# Variables globales

finalizo-encaminamiento? ;bandera que indica si se establecieron los parametros de encaminamiento

paquetes-enviados ;total de paquetes enviado por los nodo fuentes

paquetes-recibidos ;total de paquetes recibidos por el sink

nodos-muertos

# Variables de las tortugas

mensaje ;-1: no hay mensajes en el buffer, >0: mensaje de encaminamiento (correponde al id “who” del padre); -2: hay mensajes de datos en el buffer

mensaje-dato ;al ejecutar el algoritmo PI envía el nivel del padre

nivel ;distancia (en saltos) al sink

tipo ;sink, fuente ó nodo

estado ;dormido (nunca desperto),activo (está funcionando), muerto (sin batería)

contador-mensajes-recibidos

contador-mensajes-enviados

energia

buffer ;número de mensajes en buffer

.

# setup: Métodos de Inicialización

## clear-all

Reinicia todas las variables globales a 0 y llama a los siguientes métodos: clear-ticks, clear-turtles, clear-patches, clear-drawing, clear-all-plots y clear-output

## setup-globals

Incializa las varibles globales:

* set finalizo-encaminamiento? False
* set paquetes-enviados 0
* set paquetes-recibidos 0
* set nodos-muertos numero-de-nodos

## setup-patches

Inicializa el fondo de color gris

ask patches

[

set pcolor gray

]

## setup-nodes

### Inicializa las variables de los nodos

set-default-shape turtles "circle"

crt numero-de-nodos

[

;se multiplca por 0.95 para que ningun nodo quede cerca de los ejes

setxy (random-xcor \* 0.95) (random-ycor \* 0.95)

set color 3

set size 0.5

set mensaje -1

set tipo "nodo"

set estado "dormido"

set contador-mensajes-recibidos 0

set contador-mensajes-enviados 0

set energia 100

set buffer 0

]

### Establece fuentes

* Establece número de fuentes
* Llama a setup-fuentes

to setup-fuente

set shape "circle 2"

set size 1

set tipo "fuente"

set energia 200

end

### Establece el sink

* Llama a setup-sink

to setup-sink

set shape "target"

set size 1

set mensaje who

set mensaje-dato 0

set nivel 0

set tipo "sink"

set estado "despierto"

end

### Establece la topología de la red

* Llama a setup-topologia

to setup-topologia

ask turtles

[

;let tortuga-vecina - a la variable tortuga-vecina le asigna

;one-of (...) - un elemento de (...)

;other turtles - el conjunto de tortugas, excepto la que ejecuta al procedimeinto

;with [not link-neighbor? myself] - que no estan conectadas con la tortuga que ejecuta al procedimiento

repeat grado-promedio

[

let tortuga-vecina one-of (other turtles in-radius (cobertura-promedio \* 0.34) with [not link-neighbor? myself])

;si la tortuga obtenida no tiene enlace con la tortuga que ejecuta el procedimiento

if tortuga-vecina != nobody

[

;crea un enlace con la tortuga elegida

create-link-with tortuga-vecina

]

]

]

ask links [set color white]

end

* Llama a reset-ticks: reinicia el contador de ticks y las gŕaficas

# go: Métodos de ejecución

## go-encaminamiento

Con un algoritmo de propagación de la información establece los parámetros del cada nodo:

* estado de “dormido” a “despierto”
* nivel, distancia en saltos hacía el sink

to go-encaminamiento

if not finalizo-encaminamiento?

[

print "Se estan estableciendo parametros de encaminamiento..."

ask turtles

[

recibi-mensaje?

]

if all? turtles [mensaje = -1]

[

set finalizo-encaminamiento? True

print "Los parametros de encaminamiento se han establecido con exito."

]

]

end

## go-fuentes

Genera mensajes: continuamente ó por mensaje

* Si ya se terminaron de establecer los parámteros de encaminamiento
  + generacion por paquetes: llama a generar-paquete y desactiva la fuente
  + generacion continua: llama a generar-paquete

to go-fuentes

if activar-fuentes and finalizo-encaminamiento?

[

if generacion-de-paquetes = "por-paquete"

[

generar-paquete

set activar-fuentes False

]

if generacion-de-paquetes = "continua"

[

generar-paquete

]

]

end

## go-nodos

Hace que todos los nodos con estado “despierto” ejecuten el procedimiento recibi-mensaje

ask turtles with [estado != "despierto"]

[

set label buffer ;VER el buffer de los nodos en el arbol

recibi-mensaje?

]

## tick

Aumenta el contador de ticks y con ello actualiza las gráficas

# Otros Métodos

## envia-mensaje

* Si es llamado desde setup-encaminamiento, envia mensaje > 0 (variable who de su padre), entonces envia a sus vecinos que no ha sido visitados su varible who, su nivel también lo envía (en la variable mensaje-dato, de su vecino)
* Si es llamado desde go-fuentes u un nodo que quiere reenviar un mensaje, entonces

1. Llama al procedimiento protocolo-encaminamiento, para obtener el who del vecin al que enviara el mensaje de datos
2. Si el protocolo de encaminamiento le devuelve un destino y el control de congestión está en siga
   1. saca del buffer el paquete que reenviara
   2. Aumenta el buffer de su vecino
   3. le indica a su vecino que tiene un mensaje haciendo su variable mensaje = -2
   4. Aumenta su contador local de mensajes-enviados
   5. Llama al procemiento consume energia

to envia-mensaje

**;envia mensaje para establecer parametros del protocolo de encaminamiento a sus vecinos que no han sido visitados**

if mensaje >= 0

[

let emisor who

let emisor-nivel nivel

ask link-neighbors with [estado = "dormido"]

[

set mensaje emisor

set mensaje-dato emisor-nivel

set estado "despierto"

]

set mensaje -1

]

**;envia mensaje de datos**

if mensaje = -2

[

let nodo-destino protocolo-encaminamiento

if is-turtle? nodo-destino and buffer > 0 and control-de-congestion

[

saca-paquete-del-buffer

ask link-with nodo-destino

[

set color red

]

ask nodo-destino

[

set mensaje -2

set buffer buffer + 1

]

set contador-mensajes-enviados contador-mensajes-enviados + 1

consume-energia

]

]

## Recibi mensaje

* Si es llamado desde setup-encamianmiento
  1. Actualiza sus parametros de encaminamiento setup-encaminamiento
  2. Reenvía el mensaje envia-mensaje
* Si el mensaje es de datos
  1. Llama al procedmiento procesa-mensaje-de-datos
     + Si no es el sink reenvia el mensaje con envia-mensaje

to recibi-mensaje?

;si el mensaje es para establecer los parametros de encaminamiento

if mensaje >= 0

[

set-parametros-encaminamiento

envia-mensaje

]

;si el mensaje es de datos

if mensaje = -2

[

procesa-mensaje-de-datos

if tipo != "sink"

[

envia-mensaje

]

]

end

## set-parametros de encaminamiento

Establece un "protocolo de encaminamiento", crea un arbol por medio del algoritmo de propagacion de la informacion

* Establece en el nivel
* Estable ce el color verde por tener bateria llena
* Dado que el nodo desperto reduce la variable nodos-muertos

to set-parametros-encaminamiento

set nivel mensaje-dato + 1

set color lime

set nodos-muertos nodos-muertos - 1

end

## generar-paquete

Es llamado solo por nodos fuente

1. indica que hay mensajes en buffer <mensaje = -2>
2. aumenta la variable global “paquetes-enviados=paquetes-enviados+1”
3. aumenta la cantidad de mensajes en el buffer “buffer=buffer+1”
4. llama al procedimiento envia-mensaje

## procesa-mensaje-de-datos

1. Aumenta su contador local de mensajes recibidos contador-mensajes-recibidos
2. Si el nodo es sink
   * Aumenta la variable global paquetes-recibidos
   * disminuye en 1 la variable buffer
   * Si ya no hay mensajes en el buffer, pone la variable mensaje = -1

to procesa-mensaje-de-datos

set contador-mensajes-recibidos contador-mensajes-recibidos + 1

if tipo = "sink"

[

set paquetes-recibidos paquetes-recibidos + 1

saca-paquete-del-buffer

]

end

## protocolo-encaminamiento

Devuelve el who del nodo destino, el cual tiene un nivel menor al nodo que llama el procemineito

to-report protocolo-encaminamiento

let mi\_nivel nivel

report one-of link-neighbors with [nivel < mi\_nivel and estado != "muerto"]

end

## control-de-congestion

devuelve falso verdadero para saber si enviar o no un mensaje

to-report control-de-congestion

report true

end

## saca-paquete-del-buffer

* reduce la cantidad de paquetes en buffer
* Indica cuando el buffer esta vacío con la variable mensaje = -1

to saca-paquete-del-buffer

set buffer buffer - 1

if buffer = 0

[

set mensaje -1

]

end

consume-energia

* dismunuye la cantidad de energpia indicada en la variable energia-por-transmicion
* cambia los colores de los nodos segun la energia (verde, amarillo, rojo y gris)
* si se acaba la energia
  + el nodo se pinta de gris
  + su estado cambia a “muerto”
  + aumenta la variable global nodos-muertos

to consume-energia

set energia energia - energia-por-transmision

if energia > 35 and energia < 70

[

set color yellow

]

if energia > 0 and energia <= 35

[

set color red

]

if energia < 0

[

set color gray

set estado "muerto"

set nodos-muertos nodos-muertos + 1

]

end