بسم الله الرحمن الرحیم

set val(chan) Channel/WirelessChannel
به معنی تنظیم شبکه بیسیم برای این شبیه سازی است که این تنظیم در متغیر chan تنظیم میشود.

set val(prop) Propagation/TwoRayGround
Propagation یا انتشار رادیویی TwoRayGround یک مدل انتشار رادیویی چند راهی است که تلفات یا losses مسیر بین آنتن فرستنده و آنتن گیرنده را هنگامی که در قابل دید هستند پیش بینی می کند.

set val(netif) Phy/WirelessPhy
وظیفه کلاس WirelessPhy یک واسط ارسال بسته به کانال و دریافت بسته از کانال است.

set val(mac) Mac/802_11
تنظیم کردن استاندارد 802_11 که استاندارد IEEE 802.11 ، معروف به WiFi ، معماری و مشخصات شبکه های بی سیم (WLAN) را تعیین می کند. WiFi یا WLAN از امواج رادیویی با فرکانس بالا به جای کابل برای اتصال دستگاه ها در LAN استفاده می کند. کاربران متصل به شبکه های WLAN می توانند در محدوده پوشش شبکه حرکت کنند

set val(ifq) Queue/DropTail/PriQueue
Tail drop یک الگوریتم ساده برای مدیریت صف است که توسط تجهیزات شبکه برای تصمیم گیری در مورد بسته های از دست رفته استفاده می شود.

PriQueue بسته را به صورت اولویت دار مدیریت می کند

set val(ll) LL

تنظیم لایه پیوند به متغیر ll

set val(ant) Antenna/OmniAntenna

تنظیم نوع آنتن
Omnidirectional antenna در ارتباطات رادیویی ، آنتن همه جهته به دسته ای از آنتن گفته می شود که از هر جهت عمود بر یک محور قدرت رادیویی برابر ساطع می کند

set val(ifqlen) 50

تنظیم بیشترین بسته در صف ifq

set val(nn) 5

تنظیم تعداد نود شبکه

set val(rp) AODV

تنظیم پروتکل مسیریابی
پروتکل مسیریابی AODV برای استفاده توسط گره های تلفن همراه mobile nodes در یک شبکه ad hoc در نظر گرفته شده است.

set val(x) 876
set val(y) 100
set val(stop) 50.0

تنظیم محیط x و y شبیه سازی و تنظیم زمان اجرای شبیه سازی

```
set ns [new Simulator]
```
در این قسمت یک رویداد زمانبندی تنظیم میکنیم که در کد های شبیه ساز تعریف میشود که رویداد در چه زمانی باید اجرا شود !

```
set topo       [new Topography]
```
در این قسمت یک توپولوژی جدید برای این شبیه سازی ایجاد میشود

```
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
```

در این قسمت برای این توپولوژی یک محیط گرید یا توری شکل با محیط x و y که در ابتدا ایجاد کرده بودیم ایجاد میکنیم و بعد تعداد نود های این شبکه را تنظیم میکنیم که در این شبیه سازی ۵ تا است

```
set tracefile [open out.tr w]
$ns trace-all $tracefile
```

برای شبیه سازی حتما باید دو فایل tr و nam ایجاد بشوند که ابتدا فایل out.tr برای نوشتن تریس ها یا پیگیری ها یا لوگ ها ایجاد و در متغیر tracefile تنظیم میشود

```
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

فایل out.nam که جهت ثبت و ذخیره سازی نتایج حاصل از شبیه سازی و ردیابی نمودن صف به همین شکل ایجاد و در متغیر namfile تنظیم میشود

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

این دستور برای مقداردهی اولیه فایل out.nam برای ثبت حرکات گره مورد استفاده قرار می گیرد تا در نم مشاهده شود

```
set chan [new $val(chan)]
```
ایجاد یک شبکه بی سیم

```
$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) \
        -channel       $chan \
        -topoInstance  $topo \
        -agentTrace    OFF \
        -routerTrace   OFF \
        -macTrace      ON \
        -movementTrace ON
```
در این قسمت تنظیماتی که در بالا تنظیم کرده بودیم برای کل نود ها تنظیم می کنیم و به متغیر ns میدهیم

```
set n1 [$ns node]
$n1 set X_ 605
$n1 set Y_ 519
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
```
در ادامه نود ها را ایجاد میکنیم در داخل محیط و در نقطه ای با فاصله این نود ها قرار میگیرند
عدد ۲۰ هم نشان دهنده سایز نودها در nam است

```
set udp10 [new Agent/UDP]
set null01 [new Agent/Null]
```
در شبکه بی سیم باید نوع ارتباط گره ها را با هم تنظیم کنیم که برای عامل ارسال از udp و دریافت از null استفاده میکنیم

```
$ns attach-agent $n1 $udp10
```
بعد از تنظیم باید نود ها را به این عامل ها متصل کنیم
```
$udp10 set packetSize_ 1500
```
بعد سایز بسته هایی که این عامل ها می توانند انتقال دهند را تنظیم میکنیم

```
set cbr10 [new Application/Traffic/CBR]
$cbr10 attach-agent $udp10
$cbr10 set packetSize_ 1000
$cbr10 set rate_ 1.0Mb
$cbr10 set random_ null
```
در ادامه باید بسته های خود را ایجاد کنیم که با CBR این کار را انجام میدهیم CBR بسته ها را در یک نرخ ثابتی ایجاد میکنه بعد این بسته ها را به عامل های ارتباطی متصل میکنیم و سایز بسته ها و bitrate یا نرخ ارسال آن ها را تنظیم میکنیم و random هم که یک پرچم جهت مشخص کردن وجود یا عدم وجود پارازیت تصادفی در مدت زمان ارسال ترافیک است که در این جا غیر فعال است

```
set energylist(0) 100
set energylist(1) 100
set energylist(2) 100
set energylist(3) 100
set MaxEnergyNode 0
set timer 0.0
```
در این قسمت یک سریع مقادیر اولیه را تنظیم میکنیم که اول انرژی کل گره ها را در ابتدا ۱۰۰ می گذاریم و بیشترین نود در شروع را ۰ تنظیم می کنیم و تایمر شبیه سازی را در ابتدا ۰ تنظیم میکنیم

```
$ns connect $udp10 $null01
```
بعد عامل ها فرستنده و گیرنده را به هم متصل میکنیم

```
leach
```
در ادامه تابع لیچ اجرا میشه که در ادامه توضیح داده میشود
```
for {set i 0} {$i < $val(nn) } { incr i } {
        $ns at $val(stop) "\$n$i reset"
}
```
بعد عمل شبیه سازی در یک فور به کل نود ها پایان شبیه سازی اعلان میشود

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

```
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
```
و در ادامه پایان شبیه سازی اعلان و تابع finish اجرا میشود
```
proc finish {} {
        global ns tracefile namfile
        $ns flush-trace
        close $tracefile
        close $namfile
        exec nam out.nam &
        exit 0
}
```
که در این تابع بعد گرفتن tracefile و namfile این فایل ها را میبندیم دستور flush-trace هم بافر ردیاب یا tracer را تخلیه میکند و معمولا در پایان شبیه سازی اجرا میشود دستور exec هم بعد فهماندن این مفاهیم به شبیه ساز حالا این مفاهیم را اجرا و به نمایش درمی آورد و برنامه nam را برای مشاهده شبیه سازی اجرا می کنیم و دستور exit اجرای شبیه سازی را پایان میدهد

```
$ns run
```
زمانبند شبیه ساز با این دستور اجرا میشود و نود ها با هم ردوبدل بسته انجام میدهند.

توابع اصلی

تابع leach

```
proc leach {} {
        global timer timer1 MaxEnergyNode energylist
```
در این قسمت ابتدا متغیر هایی که در خارج تابع ایجاد شدند گرفته میشود
```
        while {$timer<50} {
                setcluster
```
و تا زمان مقرر شده تابع setcluster در هر دور برای خوشه بندی اجرا می شود

```
                if [expr $MaxEnergyNode==1] {
                        sendPackets1
```
در هر دور MaxEnergyNode چک میشود و سرخوشه با این روش انتخاب میشود و در نودی که انرژی بیشتری داشته باشه به عنوان سرخوشه انتخاب میشود و نود ها با اون سرخوشه فقط رد و بدل بسته دارن و بعد اجرا انرژی آن سرخوشه کم میشود ! در این شبیه سازی نود ۰ را به دلایلی که خود تعریف کردیم در این خوشه بندی ها به عنوان سرخوشه انتخاب نمی کنیم .

```
                } elseif [expr $MaxEnergyNode==2] {
                        sendPackets2
                } elseif [expr $MaxEnergyNode==3] {
                        sendPackets3
                } elseif [expr $MaxEnergyNode==4] {
                        sendPackets4
                }

                puts "$energylist(0) $energylist(1) $energylist(2) $energylist(3) "
```
دستور puts جهت ارسال یک خروجی برای چاپ به صفحه نمایش و چه در یک فایل استفاده میشود . (برای فایل تریس !)

```
        }
}
```

تابع setcluster

```
proc setcluster {} {
        global energylist MaxEnergyNode
        if {$energylist(0)>=$energylist(1) && $energylist(0)>=$energylist(2) &&
$energylist(0)>=$energylist(3)} {
                set MaxEnergyNode 1
        } elseif {$energylist(1)>=$energylist(0) && $energylist(1)>=$energylist(2) &&
$energylist(1)>=$energylist(3)} {
                set MaxEnergyNode 2
        } elseif {$energylist(2)>=$energylist(0) && $energylist(2)>=$energylist(1) &&
$energylist(2)>=$energylist(3)} {
                set MaxEnergyNode 3
        } elseif {$energylist(3)>=$energylist(0) && $energylist(3)>=$energylist(2) &&
$energylist(3)>=$energylist(1)} {
                set MaxEnergyNode 4
        }
}
```

در این تابع بعد گرفتن انرژی کل و لیست انرژی های نود ها ، چک میشوند که کدام نود از بقیه انرژی بیشتری دارد و بعد این عمل نود که انرژی بیشتری دارد به عنوان سرخوشه انتخاب میشود

تابع ارسال بسته های برای نود ۱ : اگر نود ۱ سرخوشه بود !

```
proc sendPackets1 {} {
        global ns cbr10 cbr21 cbr31 cbr41 timer energylist n1 n2 n3 n4
```

مقادیر را از توابع سراسری میگیرد

```
        $ns at [expr 0.0+$timer] "$n2 color blue"
        $n2 color "blue"
```

در زمان ۰ نود ۲ رنگش آبی است یعنی نودی که سرخوشه نیست

```
        $ns at [expr 0.0+$timer] "$n3 color blue"
        $n3 color "blue"
        $ns at [expr 0.0+$timer] "$n4 color blue"
        $n4 color "blue"
        $ns at [expr 0.0+$timer] "$n1 color red"
        $n1 color "red"
```

و نود ۱ چون سرخوشه است رنگش قرمزه !

```
        $ns at [expr 0.0+$timer] "$cbr21 start"
```

بسته ای از نود ۲ به نود ۱ ارسال میشود در زمان ۰ !

```
        $ns at [expr 0.5+$timer] "$cbr21 stop"
        $ns at [expr 1.0+$timer] "$cbr10 start"
        $ns at [expr 1.5+$timer] "$cbr10 stop"
        $ns at [expr 2.0+$timer] "$cbr31 start"
        $ns at [expr 2.5+$timer] "$cbr31 stop"
        $ns at [expr 3.0+$timer] "$cbr10 start"
```

در زمان ۳ بسته ای از نود ۱ به نود ۰ ارسال میشود

```
        $ns at [expr 3.5+$timer] "$cbr10 stop"
        $ns at [expr 4.0+$timer] "$cbr41 start"
        $ns at [expr 4.2+$timer] "$cbr41 stop"
        $ns at [expr 4.5+$timer] "$cbr10 start"
        $ns at [expr 4.8+$timer] "$cbr10 stop"
        set timer [expr $timer+5.0]
```
برای تایمر کلی شبیه سازی این اعمال ارسال بسته ها زمان ۵ صرف می شود
```
        puts "1 $timer"
```
چاپ زمان فعلی در فایل رد یاب یا trace
```
        set energylist(0) [expr $energylist(0)-9]
```
تنظیم کردن انرژی برای نود ۱ با کاهش انرژی به مقدار ۹
```
}
```

بقیه تابع های ارسال بسته به شکل بالا هستند !

خسته نباشید