%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %

% SEP-E: An Enhanced Stable Election Protocol for clustered %

% heterogeneous wireless sensor networks %

% %

% (c) Akshay Gore %

% Department of Electronics and Communication Engineering

% Chandigarh University, India

% akshaygore@live.com %

% %

% To report your comment or any bug please send e-mail to: %

% akshaygore@live.com

clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Field Dimensions - x and y maximum (in meters)

xm=100;

ym=100;

%x and y Coordinates of the Sink

sink.x=0.5\*xm;

sink.y=0.5\*ym;

%Number of Nodes in the field

n=100;

%Optimal Election Probability of a node

%to become cluster head

p=0.1;

%Energy Model (all values in Joules)

%Initial Energy

Eo=0.1;

%Eelec=Etx=Erx

ETX=50\*0.000000001;

ERX=50\*0.000000001;

%Transmit Amplifier types

Efs=10\*0.000000000001;

Emp=0.0013\*0.000000000001;

%Data Aggregation Energy

EDA=5\*0.000000001;

%Values for Hetereogeneity

%Percentage of nodes than are advanced

m=0.1;

%Percentage of nodes than are intermediate

x=0.2;

%\alpha

a=1;

%Beta

b=0.5;

%maximum number of rounds

rmax=500;

%variables

Ave\_CH=0;

sum=0;

count\_ch=0;

Throughput=0;

% packets size

Packet=40;

%%%%%%%%%%%%%%%%%%%%%%%%% END OF PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%

%Computation of do

do=sqrt(Efs/Emp);

%Creation of the random Sensor Network

figure(1);

rand('seed',19)

for i=1:1:n

S(i).xd=rand(1,1)\*xm;

XR(i)=S(i).xd;

S(i).yd=rand(1,1)\*ym;

YR(i)=S(i).yd;

S(i).G=0;

S(i).E=0;

%initially there are no cluster heads only nodes

S(i).type='N';

keep(i)=i;

temp\_rnd0=i;

%Random Election of Normal Nodes

if (temp\_rnd0>=(x+m)\*n+1)

S(i).E=Eo;

S(i).ENERGY=0;

%%%%plot(S(i).xd,S(i).yd,'o');

hold on;

end

%Random Election of intermediate Nodes

if (temp\_rnd0<(m+x)\*n+1) && (temp\_rnd0>m\*n)

S(i).E=Eo\*(1+b);

S(i).ENERGY=0.5;

%%%%plot(S(i).xd,S(i).yd,'\*');

hold on;

end

%Random Election of Advanced Nodes

if (temp\_rnd0<m\*n+1)

S(i).E=Eo\*(1+a);

S(i).ENERGY=1;

%%%%plot(S(i).xd,S(i).yd,'D');

hold on;

end

end

S(n+1).xd=sink.x;

S(n+1).yd=sink.y;

%%plot(S(n+1).xd,S(n+1).yd,'x');

%First Iteration

figure(1);

%counter for CHs

countCHs=0;

%counter for CHs per round

rcountCHs=0;

cluster=1;

countCHs;

rcountCHs=rcountCHs+countCHs;

flag\_first\_dead=0;

flag\_first\_Hdead=0;

flag\_last\_dead=0;

c=1;

for r=0:1:rmax

for i=1:1:n

if(S(i).E>0)

holder(i)=S(i).E;

id(i)=keep(i);

node= struct('energy', holder, 'id',id);

[energy,index] = sort([node.energy],'descend'); % Sort all energy values, largest first

end

end

total=0;

for k=1:length(node.energy)

energy\_level=sort(node.energy, 'descend');

total=total + node.energy(k);

end

average=total/length(node.energy);

TEnergy(r+1)=total;

AveEnergy(r+1)=average;

r

%Election Probability for Normal Nodes

pnrm=( p/ (1+a\*m+b\*x) );

%Election Probability for intermediate Nodes

pint=( p\*(1+b)/ (1+a\*m+b\*x) );

%Election Probability for Advanced Nodes

padv= ( p\*(1+a)/(1+a\*m+b\*x) );

%Operation for heterogeneous epoch

if(mod(r, round(1/pnrm) )==0)

for i=1:1:n

S(i).G=0;

S(i).cl=0;

end

end

%Operations for sub-epochs

if(mod(r, round(1/padv) )==0)

for i=1:1:n

if(S(i).ENERGY==1)

S(i).G=0;

S(i).cl=0;

end

end

end

%Operations for sub-epochs

if(mod(r, round(1/pint) )==0)

for i=1:1:n

if(S(i).ENERGY==0.5)

S(i).G=0;

S(i).cl=0;

end

end

end

hold off;

%Number of Half dead nodes

Hdead=0;

%Number of half dead Advanced Nodes

Hdead\_a=0;

%Number of half dead Normal Nodes

Hdead\_n=0;

%Number of half dead intermediate Nodes

Hdead\_in=0;

%Number of dead nodes

dead=0;

%Number of dead Advanced Nodes

dead\_a=0;

%Number of dead Normal Nodes

dead\_n=0;

%Number of dead intermediate Nodes

dead\_in=0;

%counter for bit transmitted to Bases Station and to Cluster Heads

packets\_TO\_BS=0;

packets\_TO\_CH=0;

%counter for bit transmitted to Bases Station and to Cluster Heads

%per round

PACKETS\_TO\_CH(r+1)=0;

PACKETS\_TO\_BS(r+1)=0;

figure(1);

for i=1:1:n

%Checking if the energy is less or reduced by half

if (S(i).E<=(Eo/2)) && (S(i).E>0)

plot(S(i).xd,S(i).yd,'yellow .');

Hdead=Hdead+1;

if(S(i).ENERGY==1)

Hdead\_a=Hdead\_a+1;

end

if(S(i).ENERGY==0.5)

Hdead\_in=Hdead\_in+1;

end

if(S(i).ENERGY==0)

Hdead\_n=Hdead\_n+1;

end

hold on;

end

%checking the rate of energy dissipation in normal, intermediate and

%advance nodes

if (S(i).E<=Eo)||(S(i).E>Eo)

if(S(i).ENERGY==0)

RnEnergy(r+1)=S(i).E;

end

if (S(i).ENERGY==0.5)

RINEnergy(r+1)=S(i).E;

end

if (S(i).ENERGY==1)

RAEnergy(r+1)=S(i).E;

end

end

%checking if there is a dead node

if (S(i).E<=0)

plot(S(i).xd,S(i).yd,'red .');

dead=dead+1;

if(S(i).ENERGY==1)

dead\_a=dead\_a+1;

end

if(S(i).ENERGY==0)

dead\_n=dead\_n+1;

end

if(S(i).ENERGY==0.5)

dead\_in=dead\_in+1;

end

hold on;

end

if S(i).E>0

S(i).type='N';

if (S(i).ENERGY==0)

plot(S(i).xd,S(i).yd,'o');

end

if (S(i).ENERGY==0.5)

plot(S(i).xd,S(i).yd,'p');

end

if (S(i).ENERGY==1)

plot(S(i).xd,S(i).yd,'D');

end

hold on;

end

end

plot(S(n+1).xd,S(n+1).yd,'x');

HSTATISTICS(r+1).DEAD=Hdead;

HDEAD(r+1)=Hdead;

HDEAD\_N(r+1)=Hdead\_n;

HDEAD\_IN(r+1)=Hdead\_in;

HDEAD\_A(r+1)=Hdead\_a;

%When the first node is half dead

if (Hdead==1)

if(flag\_first\_Hdead==0)

first\_Hdead=r;

flag\_first\_Hdead=1;

end

end

STATISTICS(r+1).DEAD=dead;

DEAD(r+1)=dead;

DEAD\_N(r+1)=dead\_n;

DEAD\_IN(r+1)=dead\_in;

DEAD\_A(r+1)=dead\_a;

%When the first node dies

if (dead==1)

if(flag\_first\_dead==0)

first\_dead=r;

flag\_first\_dead=1;

end

end

%Number of alive Nodes

alive=0;

%Number of alive Normal Nodes

alive\_n=0;

%Number of alive intermediate Nodes

alive\_in=0;

%Number of alive advance Nodes

alive\_a=0;

for i=1:1:n

%checking number of alive node per round

if (S(i).E>0)

alive=alive+1;

if(S(i).ENERGY==1)

alive\_a=alive\_a+1;

end

if(S(i).ENERGY==0.5)

alive\_in=alive\_in+1;

end

if(S(i).ENERGY==0)

alive\_n=alive\_n+1;

end

hold on;

end

%checking nodes status

if (S(i).E>0)

nodes\_status=1;

end

if (S(i).E<0)

nodes\_status=0;

end

STATISTICS(i).Status=nodes\_status;

Status(i)=nodes\_status;

ASTATISTICS(r+1).Live=alive;

Live(r+1)=alive;

Live\_n(r+1)=alive\_n;

Live\_in(r+1)=alive\_in;

Live\_a(r+1)=alive\_a;

end

for i=1:1:n

%checking for last dead or alive node

if (alive==1 && S(i).E>0)

if (S(i).ENERGY==1||S(i).ENERGY==0||S(i).ENERGY==0.5)

plot(S(i).xd,S(i).yd,'green .');

last\_dead=r;

Instability=last\_dead-first\_dead;

flag\_last\_dead=1;

end

end

end

countCHs=0;

cluster=1;

for i=1:1:n

if(S(i).E>0)

temp\_rand=rand;

if ( (S(i).G)<=0)

%Election of Cluster Heads for normal nodes

if( ( S(i).ENERGY==0 && ( temp\_rand <= ( pnrm / ( 1 - pnrm \* mod(r,round(1/pnrm)) )) ) ) )

countCHs=countCHs+1;

packets\_TO\_BS=packets\_TO\_BS+1;

PACKETS\_TO\_BS(r+1)=packets\_TO\_BS;

S(i).type='C';

S(i).G=100;

C(cluster).xd=S(i).xd;

C(cluster).yd=S(i).yd;

plot(S(i).xd,S(i).yd,'k\*');

distance=sqrt( (S(i).xd-(S(n+1).xd) )^2 + (S(i).yd-(S(n+1).yd) )^2 );

C(cluster).distance=distance;

C(cluster).id=i;

X(cluster)=S(i).xd;

Y(cluster)=S(i).yd;

cluster=cluster+1;

%Calculation of Energy dissipated

distance;

if (distance>do)

S(i).E=S(i).E- ( (ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance\*distance\*distance ));

Energy\_CH\_N=(ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance\*distance\*distance );

end

if (distance<=do)

S(i).E=S(i).E- ( (ETX+EDA)\*(Packet) + Efs\*Packet\*( distance \* distance ));

Energy\_CH\_N=(ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance);

end

end

% modular(r+1)=( 1 - 0.1 \* mod(r,round(1/0.1)));

% modular\_ver(r+1)=( 1 - 0.1 \* r);

%Election of Cluster Heads for intermediate nodes

if( ( S(i).ENERGY==0.5 && ( temp\_rand <= ( pint / ( 1 - pint \* mod(r,round(1/pint)) )) ) ) )

countCHs=countCHs+1;

packets\_TO\_BS=packets\_TO\_BS+1;

PACKETS\_TO\_BS(r+1)=packets\_TO\_BS;

S(i).type='C';

S(i).G=100;

C(cluster).xd=S(i).xd;

C(cluster).yd=S(i).yd;

plot(S(i).xd,S(i).yd,'k\*');

distance=sqrt( (S(i).xd-(S(n+1).xd) )^2 + (S(i).yd-(S(n+1).yd) )^2 );

C(cluster).distance=distance;

C(cluster).id=i;

X(cluster)=S(i).xd;

Y(cluster)=S(i).yd;

cluster=cluster+1;

%Calculation of Energy dissipated

distance;

if (distance>do)

S(i).E=S(i).E- ( (ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance\*distance\*distance ));

Energy\_CH\_I=(ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance\*distance\*distance );

end

if (distance<=do)

S(i).E=S(i).E- ( (ETX+EDA)\*(Packet) + Efs\*Packet\*( distance \* distance ));

Energy\_CH\_I=(ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance);

end

end

%Election of Cluster Heads for Advanced nodes

if( ( S(i).ENERGY==1 && ( temp\_rand <= ( padv / ( 1 - padv \* mod(r,round(1/padv)) )) ) ) )

countCHs=countCHs+1;

packets\_TO\_BS=packets\_TO\_BS+1;

PACKETS\_TO\_BS(r+1)=packets\_TO\_BS;

S(i).type='C';

S(i).G=100;

C(cluster).xd=S(i).xd;

C(cluster).yd=S(i).yd;

plot(S(i).xd,S(i).yd,'k\*');

distance=sqrt( (S(i).xd-(S(n+1).xd) )^2 + (S(i).yd-(S(n+1).yd) )^2 );

C(cluster).distance=distance;

C(cluster).id=i;

X(cluster)=S(i).xd;

Y(cluster)=S(i).yd;

cluster=cluster+1;

%Calculation of Energy dissipated

distance;

if (distance>do)

S(i).E=S(i).E- ( (ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance\*distance\*distance ));

Energy\_CH\_A=(ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance\*distance\*distance );

end

if (distance<=do)

S(i).E=S(i).E- ( (ETX+EDA)\*(Packet) + Efs\*Packet\*( distance \* distance ));

Energy\_CH\_A=(ETX+EDA)\*(Packet) + Emp\*Packet\*( distance\*distance);

end

end

end

end

end

%Checking average number of ClusterHeads per epoch

sum=sum+cluster;

count\_ch=count\_ch+1;

if count\_ch==10

Ave\_CH=(sum\*0.1)/(1+(m\*a)+(b\*x));

Throughput=Ave\_CH\*4;

STATISTICS(r+1).ave\_clustHd=Ave\_CH;

ave\_ch(r+1)=Ave\_CH;

STATISTICS(r+1).throughput=Throughput;

Clust\_throughput(r+1)=Throughput;

Ave\_CH=0;

sum=0;

count\_ch=0;

end

STATISTICS(r+1).CLUSTERHEADS=cluster-1;

CLUSTERHS(r+1)=cluster-1;

countmember=0;

%Election of Associated Cluster Head for Normal Nodes

for i=1:1:n

if ( S(i).type=='N' && S(i).E>0 )

if(cluster-1>=1)

min\_dis=9999;

min\_dis\_cluster=1;

for c=1:1:cluster-1

temp=min(min\_dis,sqrt( (S(i).xd-C(c).xd)^2 + (S(i).yd-C(c).yd)^2 ) );

if ( temp<min\_dis )

min\_dis=temp;

min\_dis\_cluster=c;

countmember=countmember+1;

end

CLUSTERMEM(r+1)=countmember;

end

countmember=0;

%Energy dissipated by associated Cluster Head

min\_dis;

if (min\_dis>do)

S(i).E=S(i).E- ( ETX\*(Packet) + Emp\*Packet\*( min\_dis \* min\_dis \* min\_dis \* min\_dis));

Energy\_member=ETX\*(Packet) + Emp\*Packet\*( min\_dis \* min\_dis \* min\_dis \* min\_dis);

end

if (min\_dis<=do)

S(i).E=S(i).E- ( ETX\*(Packet) + Efs\*Packet\*( min\_dis \* min\_dis));

Energy\_member=ETX\*(Packet) + Emp\*Packet\*( min\_dis \* min\_dis);

end

%Energy dissipated

if(min\_dis>0)

S(C(min\_dis\_cluster).id).E = S(C(min\_dis\_cluster).id).E- ( (ERX + EDA)\*Packet );

PACKETS\_TO\_CH(r+1)=n-dead-cluster+1;

end

S(i).min\_dis=min\_dis;

S(i).min\_dis\_cluster=min\_dis\_cluster;

end

end

end

hold on;

countCHs;

rcountCHs=rcountCHs+countCHs;

%finding live or dead node in each 4x4 grid

ni=20;

cell=zeros(5,5);

if alive ==100

for i=1:1:alive

for j=1:1:alive

u(i)=(S(i).xd);

x\_axis=int32(S(i).xd);

u\_grid(i)=idivide(x\_axis,ni,'ceil');

v(j)=(S(j).yd);

y\_axis=int32(S(j).yd);

v\_grid(j)=idivide(y\_axis,ni,'ceil');

end

end

for i =1:1:alive

%

if (Status(i) == 1)

ai= u\_grid(i);

if (ai == 0)

ai = 1;

end;

bi= v\_grid(i);

if (bi == 0)

bi = 1;

end;

cell(ai,bi)= cell(ai,bi)+ 1;

%

end

end

%stop

end

%Code for Voronoi Cells

%Unfortunately if there is a small

%number of cells, Matlab's voronoi

%procedure has some problems

warning('OFF');

[vx,vy]=voronoi(X,Y);

plot(X,Y,'r\*',vx,vy,'b-');

hold on;

voronoi(X,Y);

axis([0 xm 0 ym]);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% STATISTICS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %

% DEAD : a rmax x 1 array of number of dead nodes/round

% DEAD\_A : a rmax x 1 array of number of dead Advanced nodes/round

% DEAD\_N : a rmax x 1 array of number of dead Normal nodes/round

% CLUSTERHS : a rmax x 1 array of number of Cluster Heads/round

% PACKETS\_TO\_BS : a rmax x 1 array of number packets send to Base Station/round

% PACKETS\_TO\_CH : a rmax x 1 array of number of packets send to ClusterHeads/round

% first\_dead: the round where the first node died

% %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%