an illusion that there existed an objective function by using our simulation results. The general setup of simulation and optimization is given in Figure 23.

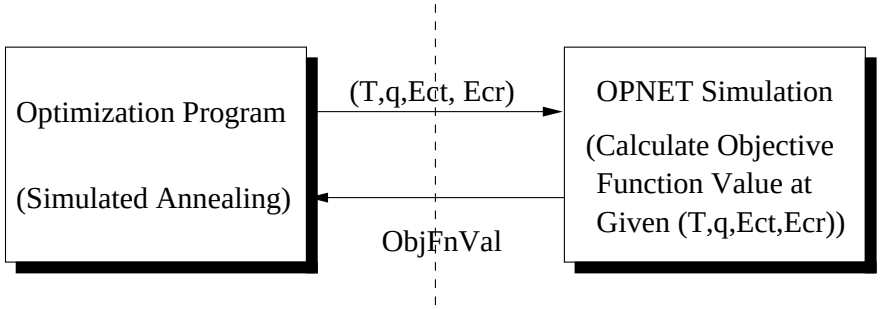

Figure 23: MURI Network Simulation and Optimization Setup

hidenseek relies on hundreds of iterations, or probably more, to converge to the global minimum. For each iteration, it feeds *OPNET* with a parameter set (T, q, Ect, Ecr) and waits for *OPNET* to return the position estimation error. Since we want to average the position estimation error over 30 *OPNET* simulations on the same parameter set (T, q, Ect, Ecr) but with different random seeds and each *OPNET* simulation is quite time-consuming, it is natural and necessary to parallelize *OPNET* simulation on different machines, with each machine running an *OPNET* simulation on the same parameter set, but with a different random seed, as shown in Figure 24. The position estimation error returned by each machine is then averaged and the averaged value is returned to *hidenseek*, upon which a new parameter set is determined by *hidenseek* for another round of simulation by *OPNET*. This procedure keeps going until the number of iterations in *hidenseek* is reached. The best parameter set is given by *hidenseek* at the end of iteration. By parallelizing *OPNET* simulation on different machines, the time span to get the final optimization result is greatly reduced.

As you may have noticed in Figure 24, *OPNET* simulations over the same parameter set are actually running on 40 different machines, however, *problem.c* only takes the first 30 returned position estimation error and returns the average of these 30 values to the optimizer without waiting the rest 10 simulations to finish. By this way, the overall

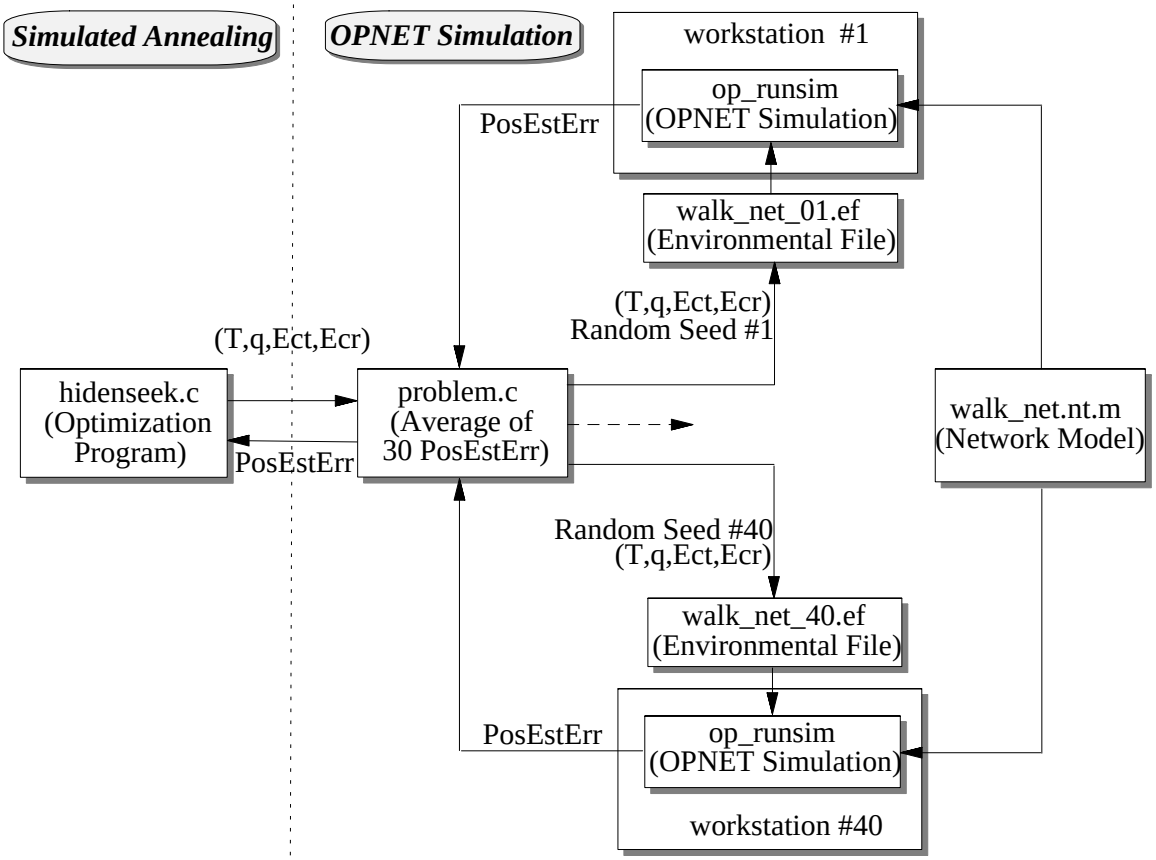

Figure 24: MURI Network Simulation and Optimization Using Parallel Structure

speed of simulation and optimization would be not reduced even if some of simulation processes were killed by other users.

4 Interaction Between Network Layer and Processing Layer

The interaction between network layer and processing layer are mainly through tables and functions. Since each layer requires tremendous amount of time to run its own simulation, it is really not feasible to run simulation at both layers simultaneously.

The idea is that we let each layer catches its own optimizing parameters. The processing layer mainly deals with the modulation, coding, and fast fading, etc., while the network layer deals with mobility, retransmission, and slow fading, etc. Then the processing layer can run its own simulation and obtain the best results with respect to its own optimization parameters. These results are kept in a table and is used by the interface functions to provide network

layer with desired values.

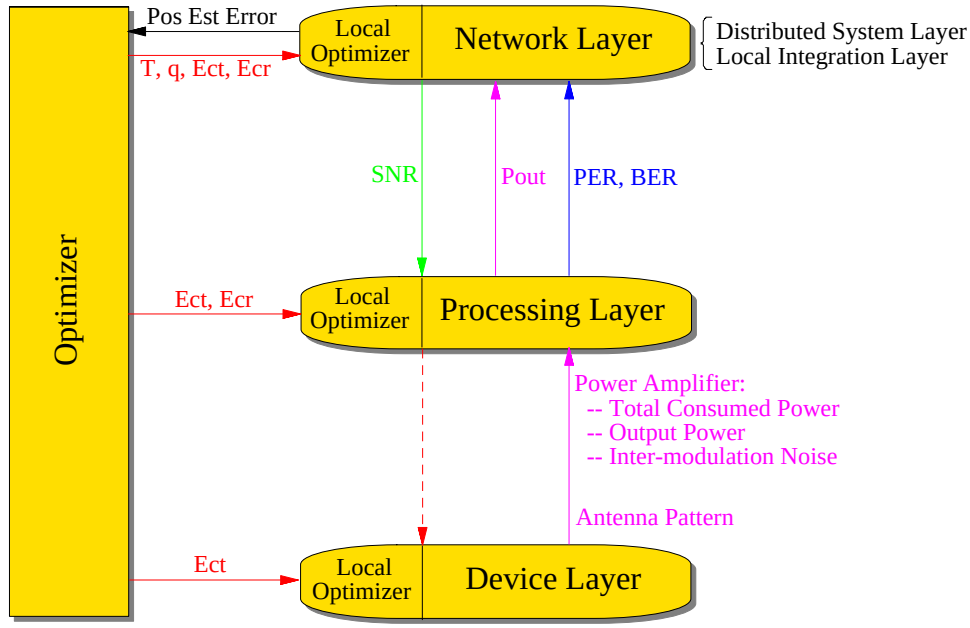

Figure 25: Network Layer and Processing Layer Interaction

The logical sequence of the interactions between network layer and processing layer is given below:

- Step 1: Optimization program determines parameters E_{ct}, E_{cr}, T, q.

- Step 2: Device layer determines P_{out} (amplifier output power).

$$P_{out} = f_1(E_{ct}) \tag{17}$$

where E_{ct} : transmitter energy constraint per packet

- Step 3: Network layer determines SNR (Signal to Noise Ratio).

$$SNR = \frac{P_r \cdot T_s}{N_0} \tag{18}$$

where P_r : received power out of receiver antenna

T_s : channel symbol duration N_0 : thermal noise power density

$$P_r = P_{out} 4 \left(\frac{\lambda_c}{4\pi d}\right)^2 G_t G_r \eta_t \eta_r sin^2 \left(\frac{2\pi h_t h_r}{\lambda_c d}\right) \tag{19}$$

- Step 4: Processing Layer determines PER (Packet Error Rate), which is computed by calling the function f_4^* (this function uses the table generated by the processing layer).

$$PER = f_4^*(E_{ct}, E_{cr}, SNR) \tag{20}$$

where E_{cr} : receiver energy constraint per packet

5 Interaction Between Network Layer and Device Layer

Right now, the only data that network layer directly gets from device layer is the antenna model. The antenna model provides an antenna gain, which is incorporated into the network layer pipeline stage for each packet transmission.

5.1 Antenna Model

Each node carries a dipole antenna (whip antenna) of $1m$ high with a certain efficiency at the carrier frequency. Since all the nodes are on the same altitude and each node transmits packets using a carrier frequency of $30MHz$, the corresponding antenna efficiency is 14.814%, and the directivity gain is $1.775dB$. The overall antenna gain is therefore $1.775 + 10Log_{10}0.14814 = -6.518dB$, which means that the actual radiated power out of antenna is $6.518dB$ less than the input power into antenna.

6 Simulation Results

One example given in this section may show how computationally intensive the integration of simulation and optimization and coupling of other layers can be. If *hidenseek* iterates 200 times and we are using parallel simulation and optimization structure as shown in Figure 24, one realization of the integration of simulation and optimization took 24 hours for a network model with 9 nodes, each with a battery capacity $1.0J$.

Table 2: **Statistics for Typical Network Simulation**

Number of Nodes	9
Coordinates of Destination	$(6000, 6000)m$
Simulation Time	Around $7000s$
Initial Drop Zone	$1km \times 1km$
Range on Decision Variables	$T \in [4, 30]s$, $q \in [0, 0.5]$ $Ect \in [0.00042631, 0.000894988]J$, $Ecr \in [0.000195, 0.000260]J$
Battery Capacity	$1.0J$
Number of Iterations	200
RMS Position Est Error	$1.195902m$
Optimal Decision Variables	$T^* = 21.040825$, $q^* = 0.100498$ $Ect^* = 0.000845J$, $Ecr^* = 0.000213J$
Computing Time	About 24 hours, 40 workstations in parallel

The network performance versus energy tradeoff is given in Figure 26.

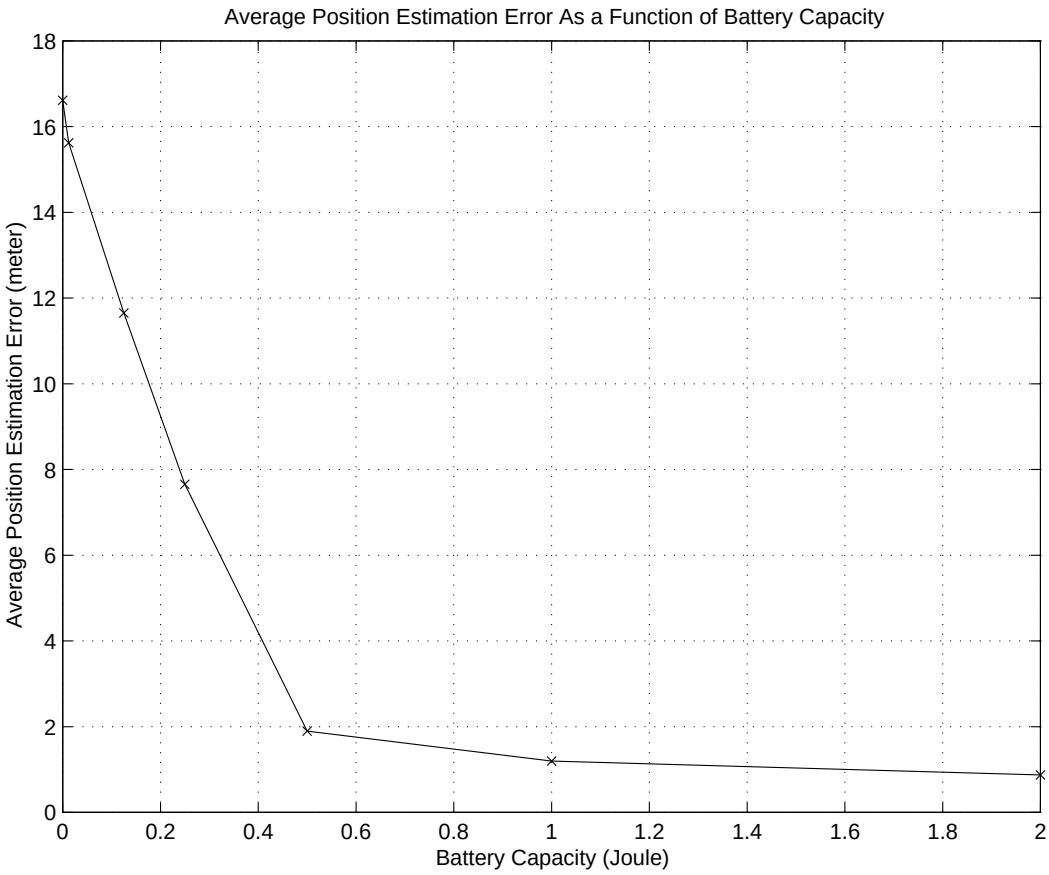

Figure 26: Average Position Estimation Error versus Energy Constraint

7 Analysis of Simplified Network Scenario

This section will be written later when a final complete analysis is done.

8 Proposed Power Adjustification

Based on the PER versus propagation distance formula in AWGN, we can set our target of a particular PER that we want to achieve, then calculate the P_t based on the current estimated distance and inverse formula of Eqn (5).

Suppose BER is small, then Eqn (9) becomes

$$PER \simeq 1 - (1 - BER)^L = 1 - (1 - L \cdot BER) = L \cdot BER \tag{21}$$

which implies that

$$BER \simeq \frac{PER}{L} \tag{22}$$

Since BER is small, we can also approximate Eqn (8) by

$$BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \simeq e^{-\frac{E_b}{N_0}} \tag{23}$$

Then, By Eqn (21) and Eqn (23), we have

$$E_b \simeq N_0 ln\left(\frac{L}{PER}\right) \tag{24}$$

Based on Eqn (24) and Eqn (6), we have

$$P_r = \frac{E_b}{T_b} = E_b R_b \simeq N_0 R_b ln\left(\frac{L}{PER}\right) \tag{25}$$

Based on current propagation model of Eqn (5), we have

$$P_t = \frac{P_r}{4\left(\frac{\lambda_c}{4\pi d}\right)^2 G_t G_r sin^2\left(\frac{2\pi h_t h_r}{\lambda_c d}\right)} \tag{26}$$

Assuming that $d >> max(h_t, h_r)$, then

$$sin\left(\frac{2\pi h_t h_r}{\lambda_c d}\right) \simeq \frac{2\pi h_t h_r}{\lambda_c d} \tag{27}$$

Substitute Eqn (25) and Eqn (27) into Eqn (26), we have

$$P_t \simeq \frac{N_0 R_b ln\left(\frac{L}{PER}\right)}{4\left(\frac{\lambda_c}{4\pi d}\right)^2 G_t G_r \left(\frac{2\pi h_t h_r}{\lambda_c d}\right)^2} = \frac{N_0 R_b d^4 ln\left(\frac{L}{PER}\right)}{G_t G_r h_t^2 h_r^2} \tag{28}$$

Suppose we are targeting at $PER = 10^{-3}$ and assume that we use the value of parameters given in Table 1, then power allocation strategy for each packet transmission is

$$P_t \simeq 9.8533 \cdot 10^{-16} \cdot d^4 \quad W \tag{29}$$

We want the power allocation strategy to be simple because we want to save the processing power at each mobile nodes. It is worth noting that any multiplicative factor of power loss or gain can be thrown into the coefficient before d^4 in Eqn (29).

At the moment, the power allocation is not used in network scenario, because we don't want to add another degree of freedom for optimization to complicate our problem.

9 Conclusions and Future Work

The *basic research* contribution of this part of the MURI is:

> *the synthesis of an optimization procedure that is based on a decomposition of the overall system using the four-layer hierarchy described in the proposal*

We have "broken" a complex optimization problem with exorbitant computational requirements into one where:

- the optimization is "divided" between the *device* layer, the *processing* layer, and the *network* layer (for distributed system and local integration layers), by exploiting the *structure* in the system, and *without* loss of optimality
- this decomposition reveals the information (*key parameters*) that needs to be exchanged among layers in order to achieve the desired optimal solution
- the optimization at the device and processing layers can be done separately from the optimization at the network layer, using the previously-described *tables* as the coupling mechanism
- the simulation-based integrated optimization at the network layer *is* computationally tractable

Refined models and algorithms can be implemented within the current framework. Of particular interest are:

1. *Terrain constraints*
2. *Interference*
3. *Power adaptation*
4. *Multi-hop routing*
5. *Retransmission strategy*

Our focus will be to investigate *how to* modify (if necessary) the current simulation-based optimization procedure in order to incorporate

- the above-mentioned refinements
- the results of the on-going basic research at all layers

Acknowledgment

I would like to thank Prof. Wayne Stark, Prof. Stéphane Lafortune, Prof. Demosthenis Teneketzis, Prof. Kimberly Wasserman, Karl Kluge and all the students George Barrett, Rami DeBouk, Christopher Lott and Tara Javidi for their helpful opinions during our weekly meeting, Olivier Boivineau for the contribution to MATLAB program of network simulation, J.D. Shumpert for the contribution to the antenna model, and Andrew Worthen for the interface functions that connects the network layer and the processing layer.

References

[1] Mil 3, "OPNET Modeler Manual," 1997.

[2] Pablo Aghemo, George Barrett, Olivier Boivineau, Rami Debouk, Utkan Demirci, Christopher Lott, "Very High Level Scenario for the MURI Simulation," 1998

[3] Karl Kluge, "Wireless Network Simulation and Design Optimization Plan for the MURI 'Low Energy Electronics Design for Mobile Platforms', " January 7, 1999.

[4] H. Edwin Romeijn and Robert L. Smith, "Simulated Annealing for Constrained Global Optimization," Journal of Global Optimization 5:101-126, 1994.

Appendix A: Running Simulation

In order to have the simulation running parallel on different CAEN workstations, we should have the following software:

`masc` for disturbing simulation over CAEN workstations

`hidenseek` for optimization

`op_runsim` for simulation.

There are mainly two steps to set up and run simulation and optimization.

1. We should first start `masc` program in directory `masc` on EECS machine to dispatch simulation processes. `masc` logs into 40 CAEN workstations and starts a `deamon` process on each of these 40 CAEN workstations. Type

   ```
   nohup masc INI < in.dat &
   ```

   at the shell prompt to run this dispatcher background. In the above command, INI is the initial input file for `masc` and `in.dat` is used to assign each `deamon` an ID number. A typical in.dat file is :

   ```
   40
   0-39
   ```

   Each `deamon` monitors the trigger of the input file to the network simulator and output file written by the network simulator.

2. After all the `deamon` processes have been disseminated, we should then run the optimizer 'hidenseek' in directory `sim` on EECS machine. Type

   ```
   nohup hidenseek &
   ```

   at the command line to run optimizer background.

   `hidenseek` generates the parameter set to be evaluated by the network simulator. The parameter set is used by the `problem` interface to generate input files (called environmental files) to the `op_runsim` simulator. `op_runsim` simulator takes the environmental file and the network model we developed to run the simulation. At the end of each simulation, it writes the result to a file, which will be read by `problem` interface. `problem` interface takes the average of the results and returns it as the value of the objective function to `hidenseek`.