**Trust3.tcl**

# Define options

set val(chan) Channel/WirelessChannel

set val(prop) Propagation/TwoRayGround

set val(netif) Phy/WirelessPhy

set val(mac) Mac/802\_11

set val(ifq) Queue/DropTail/PriQueue

set val(ll) LL

set val(ant) Antenna/OmniAntenna

set val(ifqlen) 150

set val(nn) 25

set val(rp) AODV

set val(x) 800

set val(y) 500

set val(stop) 150

#-------Event scheduler object creation--------#

set ns [new Simulator]

#used to schedule the events that are running at the same time

$ns use-scheduler Heap

#Creating trace file and nam file

set tracefd [open aodv.tr w]

set namtrace [open aodv.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object

set topo [new Topography]

$topo load\_flatgrid $val(x) $val(y)

create-god $val(nn)

# configure the nodes

$ns node-config -adhocRouting $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \

-phyType $val(netif) \

-channelType $val(chan) \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON \

-macTrace OFF \

-movementTrace ON

# node creation

for {set i 0} {$i < $val(nn) } { incr i } {

set node\_($i) [$ns node]

$node\_($i) color "black"

$node\_($i) label Node$i

}

#initial location of nodes

for {set i 0} {$i < $val(nn)} {incr i} {

$node\_($i) set X\_ [expr rand()\*$val(x)]

$node\_($i) set Y\_ [expr rand()\*$val(y)]

$node\_($i) set Z\_ 0

$node\_($i) color "green"

}

for {set i 0} {$i < $val(nn)} {incr i} {

$ns at 0.1 "$node\_($i) color darkviolet"

$ns at 0.1 "$node\_($i) label Node$i"

}

#generation of movements

$ns at 2.0 "$node\_(0) setdest 20.0 10.0 0.0"

$ns at 5.0 "$node\_(24) setdest 60.0 50.0 50.0"

$ns at 7.0 "$node\_(2) setdest 120.0 100.0 50.0"

$ns at 0.0 "$node\_(17) setdest 140.0 150.0 50.0"

$ns at 0.0 "$node\_(20) setdest 160.0 180.0 50.0"

$ns at 12.0 "$node\_(19) setdest 130.0 230.0 6.0"

$ns at 23.0 "$node\_(5) setdest 40.0 165.0 9.0"

$ns at 23.0 "$node\_(6) setdest 95.0 160.0 0.0"

$ns at 20.0 "$node\_(7) setdest 50.0 180.0 13.0"

$ns at 3.0 "$node\_(4) setdest 80.0 156.0 5.0"

$ns at 6.0 "$node\_(10) setdest 56.0 200.0 10.0"

$ns at 13.0 "$node\_(15) setdest 88.0 20.0 45.0"

$ns at 6.0 "$node\_(3) setdest 177.0 120.0 45.0"

$ns at 23.0 "$node\_(1) setdest 140.0 70.0 20.0"

set fp2 [open "/home/user01/Documents/anu/input.txt" r]

set filed [read $fp2]

puts $filed

set data [split $filed "\n"]

puts $data

foreach line2 $data {

set length2 [string length $line2]

if {$length2 != 0} {

lassign $line2 src dest

puts $src

set tcp1 [new Agent/TCP/Newreno]

$tcp1 set class\_ 2

set sink1 [new Agent/TCPSink]

$ns attach-agent $node\_($src) $tcp1

$ns attach-agent $node\_($dest) $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ns at 3.0 "$node\_($src) color brown"

$ns at 3.0 "$node\_($dest) color blue"

$ns at 3.0 "$ftp1 start"

}

}

close $fp2

set fp1 [open "/home/user01/Documents/anu/mali.txt" r]

set file\_data [read $fp1]

set data1 [split $file\_data "\n"]

foreach line $data1 {

set length1 [string length $line]

if {$length1 !=0} {

set id $line

#$ns at 3.2 "[$node\_($id) set ragent\_] hacker"

$ns at 3.2 "$node\_($id) color red"

$ns at 3.2 "$node\_($id) label malicious"

}

}

close $fp1

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} { incr i } {

# 20 defines the node size for nam

$ns initial\_node\_pos $node\_($i) 20

}

# Telling nodes when the simulation ends

for {set i 0} {$i < $val(nn) } { incr i } {

$ns at $val(stop) "$node\_($i) reset";

}

# ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

#$ns at 149.01 "puts \"end simulation\" ; $ns halt"

proc stop {} {

global ns tracefd namtrace

$ns flush-trace

close $tracefd

close $namtrace

exec nam aodv.nam &

exit 0

}

#$ns at 150.01 "puts \"end simulation\" ; $ns halt"

$ns run

**Attempt.awk**

BEGIN{

for(i=0;i<100;i++)

a[i]=0;

}

{

if($1=="s")

{

for(j=0;j<100;j++)

{

if($3=="\_"j"\_")

{

a[j]=a[j]+1;

}

}

}

}

END {

for (key in a) {

print a[key]

}

}

**Duration.awk**

BEGIN{

for(i=0;i<100;i++)

a[i]=0;

for(i=0;i<100;i++)

b[i]=0;

}

{

if($1=="r")

{

for(j=0;j<100;j++)

{

if($3=="\_"j"\_")

{

a[j]=a[j]+$2;

b[j]=b[j]+1;

}

}

}

}

END {

for(j=0;j<100;j++){

a[j]=a[j]/b[j];

}

for (key in a) {

print a[key];

}

}

**Waiting.awk**

BEGIN{

for(i=0;i<100;i++)

a[i]=0;

for(i=0;i<100;i++)

b[i]=0;

}

{

if($1=="s")

{

for(j=0;j<100;j++)

{

if($3=="\_"j"\_")

{

if(b[j]==0)

{

b[j]=$2;

}

else

{

xx=$2-b[j];

b[j]=$2

a[j]=a[j]+xx;

}

}

}

}

}

END {

for (key in a) {

print a[key]"\n";

}

}

**Covariance.cpp**

#include <iostream>

#include <cmath>

#include <cstring>

#include <fstream>

using namespace std;

class StdDeviation

{

private:

int max;

double value[100];

double mean;

public:

double calmean()

{

double sum = 0;

for(int i = 0; i < max; i++)

sum += value[i];

return (sum / max);

}

double calvar()

{

mean = calmean();

double temp = 0;

for(int i = 0; i < max; i++)

{

temp += (value[i] - mean) \* (value[i] - mean) ;

}

return temp / max;

}

/\* double CalculateSampleVariane()

{

mean = CalculateMean();

double temp = 0;

for(int i = 0; i < max; i++)

{

temp += (value[i] - mean) \* (value[i] - mean) ;

}

return temp / (max - 1);

}\*/

int SetValues(double \*p, int count)

{

if(count > 100)

return -1;

max = count;

for(int i = 0; i < count; i++)

value[i] = p[i];

return 0;

}

double calstddev()

{

return sqrt(calvar());

}

/\* double Calculate\_SampleStandardDeviation()

{

return sqrt(CalculateSampleVariane());

}\*/

};

class FinanceCalculator

{

private:

double XSeries[100];

double YSeries[100];

StdDeviation x;

StdDeviation y;

public:

double sx,sy;

int max;

void SetValues(double \*xvalues, double \*yvalues, int count)

{

for(int i = 0; i < count; i++)

{

XSeries[i] = xvalues[i];

YSeries[i] = yvalues[i];

}

x.SetValues(xvalues, count);

y.SetValues(yvalues, count);

max = count;

}

double calcovar()

{

double xmean = x.calmean();

double ymean = y.calmean();

double total = 0;

for(int i = 0; i < max; i++)

{

total += (XSeries[i] - xmean) \* (YSeries[i] - ymean);

}

return total / max;

}

double calcor()

{

double cov = calcovar();

sx=x.calstddev();

sy=y.calstddev();

double corr = cov / ( sx \* sy);

return corr;

}

};

int main()

{

int i=0,j=0;

double dValue,dValue1,ysum=0,xsum=0,xsq=0,xx=0,xy=0,a,b;

double yarr[100],xarr[100],xyarr[100],xxarr[100];

ifstream infile;

ifstream infile1;

FinanceCalculator calc;

{

infile.open("valxx.txt");

infile1.open("valyy.txt");

infile >> dValue;

infile1 >> dValue1;

while(i<100)

{

xarr[i]=dValue;

yarr[i]=dValue1;

infile >> dValue;

infile1 >> dValue1;

i++;

}

int n=i;

for(int j=0;j<i;j++)

{

ysum=ysum+yarr[j];

}

for(int j=0;j<i;j++)

{

xsum=xsum+xarr[j];

}

xsq=xsum\*xsum;

for(int j=0;j<i;j++)

{

xxarr[j]=xarr[j]\*xarr[j];

}

for(int j=0;j<i;j++)

{

xx=xx+xxarr[j];

}

for(int j=0;j<i;j++)

{

xyarr[j]=xarr[j]\*yarr[j];

}

for(int j=0;j<i;j++)

{

xy=xy+xyarr[j];

}

a=(ysum\*xsq)-(xsum\*xy)/(n\*xsq)-xx;

b=(n\*xy)-(xsum\*ysum)/(n\*xx)-xsq;

printf("%f\n%f",a,b);

/\*calc.SetValues(xarr,yarr,sizeof(xarr) / sizeof(xarr[0]));

double cor=calc.calcor();

double cov=calc.calcovar();

printf("Covariance = %.10lf\n", cov);

printf("Correlation = %.10lf\n", cor);

double m=cor\*(calc.sx/calc.sy);

printf("slope=%f\n",m);

if(cor>0.5)

{

if(m>0.5)

printf("greedy node is not exist in this network");

else

printf("run watchdog tool");

}

else

{

printf("run watchdog tool");

}

\*/

}

}

**FuzzyAttempt.cpp**

#include <iostream>

#include <cmath>

#include <cstring>

#include <fstream>

using namespace std;

class CTrapezoid

{

private:

double dLeftMiddle, dRightMiddle;

double dLeft, dRight,t;

double mid,mid1,mid2;

public:

void setMiddle(double dL, double dR)

{

dLeftMiddle=dL; dRightMiddle=dR;

}

void setinterval(double l,double r)

{

dLeft=l;

dRight=r;

}

void middle()

{

mid=(dLeftMiddle+dRightMiddle)/2;

mid1=(dLeftMiddle+mid)/2;

mid2=(mid+dRightMiddle)/2;

cout<<mid<<"\n";

cout<<mid1<<"\n";

cout<<mid2<<"\n";

}

double getValuex(double t)

{

if(t<=dLeftMiddle){

return 0;}

else if(t<mid && t>dLeftMiddle){

//cout<<((t-dLeftMiddle)/(mid-dLeftMiddle));

if(t<mid1)

return 0;

else

return 0.5;

}

else if(t>mid && t<dRightMiddle)

{

//cout<<((t-mid)/(dRightMiddle-mid));

if(t>mid2)

return 1;

else

return 0.5;

}

else if(t>=dRightMiddle){

return 1;}

else if(t>=mid1 && t<=mid2){

return 0.5;}

else

return 0;

}

};

main()

{

CTrapezoid FuzzySet;

double dLeft, dRight,dL,dR;

dLeft=0;

dRight=6862;

dL=274.48;

dR=5146.5;

FuzzySet.setinterval(dLeft,dRight);

FuzzySet.setMiddle(dL,dR);

FuzzySet.middle();

double dValue;

ifstream infile;

infile.open("attempt.txt");

ofstream outfile;

outfile.open("attemptout.txt");

cout << "Reading from the file" << endl;

infile >> dValue;

int c=0;

while(infile != NULL)

{

cout << dValue<< endl;

//if((dValue>=dLeft)&&(dValue<=dRight))

//cout<<"In the interval";

//else

//cout<<"Not in the interval";

// cout<<endl;

double x=FuzzySet.getValuex(dValue);

if(x==1)

{

cout<<"high";

outfile <<"high\n";

}

else if(x==0)

{

cout<<"low";

outfile <<"low\n";

}

else{

cout<<"mid";

outfile <<"mid\n";

}

c=c+1;

infile >> dValue;

}

}

**FuzzyDuration.cpp**

#include <iostream>

#include <cmath>

#include <cstring>

#include <fstream>

using namespace std;

class CTrapezoid

{

private:

double dLeftMiddle, dRightMiddle;

double dLeft, dRight,t;

double mid,mid1,mid2;

public:

void setMiddle(double dL, double dR)

{

dLeftMiddle=dL; dRightMiddle=dR;

}

void setinterval(double l,double r)

{

dLeft=l;

dRight=r;

}

void middle()

{

mid=(dLeftMiddle+dRightMiddle)/2;

mid1=(dLeftMiddle+mid)/2;

mid2=(mid+dRightMiddle)/2;

cout<<mid<<"\n";

cout<<mid1<<"\n";

cout<<mid2<<"\n";

}

double getValuex(double t)

{

if(t<=dLeftMiddle){

return 0;}

else if(t<mid && t>dLeftMiddle){

//cout<<((t-dLeftMiddle)/(mid-dLeftMiddle));

if(t<mid1)

return 0;

else

return 0.5;

}

else if(t>mid && t<dRightMiddle)

{

//cout<<((t-mid)/(dRightMiddle-mid));

if(t>mid2)

return 1;

else

return 0.5;

}

else if(t>=dRightMiddle){

return 1;}

else if(t>=mid1 && t<=mid2){

return 0.5;}

else

return 0;

}

};

main()

{

CTrapezoid FuzzySet;

double dLeft, dRight,dL,dR;

dLeft=0;

dRight=150;

dL=5;

dR=89;

FuzzySet.setinterval(dLeft,dRight);

FuzzySet.setMiddle(dL,dR);

FuzzySet.middle();

double dValue;

ifstream infile;

infile.open("duration.txt");

ofstream outfile;

outfile.open("durationout.txt");

cout << "Reading from the file" << endl;

infile >> dValue;

int c=0;

while(infile != NULL)

{

double x=FuzzySet.getValuex(dValue);

if(x==1)

{

cout<<"high\n";

outfile<<"high\n";

}

else if(x==0)

{

cout<<"low\n";

outfile<<"low\n";

}

else

{

cout<<"mid\n";

outfile<<"mid\n";

}

infile >> dValue;

}

}

**FuzzyWaiting.cpp**

#include <iostream>

#include <cmath>

#include <cstring>

#include<fstream>

using namespace std;

class CTrapezoid

{

private:

double dLeftMiddle, dRightMiddle;

double dLeft, dRight,t;

double mid,mid1,mid2;

public:

void setMiddle(double dL, double dR)

{

dLeftMiddle=dL; dRightMiddle=dR;

}

void setinterval(double l,double r)

{

dLeft=l;

dRight=r;

}

void middle()

{

mid=(dLeftMiddle+dRightMiddle)/2;

mid1=(dLeftMiddle+mid)/2;

mid2=(mid+dRightMiddle)/2;

cout<<mid<<"\n";

cout<<mid1<<"\n";

cout<<mid2<<"\n";

}

double getValuex(double t)

{

if(t<=dLeftMiddle){

return 0;}

else if(t<mid && t>dLeftMiddle){

//cout<<((t-dLeftMiddle)/(mid-dLeftMiddle));

if(t<mid1)

return 0;

else

return 0.5;

}

else if(t>mid && t<dRightMiddle)

{

//cout<<((t-mid)/(dRightMiddle-mid));

if(t>mid2)

return 1;

else

return 0.5;

}

else if(t>=dRightMiddle){

return 1;}

else if(t>=mid1 && t<=mid2){

return 0.5;}

else

return 0;

}

};

main()

{

CTrapezoid FuzzySet;

double dLeft, dRight,dL,dR;

dLeft=0;

dRight=15430;

dL=617.2;

dR=11572.5;

FuzzySet.setinterval(dLeft,dRight);

FuzzySet.setMiddle(dL,dR);

FuzzySet.middle();

double dValue;

ifstream infile;

infile.open("waiting.txt");

ofstream outfile;

outfile.open("waitingout.txt");

cout << "Reading from the file" << endl;

infile >> dValue;

while(infile != NULL)

{

double x=FuzzySet.getValuex(dValue);

if(x==1)

{

outfile<<"low\n";

cout<<"low\n";

}

else if(x==0)

{

cout<<"high\n";

outfile<<"high\n";

}

else

{

cout<<"mid\n";

outfile<<"mid\n";

}

infile >> dValue;

}

}

**Final.cpp**

#include <iostream>

#include <cmath>

#include <cstring>

#include<fstream>

using namespace std;

int main() {

int i=0;

string d1,d2,d3,cc;

int d11;

char c,c1;

ifstream infile1,infile2,infile3;

infile1.open("attemptout.txt");

infile2.open("durationout.txt");

infile3.open("waitingout.txt");

ofstream out;

out.open("final.txt");

infile1 >> d1;

infile2 >> d2;

infile3 >> d3;

while(i<100)

{

if(d1=="low")

{

if(d2=="low")

{

cout<<"Normal\n";

cc="Normal";

}

}

if(d1=="low")

{

if(d2=="mid")

{

cout<<"Normal\n";

cc="Normal";

}

}

if(d1=="low")

{

if(d2=="high")

{

cout<<"Suspected\n";

cc="Suspected";

}

}

if(d1=="mid")

{

if(d2=="low")

{

cout<<"Normal\n";

cc="Normal";

}

}

if(d1=="mid")

{

if(d2=="mid")

{

cout<<"Suspected\n";

cc="Suspected";

}

}

if(d1=="mid")

{

if(d2=="high")

{

cout<<"Greedy\n";

cc="Greedy";

}

}

if(d1=="high")

{

if(d2=="low")

{

cout<<"Suspected\n";

cc="Suspected";

}

}

if(d1=="high")

{

if(d2=="mid")

{

cout<<"Suspected\n";

cc="Suspected";

}

}

if(d1=="high")

{

if(d2=="high")

{

cout<<"Greedy\n";

cc="Greedy";

}

}

cout<<"ssssssssssssss";

if(cc=="Normal")

{

if(d3=="low")

{

cout<<"Normal\n";

out <<"Normal\n";

}

}

if(cc=="Normal")

{

if(d3=="mid")

{

cout<<"Normal\n";

out <<"Normal\n";

}

}

if(cc=="Normal")

{

if(d3=="high")

{

cout<<"Suspected\n";

out <<"Suspected\n";

}

}

if(cc=="Suspected")

{

if(d3=="low")

{

cout<<"Suspected\n";

out <<"Suspected\n";

}

}

if(cc=="Suspected")

{

if(d3=="mid")

{

cout<<"Suspected\n";

out <<"Suspected\n";

}

}

if(cc=="Suspected")

{

if(d3=="high")

{

cout<<"Greedy\n";

out <<"Greedy\n";

}

}

if(cc=="Greedy")

{

if(d3=="low")

{

cout<<"Greedy\n";

out <<"Greedy\n";

}

}

if(cc=="Greedy")

{

if(d3=="mid")

{

cout<<"Greedy\n";

out <<"Greedy\n";

}

}

if(cc=="Greedy")

{

if(d3=="high")

{

cout<<"Greedy\n";

out <<"Greedy\n";

}

}

infile1 >> d1;

infile2 >> d2;

infile3 >> d3;

i++;

}

return 0;

}