



PROYECTO DE GRADO

Presentado ante la ilustre UNIVERSIDAD DE LOS ANDES como requisito final para
obtener el Título de INGENIERO DE SISTEMAS

SIMULADOR PARA PROTOCOLO DE ENRUTAMIENTO QUE PROVEE ANONIMATO BASADO EN ALGORITMOS DE OPTIMIZACIÓN DE COLONIAS DE HORMIGAS ARTIFICIALES

Por

Br. David Santiago Cadavid Ardila

Tutor: MSc. Rodolfo Sumoza

Febrero 2016

©2016 Universidad de Los Andes, Mérida, Venezuela



PROYECTO DE GRADO CALIFICACIÓN FINAL

Título del Proyecto de Grado: **"Simulador para Protocolo de Enrutamiento que provee Anonimato Basado en Algoritmos de Optimización de Colonias de Hormigas Artificiales"**

Bachiller: **DAVID SANTIAGO, CADAVID ARDILA**

C.I. 19.144.886

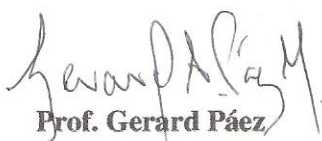
1) Calificación del (de la) Profesor(a) Tutor(a):	20	(20%)
2) Calificación del manuscrito final:	20	(40%)
3) Calificación de la defensa oral:	20	(40%)
Calificación final:	20	(puntos)


Los suscritos miembros del Jurado asignan como calificación final del Proyecto de Grado la nota de:

20	(puntos)	Veinte	(puntos)
Números		Letras	

Observaciones: Los Miembros del Jurado calificador sugerimos que el presente trabajo sea accesible a la comunidad Universitaria en general, a través de la publicación de un artículo científico, donde se destaquen los principales resultados de este trabajo. En consecuencia le otorgamos **MENCIÓN PUBLICACIÓN**.


Prof. Rodolfo Sumoza
Tutor


Prof. Gerard Pérez
Jurado


Prof. Junior Altamiranda
Jurado



Simulador para protocolo de enrutamiento que provee Anonimato basado en algoritmos de optimización de colonias de hormigas artificiales

Br. David Santiago Cadavid Ardila

Proyecto de Grado — Sistemas Computacionales

Escuela de Ingeniería de Sistemas, Universidad de Los Andes, 2016

Resumen: El Anonimato en Internet, procura evitar que a los usuarios se les profile a través de relaciones con sus acciones y vínculos con otros usuarios mediante técnicas de análisis de tráfico. Para proveer Anonimato existe un protocolo que usa algoritmos de colonias de hormigas artificiales para reducir la latencia que se crea en los sistemas de este tipo. En este trabajo se presentan el diseño, arquitectura e implementación de un simulador para dicho protocolo, el cual se realizó como un módulo de NS3.

Se realizaron pruebas piloto comparativas con y sin la optimización que brindan los algoritmos ACO con el fin de validar el funcionamiento de todos sus componentes y se verificaron las salidas generadas, las cuales eran acorde a lo esperado.

Palabras clave: Simulación, Anonimato, colonias de hormigas artificiales.

El Proyecto de Grado titulado “**Simulador para protocolo de enrutamiento que provee Anonimato basado en algoritmos de optimización de colonias de hormigas artificiales**”, realizado por Br. **David Santiago Cadavid Ardila**, C.I. N° 19.144.886, fue presentado el día 24 de febrero de 2016, en el salón de reuniones de la Escuela de Sistemas, ante el Jurado evaluador conformado por:

Tutor: MSc. Rodolfo Sumoza

Jurado: Dr. Gerard Paez

Jurado: Dr. Junior Altamiranda

A Dios
a mi familia
y a Albert en el cielo

Índice

Índice de tablas	IX
Índice de figuras	X
Agradecimientos	XI
1. Introducción	1
1.1. Antecedentes	2
1.2. Planteamiento del problema	3
1.3. Objetivos	4
1.3.1. Objetivo general	4
1.3.2. Objetivos específicos	4
1.4. Metodología	4
1.5. Alcance	5
2. Marco teórico	6
2.1. Simulación por eventos discretos	6

2.1.1. Terminología	7
2.2. Anonimato	8
2.2.1. Análisis de tráfico	8
2.3. Optimización a través de colonias de hormigas artificiales	8
2.3.1. AntNet	10
2.4. Protocolo a simular	11
2.4.1. Topología	12
2.4.2. Enrutamiento	13
2.4.3. Tablas de probabilidad	13
2.4.4. Hormigas	14
2.4.5. Funcionamiento	15
2.5. NS3	17
2.5.1. Estructura	17
2.5.2. Conceptos básicos	18
2.5.3. Instalación	19
2.5.4. Ejecución	20
2.5.4.1. <code>configure</code>	21
2.5.4.2. <code>build</code>	21
3. Simulador <i>ARAP</i>	22
3.1. Estructura de directorios en NS3	23

3.2. Topología	24
3.3. Parámetros	25
3.3.1. Archivo de configuración	25
3.3.2. Configuración del simulador	26
3.4. ArapNode	27
3.4.1. Envío de hormigas exploradoras	27
3.4.2. Envío de hormigas de carga	28
3.4.3. Tabla de enrutamiento	31
3.4.4. Recepción de hormigas	32
3.4.4.1. Peticiones	32
3.4.4.2. Respuestas	32
3.5. Estructura de las hormigas	33
3.6. Manejo de estadísticas	35
3.6.1. Hormigas exploradoras	36
3.6.2. Estadísticas de hormigas de carga	39
3.7. Arquitectura	40
3.8. Ejecución	49
3.8.1. <code>configure</code> y <code>debug</code>	49
3.8.2. <code>run</code>	50
3.9. Resultados	51

4. Pruebas y resultados	53
4.1. Pruebas	53
4.1.1. Parámetros de las simulaciones	54
4.2. Resultados	57
5. Conclusiones y recomendaciones	63
5.1. Conclusiones	63
5.2. Recomendaciones	64
Apéndices	
A. Descripción de parámetros	67
B. Cómo agregar nuevos parámetros	80
C. Especializar ArapPathManager	82
Referencias bibliográficas	84

Índice de tablas

2.1. Tabla de probabilidad en red de cuatro nodos	14
3.1. Tabla de enrutamiento de nodo A	31
4.1. Valores de parámetros globales usados en las pruebas	54
4.2. Valores de parámetros locales usados en las pruebas	55
4.3. Cantidad de hormigas de carga creadas en las pruebas realizadas	58
4.4. Promedio de las medias (en segundos) de los tiempos de respuesta de las hormigas de carga en las pruebas realizadas	59
4.5. Varianza de las medias (en segundos cuadrados) de los tiempos de respuesta de las hormigas de carga en las pruebas realizadas	60
4.6. Extracto de archivo que contiene caminos de hormigas de carga	61

Índice de figuras

2.1. Proceso de estigmergia en una colonia de hormigas	10
2.2. Topología de malla completa	12
2.3. Diagrama de funcionamiento del enrutamiento cebolla	13
2.4. Estructura de componentes que conforman NS3	18
3.1. Estructura de directorios del simulador ARAP	23
3.2. Topología estrella usada por el simulador	24
3.3. Estructura de hormiga con tres (3) capas	34
3.4. Estructura genérica de DATA en las hormigas del simulador ARAP . .	34
3.5. Diagrama relacional de clases del simulador ARAP	41
3.6. Diagrama de clase ArapSimulator	42
3.7. Diagrama de clase ArapNode	43
3.8. Diagrama de clase ArapAnts	44
3.9. Diagrama de las clases RoutingTableRow y LoadAntsStatistics	45
3.10. Diagrama de las clases ArapPathManager y PathManagerFactory . . .	46
3.11. Diagrama de las clases SmartPathManager y ExplorerAntsStatistics . .	47

Agradecimientos

Llegar a este punto del camino ha sido resultado de un arduo trabajo y esfuerzo. Ha sido un largo recorrido, en el cual pude conocer maravillosas personas que pusieron su grano de arena y me ayudaron a dar muchos de los pasos que me trajeron hasta aquí, al final de este camino que no es más que el comienzo de uno nuevo, una nueva etapa con nuevas pruebas y oportunidades.

Agradezco primeramente a Dios por darme la vida, la salud y los medios para lograr esta meta.

A mi madre Martha y mi hermano Carlos quienes estuvieron siempre a mi lado dándome su apoyo, especialmente en los momentos difíciles.

A mis amigos Arcelia, Luis, Johan, Roger y muchos más con quienes he compartido experiencias no solo académicas sino de vida.

A los profesores que tuvieron la dedicación y la paciencia para enseñarme y para aclarar mis muchas dudas, de los cuales menciono sin algún orden particular a los profesores: Rafael Rivas, Gerard Paez, Demián Gutierrez, y de manera especial a mi tutor Rodolfo Sumoza, quien durante los últimos meses me ha brindado toda su ayuda siempre con la mejor disposición y ánimo. A todos agradezco su dedicación, y su aporte hacia mí no solo en lo académico sino a nivel personal.

Por último pero no menos importante, a mi Luz, por su apoyo y su energía, que aún con la distancia podía sentir a mi lado.

Capítulo 1

Introducción

El Internet ha evolucionado de gran manera desde sus inicios permitiendo actualmente la comunicación de usuarios alrededor del mundo de manera rápida y sencilla, transmitiendo diferentes tipos de datos a través de toda la red. Estas comunicaciones generalmente se realizan en redes seguras y no seguras, desde las cuales se puede conocer no sólo la información que se está transmitiendo, sino que debido a las propiedades de las comunicaciones se puede relacionar o vincular a los usuarios con las acciones que realizan en el Internet mediante técnicas de análisis de tráfico [1].

Desde sus inicios estas técnicas han sido usadas por los administradores de red para identificar y facilitar la resolución de distintos problemas, como pérdida de paquetes, identificación de cuellos de botella, calidad de servicio, entre otros [2]. En la actualidad empresas como Google, Facebook o Amazon, entre otras, las usan para crear perfiles de los usuarios con fines de diversa índole: comerciales, calidad de servicio, entre otros.

El principal problema para los usuarios con respecto al análisis de tráfico recae en que se está obteniendo de manera directa o indirecta información privada sin su consentimiento, lo cual viola su derecho a la privacidad [3]. Así mismo existen situaciones en las que por protección el usuario no desea que se le relacione directamente con ciertas comunicaciones, por ejemplo un informante de una empresa u organización.

Es a raíz de esto que nace el tema del Anonimato.

En este proyecto de grado se detalla la creación de un simulador para un protocolo que ofrece Anonimato mediante la adaptación de algoritmos de *colonias de hormigas artificiales*, ésto inicialmente ameritó el estudio de algunos simuladores de redes de comunicación de datos creados previamente, de los cuales se seleccionó el simulador de redes de propósito general **NS3**, principalmente por su popularidad en el área de estudio y el amplio soporte que tiene por parte de la comunidad.

NS3 es un simulador de eventos discretos, actualmente se encuentra en la versión 3.24.1, cuenta con documentación, tutoriales y manuales de usuario que facilitan el aprendizaje para aquellos que deseen usar la herramienta.

El documento a continuación se estructura de la siguiente manera: En el capítulo uno (1) se plantean los objetivos y las justificaciones para la realización del proyecto; en el capítulo dos (2) se exponen las bases teóricas sobre las que se define el protocolo y el simulador; en el capítulo tres (3) se explica la arquitectura del simulador y el funcionamiento de los distintos componentes que lo conforman; el capítulo cuatro (4) detalla las pruebas realizadas para validar el correcto funcionamiento del simulador y los resultados de las mismas; en el capítulo cinco (5) se realizan conclusiones sobre el proyecto realizado y se ofrecen recomendaciones para trabajos futuros. En los apéndices se detallan los parámetros definidos para el simulador y se describen métodos para extender las funcionalidades del mismo y/o cambiar algunos comportamientos.

1.1. Antecedentes

Los protocolos propuestos hasta ahora, para poder proveer Anonimato aumentan la latencia en las comunicaciones, debido a que los datos deben tomar caminos no directos, lo cual incrementa los tiempos de respuesta. Una posible solución a este problema lo representa el protocolo para el cual se creó el simulador, el cual fue planteado en [4]; este protocolo busca por medio de los algoritmos de optimización de colonias de hormigas artificiales ofrecer una mejora en el enrutamiento realizado en

cuanto a los tiempos de respuesta (latencia), conociendo y tendiendo a usar los mejores caminos entre todos los nodos disponibles de la red anónima. En este caso el factor que indica que un camino es mejor que otro es la latencia.

En [5] y [6] se definen las bases teóricas del funcionamiento de los algoritmos de optimización de colonias de hormigas artificiales y cómo han sido usados para resolver distintos tipos de problemas de enrutamiento, también se especifica un modelo para la simulación de los mismos. Cabe decir que hasta ahora, estos algoritmos no han sido usados en protocolos que provean Anonimato.

En [7] se explica la creación de un simulador de eventos discretos enfocado en la capa de aplicación, según el modelo OSI. En éste también se ofrece un modelo de pruebas para diferentes topologías de red. En [8] se realiza la implementación de un simulador usando NS3 y se explican sus principios de funcionamiento, su estructura y el formato de la salida de datos que pueden arrojar las simulaciones realizadas.

1.2. Planteamiento del problema

Cuando se desea agregar o quitar alguna característica que cambie el funcionamiento de un sistema, es primordial validar cómo estos cambios afectan el sistema en general y qué partes del mismo específicamente, en ocasiones no es posible o conveniente realizar las pruebas sobre el mismo sistema por distintos motivos, por ejemplo puede ser muy costoso, y esto a su vez conlleva a pérdidas materiales, o porque sencillamente no son posibles debido a las características del sistema u otros factores. Estas son unas de las principales razones del uso de las herramientas de simulación. En la actualidad existe una carencia de herramientas para probar propuestas basadas en sistemas emergentes que provean Anonimato, como el protocolo mencionado en este trabajo, es por esto que se desarrolló un simulador que permita probar y validar protocolos de red en este contexto.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un simulador para un protocolo de capa aplicación que provee Anonimato a través de la adaptación de algoritmos de optimización de colonias de hormigas artificiales.

1.3.2. Objetivos específicos

- Modelar el protocolo a simular según la estructura de trabajo de NS3.
- Implementar (codificar) el modelo realizado en NS3.
- Realizar pruebas piloto en el simulador creado.

1.4. Metodología

El proyecto se puede estructurar en dos partes: La creación del simulador y las pruebas piloto sobre el mismo. Para la primera parte se realizaron las siguientes actividades:

- Estudiar la forma cómo se crean y funcionan los distintos módulos de NS3.
- Configurar el ambiente de trabajo.
- Crear el modelo del protocolo para el simulador.
- Codificar los distintos componentes del modelo en NS3.

Para la segunda etapa se realizaron las siguientes actividades:

- Diseñar un conjunto de pruebas piloto para validar el funcionamiento del simulador.
- Realizar las pruebas diseñadas.

1.5. Alcance

En este proyecto se creó un simulador para el protocolo mencionado anteriormente, el cual se diseñó de tal manera que permitiera configurar cada simulación a realizar, a su vez posee una arquitectura que permite extender su comportamiento ya sea agregando nuevos parámetros y/o definiendo nuevas formas de actualizar las tablas de probabilidad y de crear los caminos en base a éstas ¹.

Así mismo se realizaron pruebas piloto para verificar la estabilidad y correcto funcionamiento del sistema creado. Se realizaron comparaciones entre el funcionamiento de protocolos optimizados y protocolos sin la optimización que proveen los algoritmos de colonias de hormigas artificiales. Sólo se llegó a probar el simulador, con el fin de permitir que en trabajos futuros se hagan las pruebas formales del protocolo.

¹Las guías para usar y extender el simulador según lo mencionado se encuentran en los apéndices al final del documento.

Capítulo 2

Marco teórico

En este capítulo se dan a conocer algunos conceptos y bases teóricas que sirven para introducir al lector en el tema de investigación. Se empieza por mencionar algunas nociones referentes a la simulación, siguiendo con conceptos específicos al tema como lo son los *algoritmos ACO* y *Anonimato*. Luego se da una descripción general del protocolo a simular y se termina con una descripción de la herramienta usada para desarrollar el simulador.

2.1. Simulación por eventos discretos

La simulación se ha convertido en una herramienta de gran utilidad en muchos campos de investigación que permite estudiar un sistema según un modelo del mismo. La finalidad de la simulación es principalmente verificar hipótesis de los modelos a simular, así como también predecir el comportamiento de un sistema, identificar distintas variables y características del mismo, todo esto sin tener que trabajar con el sistema real, sea un sistema físico como una línea de producción de una planta o un sistema lógico como una red de computadores, de hecho es posible que el sistema que se desee simular no exista.

En este trabajo se usó la simulación por eventos discretos. Este tipo de simulación se usa con aquellos modelos en los cuales el estado del sistema está definido para instantes particulares de tiempo, y donde dicho estado sólo cambia durante esos instantes, los cuales están definidos por la ocurrencia de un evento [9].

2.1.1. Terminología

Evento

Acción que genera un cambio en el estado del sistema, una simulación por eventos discretos está conformada por una serie de eventos que se ejecutan de manera ordenada cronológicamente, y finaliza cuando no hay más eventos por simular o cuando se alcanza el tiempo máximo de simulación [10].

Variables de estado

Es el conjunto de variables, cuyos valores en un tiempo determinado definen el estado del sistema [10].

Tiempo de simulación

Este tiempo hace referencia al reloj interno de la simulación, mediante el cual se rige el orden de ejecución de los eventos de la simulación. Este reloj generalmente trabaja con un tiempo acelerado, es decir, un segundo de simulación no equivale a un segundo de tiempo real [10].

Condiciones iniciales

Son los valores que se le da a las variables de estado al momento de iniciar la simulación, estos valores pueden haberse obtenido de la observación o de experimentos previos, siendo usados en el segundo caso como una forma de *reanudar* una simulación a partir de un tiempo dado, siempre y cuando se conozca el estado del sistema en dicho tiempo [11].

2.2. Anonimato

El Anonimato en Internet, consiste en un conjunto de protocolos, mecanismos y sistemas que procuran proteger a los usuarios de ataques basados en el análisis de tráfico. Fundamentalmente busca impedirle al atacante el poder relacionar un usuario con sus acciones en la red. Cabe destacar que las técnicas de Anonimato dependen del control que tenga el atacante sobre la red, por ejemplo, ninguno de los protocolos existentes hasta ahora puede ofrecer Anonimato contra un atacante que controle toda la red [12]. Con el análisis de tráfico, se pueden registrar las acciones de los usuarios y en base a esos registros se pueden crear perfiles de éstos sin su consentimiento. Al aplicar técnicas de Anonimato, se impide en cierta medida la creación de estos perfiles [13]. Se han estudiado diferentes estrategias que ofrecen protección a este tipo de ataques, siendo uno de los ejemplos más populares la red **TOR** [14].

2.2.1. Análisis de tráfico

El análisis de tráfico consiste en técnicas con las cuales se analiza e infiere información a partir de las características del tráfico de las comunicaciones [15]. Para obtener información se puede basar por ejemplo en el origen y destino de las comunicaciones, su tamaño, su frecuencia, patrones de comunicación, etcétera.

2.3. Optimización a través de colonias de hormigas artificiales

El estudio de las sociedades de insectos, y en especial de las colonias de hormigas ha llevado a descubrir que la manera en que éstas se comunican para realizar tareas de manera sincronizada, como recolectar comida desde un lugar dado al nido, se basa en estímulos químicos (feromonas) que liberan en el aire. Este proceso de colaboración a través del medio físico se conoce como *estigmergia*, y se puede describir de la siguiente

manera:

- Varias hormigas (exploradoras) salen del nido a buscar comida y cada una toma caminos aleatorios, dejando a su paso un rastro de feromona.
- Cuando una hormiga encuentra comida, empieza un recorrido de vuelta al nido por el mismo camino que llegó, liberando a su vez la misma feromona, de esta manera el camino recorrido por dicha hormiga ahora posee una concentración mayor de feromonas.
- Cuando salen más hormigas del nido, éstas ya no seleccionan un camino totalmente aleatorio, sino que son guiadas por las feromonas dejadas por las hormigas anteriores, y éstas a su vez continúan dejando el rastro de feromonas y por ende fortaleciendo el camino seleccionado.
- En el caso de los otros caminos que no son tan transitados, el rastro de feromonas se va evaporando, y al no ser reforzado le indica a las demás hormigas que dichos caminos no son óptimos.
- Luego de cierto tiempo se aprecia que las hormigas comienzan a recorrer un único camino de manera ordenada, el cual contiene una mayor concentración de feromonas respecto a los otros.

En la figura 2.1 se muestra el estado del proceso en tres momentos distintos: Al principio cuando la primera hormiga exploradora sale en busca de alimento, un tiempo después cuando se encuentran varias hormigas recorriendo distintos caminos y por último un punto en el que las hormigas convergen a un mismo camino.

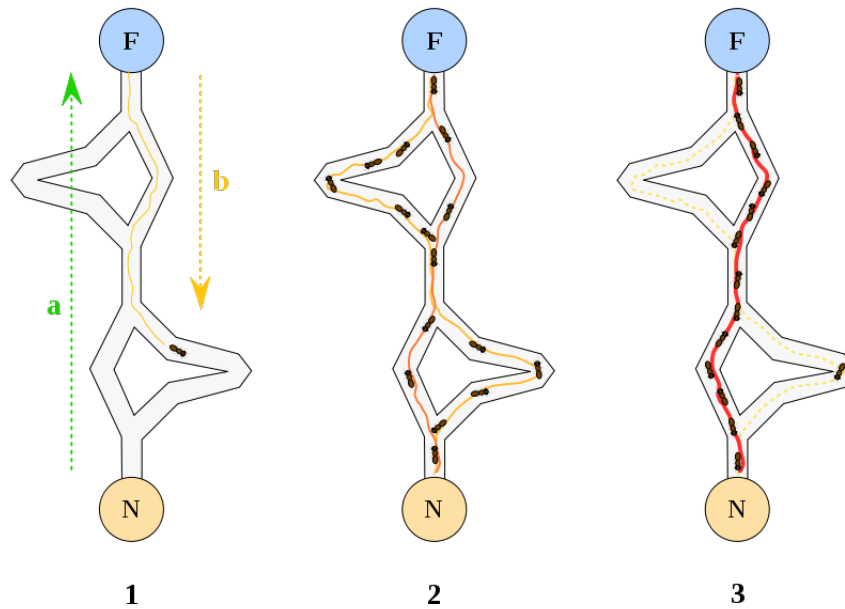


Figura 2.1: Proceso de estigmergia en una colonia de hormigas ¹

En 1992 Marco Dorigo propuso un algoritmo basado en este comportamiento, el cual se usa para encontrar el camino óptimo en un grafo, y a partir de éste se han implementado diferentes variaciones para resolver distintos tipos de problemas, como el problema del viajero y otros problemas de optimización combinatoria.

2.3.1. AntNet

Es un algoritmo *ACO* ² usado para enrutamiento de tráfico de red, el objetivo de usarlo es mantener un estado actualizado de la red en cada nodo, y con la información obtenida poder seleccionar buenos caminos ³ para los paquetes que viajan a través de dicha red. Su funcionamiento se puede describir de la siguiente manera:

- En intervalos regulares de tiempo, cada nodo envía *hormigas artificiales* hacia

¹Autor: Johann Dréo. 27 de mayo de 2006

²Ant Colony Optimization, que en español se traduce a Optimización por Colonias de Hormigas

³Aquellos cuya latencia sea menor en comparación al resto

nodos destino, dichos destinos no son inmediatos sino que requieren cierto número de saltos para llegar a ellos.

- Cada hormiga artificial busca un camino de costo mínimo entre su origen y el destino, para hacer esto se mueve un nodo a la vez, y en cada nodo selecciona el siguiente basado en una regla heurística. Es mediante esta heurística y la memoria de la hormiga que se simula el proceso de estigmergia.
- Mientras se mueven entre nodos, las hormigas van almacenando en su memoria información respecto al tiempo de viaje, congestión en los nodos, y los identificadores de los nodos que conforman el camino recorrido.
- Cuando una hormiga llega al destino comienza el camino de regreso a su origen (nido) a través del mismo camino recorrido pero en dirección contraria, durante este recorrido se van actualizando los modelos locales de cada nodo (perteneciente al camino) según los valores almacenados en la hormiga respecto al camino y la calidad del mismo.
- Una vez que una hormiga llega de vuelta a su nido, ésta desaparece del sistema.

2.4. Protocolo a simular

Como se ha mencionado anteriormente el simulador realizado en este trabajo está diseñado en torno al protocolo propuesto en [4]. El cual es un protocolo de enrutamiento a nivel de capa aplicación según el modelo OSI, descrito en [16]. Este protocolo está diseñado para proteger contra atacantes que controlen el sistema de comunicación, por ejemplo una compañía de comunicaciones, pero no ofrece protección contra un atacante que controle toda la red, es decir el sistema de comunicación y los nodos pertenecientes a la red anónima. Para comprender su funcionamiento es necesario comentar antes algunos puntos importantes respecto al mismo:

2.4.1. Topología

El protocolo está diseñado para funcionar en una red de nodos *P2P*⁴ llamada *Red Anónima*, en ésta, cada nodo conoce y mantiene una conexión lógica con todos los demás nodos de la red, formando de esta manera una topología de malla completa como la que se muestra en la figura 2.2.

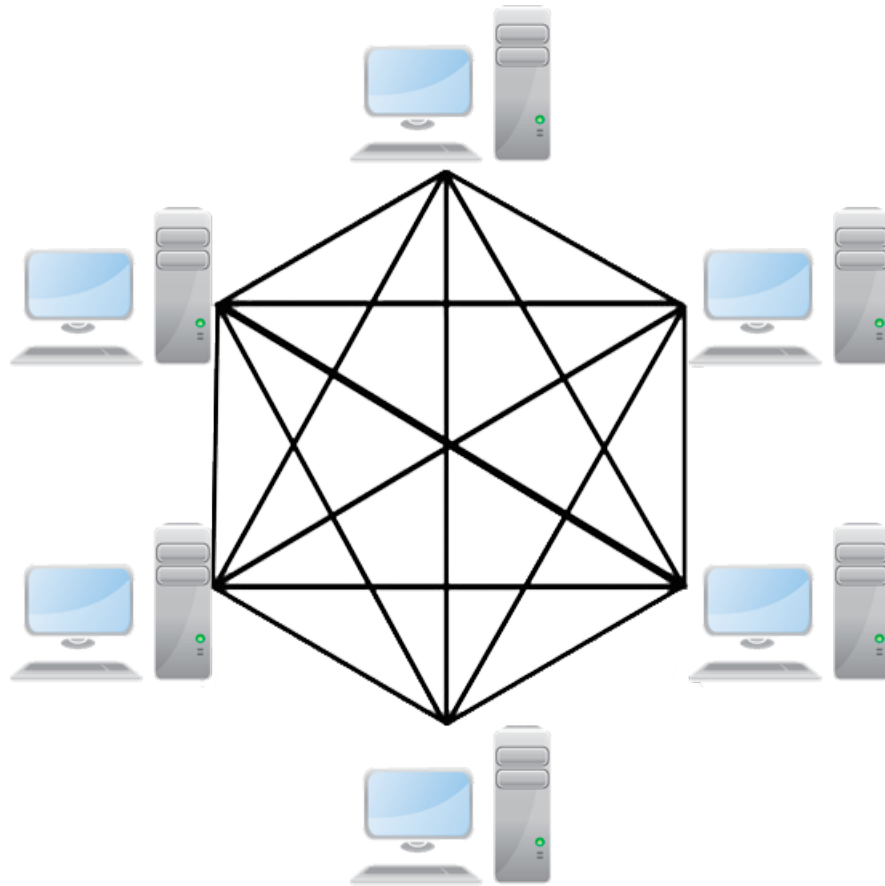


Figura 2.2: Topología de malla completa

⁴Peer to Peer. Se llama así a las redes donde cada nodo actúa como cliente y servidor a la vez

2.4.2. Enrutamiento

El protocolo está basado en la técnica para proveer Anonimato llamada *Onion Routing*⁵, la cual consiste en encapsular los mensajes en capas encriptadas (similares a las capas de una cebolla), los cuales luego se envían a través de varios nodos (llamados *enrutadores cebolla*), donde en cada uno se va eliminando una capa, hasta que llegue al nodo destino y en éste se puede ver el mensaje original que se envió desde el origen. En la figura 2.3 se puede ver la estructura descrita para un enrutamiento que consiste de tres (3) saltos: Dos enrutadores y el destino. El Anonimato se da ya que cada nodo sólo conoce el nodo anterior y el siguiente en el camino, y sólo los nodos origen y destino saben quienes son ellos respectivamente, de esta manera no es posible conocer todo el camino recorrido por los mensajes (a menos que un atacante tenga control de toda la red).

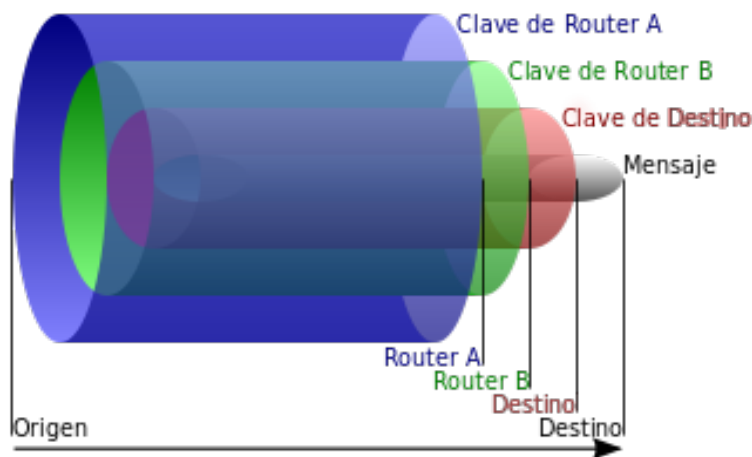


Figura 2.3: Diagrama de funcionamiento del enrutamiento cebolla

2.4.3. Tablas de probabilidad

La mejora que propone el protocolo respecto a otros como TOR, se basa en la selección de caminos utilizando distribuciones de probabilidad, en comparación a

⁵Enrutamiento Cebolla

una selección totalmente aleatoria. Para esto cada nodo debe mantener una tabla de probabilidad, la cual se usa para seleccionar los nodos pertenecientes a las rutas por las que se envían los mensajes para llegar de un origen a un destino.

Esta tabla se actualiza mediante el uso del algoritmo ACO e indica los distintos nodos por los que puede pasar un mensaje para llegar a un destino específico, cada valor de la tabla es una probabilidad, y cada fila es una distribución de probabilidad. Si se usan los índices i y j para identificar las filas y columnas respectivamente, el valor del elemento T_{ij} indica la probabilidad de pasar por el nodo N_j para llegar al nodo N_i . Por ejemplo en la tabla 2.1, el elemento T_{13} (en negrita) indica que hay una probabilidad de 59% de seleccionar a **D** como nodo intermedio para enviar un mensaje desde **A** hasta **B**. Se puede apreciar que las probabilidades en la diagonal principal siempre son 0, esto ocurre para evitar que se hagan envíos directos, por ejemplo, si desde **A** se desea enviar un mensaje a **B**, **B** no puede pertenecer al camino (de manera implícita ya pertenece como nodo final) ya que viola las reglas del enrutamiento cebolla y no se estaría proveyendo Anonimato.

	B	C	D	Σ
B	0	0.41	0.59	1
C	0.5	0	0.5	1
D	0.3	0.7	0	1

Tabla 2.1: Tabla de probabilidad de nodo **A**, en una red de 4 nodos **A,B,C,D**

2.4.4. Hormigas

Ya se había mencionado que el protocolo hace uso de algoritmos ACO, los cuales introducen el concepto de *agentes*, también llamados hormigas en este contexto. Es en éstas que se encapsulan los mensajes y se envían a través de la red anónima. Para efectos del protocolo se manejan dos tipos de hormigas: *exploradoras* y *de carga*. Ambos tipos de hormigas son del mismo tamaño y su estructura es prácticamente igual, cambiando

sólo la carga que transportan.

Los hormigas exploradoras son usadas con el fin de obtener información sobre la red y de esta manera actualizar las tablas de probabilidad de cada uno de los nodos. Mientras que las hormigas de carga son las que transportan los mensajes desde el origen al destino; cada vez que sea necesario enviar una, ésta debe obtener un camino, para lo cual se hace uso de las tablas de probabilidad.

2.4.5. Funcionamiento

Ya explicados los puntos anteriores se puede resumir fácilmente el funcionamiento del protocolo de la siguiente manera:

- Se considera la red P2P Anónima de N nodos.
- Se realizan las configuraciones iniciales, y cada nodo solicita la lista de los demás nodos y sus claves públicas de encriptación [17].
- Se inicializan las tablas de probabilidad con probabilidad $\frac{1}{(N-2)}$ ⁶
- Cada nodo envía hormigas exploradoras periódicamente, las cuales exploran todos los posibles caminos de *dos* (2) saltos para ir desde un nodo origen a un nodo destino a través de los distintos nodos intermedios (Origen \rightarrow Intermedio \rightarrow Destino). Se usan dos saltos para mostrar el envío indirecto de mensajes sin pérdida de generalidad. Para trabajos futuros, será el diseño de experimentos que indique el número de saltos requeridos.

La exploración de caminos en un nodo **A** se realiza según los siguientes pasos, los cuales son análogos para todos los nodos de la red:

1. **A** fija a un nodo **C** como *destino*.

⁶Para inicializar las probabilidades de manera uniforme. El denominador $N-2$ corresponde a todos los nodos excepto al que pertenece la tabla y el que actúa como destino en cada fila

2. **A** fija un nodo **B** como *intermedio*, y crea una hormiga que recorra el camino $\mathbf{A} \rightarrow \mathbf{B} \rightarrow \mathbf{C}$, luego repite este paso cambiando el nodo intermedio hasta que ha usado a todos los nodos como intermedios (excepto **A** y **C** que ya son origen y destino respectivamente).
3. **A** cambia el nodo destino (**C**) por otro nodo cuyos caminos no han sido explorados (que no sea **A**) y repite los pasos anteriores hasta que haya explorado todos los caminos para todos los destinos posibles en la red anónima.

Cuando una exploradora vuelve al nodo donde se originó (*nido*), se calcula el tiempo de viaje de ésta y se actualiza la tabla de probabilidad de dicho nodo. Siguiendo con la notación anterior, se actualizaría la tabla del nodo A en la posición T_{CB}

- Cuando un nodo desea enviar un mensaje de manera anónima, se debe verificar si el mensaje es más grande que el tamaño máximo de carga de la hormiga, de ser así, se fragmenta el mensaje en varias partes y se envía en varias hormigas, cada parte del mensaje incluye un número de secuencia, a través del cual se podrá reensamblar el mensaje en el nodo final.
- Cada hormiga que se envía debe tomar un camino, el cual es tan largo como la cantidad de saltos con que esté configurado el protocolo. El camino se selecciona junto con la tabla de probabilidad de manera que:
 - Ningún nodo se repita en el camino.
 - El nodo origen no puede pertenecer al camino.
 - No puede usarse el nodo final como nodo intermedio.
- Cada vez que una hormiga llega a un nodo intermedio, éste remueve una capa de la cebolla, la cual contiene información necesaria para continuar enviando la hormiga hasta el destino.
- Una vez que llegan todas las partes del mensaje al destino final, éste se reensambla.

- Para enviar la respuesta, se usa el mismo camino recorrido pero en sentido contrario, para esto se hace uso de tablas de enrutamiento en cada nodo, las cuales almacenan la información de cada hormiga respecto al nodo anterior y el nodo siguiente a recorrer.

Para proveer protección de los datos transmitidos cada capa de la cebolla que se crea en las hormigas debe ir encriptada, para esto se usan métodos de *encriptamiento asimétrico PKI*, los cuales están conformados por pares de claves públicas y privadas, las públicas son conocidas por toda la red y se usan principalmente para encriptar, mientras que cada clave privada sólo es conocida por el nodo al que pertenece y permiten desencriptar el fragmento de mensaje que se encriptó con la clave pública correspondiente. [17]

2.5. NS3

Es un simulador de redes por eventos discretos, está desarrollado en *C++* bajo licencia *GNU GPLv2*. Principalmente se usa para fines educativos y de investigación [18]. Permite simular múltiples tipos de redes, como wifi, punto a punto, entre otras, y a su vez permite controlar detalles de la simulación, lo cual da al investigador bastante control y libertad.

2.5.1. Estructura

NS3 es un conjunto de bibliotecas y clases que modelan los elementos que conforman distintas redes, tales como dispositivos de red inalámbricos, canales de transmisión, protocolos de red, aplicaciones, etcétera.

Se puede describir usando un modelo por capas como el mostrado en la figura 2.4, donde en la capa inferior se encuentran las clases que manejan los comportamientos básicos del simulador: La cola de eventos, el avance del reloj de simulación, flujos de

variables aleatorias, etcétera. En las siguientes capas se definen los modelos generales de redes, como nodos, protocolos de Internet y sockets. En las capas adyacentes se encuentran los módulos específicos para ciertos protocolos o aplicaciones, por ejemplo un servidor de envío de mensajes que usa TCP. Es en esta capa donde se realizó el desarrollo del simulador, el cual de manera simplificada se puede ver como un módulo nuevo que extiende la funcionalidad de NS3.

Se puede extender el modelo a capas para identificar clases a alto nivel que se usan a modo de *wrapper*⁷ y las pruebas que posee cada módulo para validar su correcto funcionamiento de manera independiente.

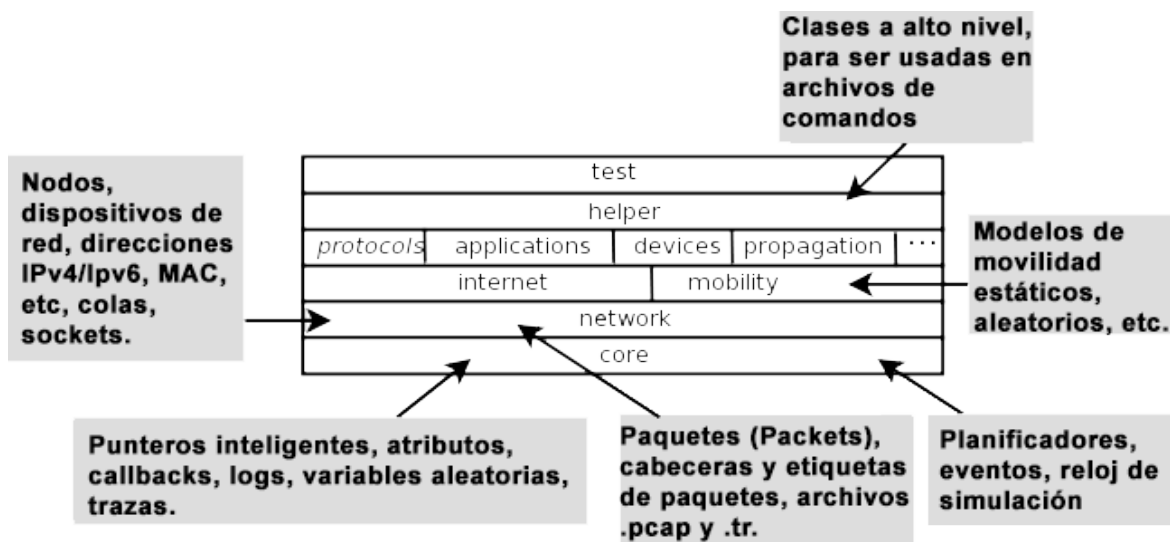


Figura 2.4: Estructura de componentes que conforman NS3

2.5.2. Conceptos básicos

NS3 contiene abstracciones de elementos fundamentales que conforman una red, de los cuales cabe destacar algunos de los siguientes conceptos ya que se estarán usando a lo largo del trabajo.

Node

⁷Patrón de diseño usado para *envolver* varios procedimientos en un conjunto simplificado de funciones y reducir la cantidad de código a escribir

Abstracción del dispositivo de cómputo básico. Es básicamente la abstracción de un computador. Al que se le instalan aplicaciones, pilas protocolares y dispositivos.

Application

Abstracción básica de un programa de usuario que genera alguna actividad a ser simulada. Por ejemplo un programa que envíe mensajes periódicamente y otro que los reciba y envíe una respuesta según el mensaje.

Packet

Implementación de buffer de bytes que permite que se pasen datos a través de las distintas API de NS3, en vez de tener que recurrir al uso de punteros base de C++.

Net-Device

Abstracción de los dispositivos de red y sus controladores. Los Net-Device se instalan en los nodos, un mismo nodo puede poseer varios de éstos (al igual que en el caso de uso real).

Channel

Medio de conexión y transmisión de los datos. Se conectan a los dispositivos de red, los cuales a su vez se conectan a los nodos.

Topology Helpers

Clases de ayuda que facilitan la creación de distintas topologías de red comunes, como es la *topología estrella*.

2.5.3. Instalación

La instalación de NS3 consiste en obtener el código fuente de la página oficial del software, y compilarlo. No es necesario instalarlo en una carpeta específica del sistema, NS3 puede ser ejecutado desde la carpeta donde se compile.

El software está desarrollado principalmente para sistemas *GNU/Linux*, y a pesar que existe cierto soporte para otros sistemas, no se garantiza el funcionamiento correcto de todos los módulos y características que ofrece.

Los requisitos mínimos del sistema para realizar simulaciones básicas son: Algún compilador de C++ (puede ser *Gcc/G++* v3.4 o superior) y un intérprete de *Python* v2.4 o superior, sin embargo si se desean habilitar funciones avanzadas u otras características que ofrece el software (como el módulo NetAnim usado para visualizar la red a simular en tiempo real) se deben instalar paquetes extra, los cuales no se detallan aquí, pero si el lector lo desea puede verificar la wiki oficial de NS3 [19] para más detalles.

En resumen, para garantizar que todos los módulos y características de NS3 funcionen sin problemas al momento de usarlas, se debe usar el siguiente comando para instalar los paquetes necesarios. Por ejemplo en un ambiente que use **apt-get** como gestor de paquetes se usa:

```
sudo apt-get install gcc g++ python python-dev mercurial
bzip2 gdb valgrind gsl-bin libgsl0-dev libgsl0ldbl flex
bison tcpdump sqlite sqlite3 libsqlite3-dev libxml2
libxml2-dev libgtk2.0-0 libgtk2.0-dev uncrustify
doxygen graphviz imagemagick texlive texlive-latex-extra
texlive-generic-extra texlive-generic-recommended texinfo
dia texlive texlive-latex-extra texlive-extra-utils
texlive-generic-recommended texi2html python-pygraphviz
python-kiwi python-pygoocanvas libgoocanvas-dev
python-pygccxml
```

2.5.4. Ejecución

En la estructura de directorios de NS3 se observa una carpeta llamada *scratch*, es en esta carpeta donde debe ir el archivo **.cc** principal, que contiene el método *main*

junto con la definición de la simulación a realizar.

También en el directorio principal de NS3 se encuentra el archivo ejecutable *waf*. Waf es una herramienta, desarrollada en Python, que asiste en la configuración y compilación de programas o bibliotecas, similar a lo que hace *GNU Make*. Los detalles de los distintos comandos y los argumentos que usa cada uno se pueden encontrar en el tutorial oficial de NS3 [20], sin embargo a continuación se mencionaran brevemente los comandos más relevantes:

2.5.4.1. configure

Waf usa este comando para identificar qué paquetes y versiones de los mismos están instalados en el sistema, para identificar de esta manera el compilador a usar y los módulos que pueden o no ser habilitados. Este comando se debe ejecutar al menos una vez antes de usar NS3 por primera vez, luego se puede usar cuando se desee reconfigurar algunas opciones de NS3, por ejemplo una opción común es cambiar entre el modo de compilación *debug* (desarrollo) y *optimized* (producción).

2.5.4.2. build

Comando usado para compilar los módulos de NS3, este comando se debe ejecutar cada vez que se haga algún cambio al código de NS3 o de la simulación a realizar.

Finalmente para ejecutar una simulación luego que se ha compilado el código de ésta, se debe ejecutar el comando:

```
./waf --run="scratch/<archivo-principal>"
```

Donde *<archivo-principal>* es el nombre del archivo que contiene el método main dentro de la carpeta *scratch*, sin incluir la extensión *.cc*.

Capítulo 3

Simulador *ARAP*

El simulador desarrollado, llamado *ARAP* (*Ant Routing Anonymity Protocol*) se puede definir según cuatro componentes principales:

1. Manejo de parámetros
2. Aplicación ArapNode
3. Hormigas artificiales
4. Manejo de estadísticas

Los cuales están implementados mediante una serie de clases en conjunto con las que ofrece NS3, dichas clases se describen en la sección 3.7. El software está desarrollado bajo la licencia GNU GPLv2. En este capítulo se describe el simulador desarrollado desde el punto de vista de sus componentes, se indican sus características y se explica su funcionamiento.

3.1. Estructura de directorios en NS3

Los archivos y directorios que definen al simulador *ARAP* dentro de la estructura de NS3 se pueden ver en la figura 3.1. El simulador está diseñado como un nuevo módulo de NS3 llamado *anonymity* en el directorio *src*. Dentro de la carpeta *anonymity* se encuentran varias carpetas más según la estructura de los módulos de NS3, y es en la carpeta llamada *model* que se encuentra el código del simulador.

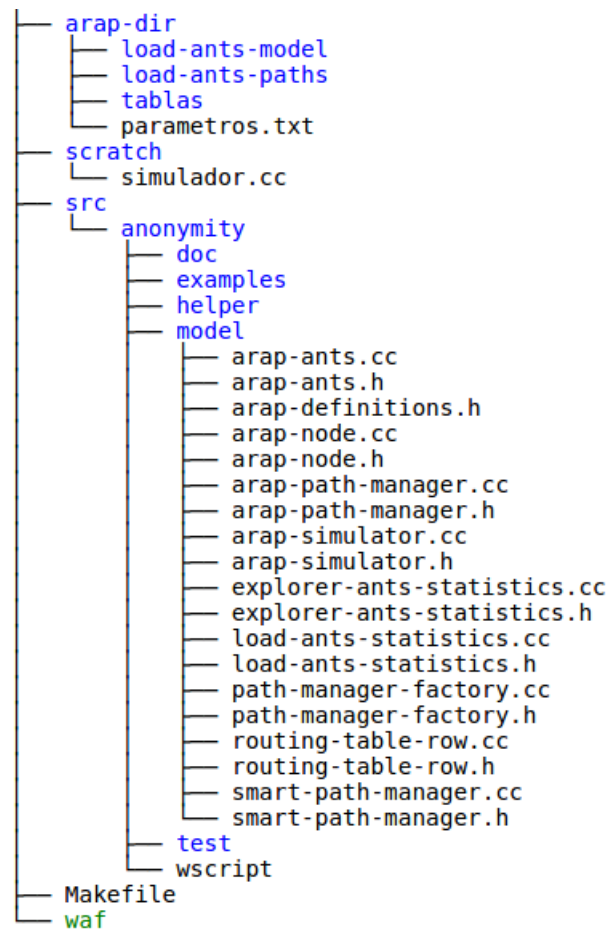


Figura 3.1: Estructura de directorios del simulador ARAP

Así mismo dentro la carpeta *scratch* (en el directorio raíz de NS3) se encuentra un archivo llamado *simulador.cc*, éste contiene el método *main* en el cual se ejecuta el simulador. Volviendo al directorio raíz, dentro de éste se encuentra otro directorio

llamado *arap-dir*, en éste se encuentra el archivo de configuración (*parametros.txt*) que se usa para configurar cada simulación a realizar, así mismo se encuentran una serie de carpetas (en principio vacías) en las cuales se crean distintos archivos en formato **.csv**¹ con los resultados de las simulaciones.

3.2. Topología

En la sección 2.4.1 se mencionó que el protocolo está diseñado para redes P2P. El simulador está definido para usar una topología tipo estrella como la mostrada en la figura 3.2, donde los nodos de los extremos son los que pertenecen a la red anónima, y el nodo central (también llamado HUB) se usa para simular el Internet, y para enrutar los distintos paquetes hacia los distintos nodos de los extremos.

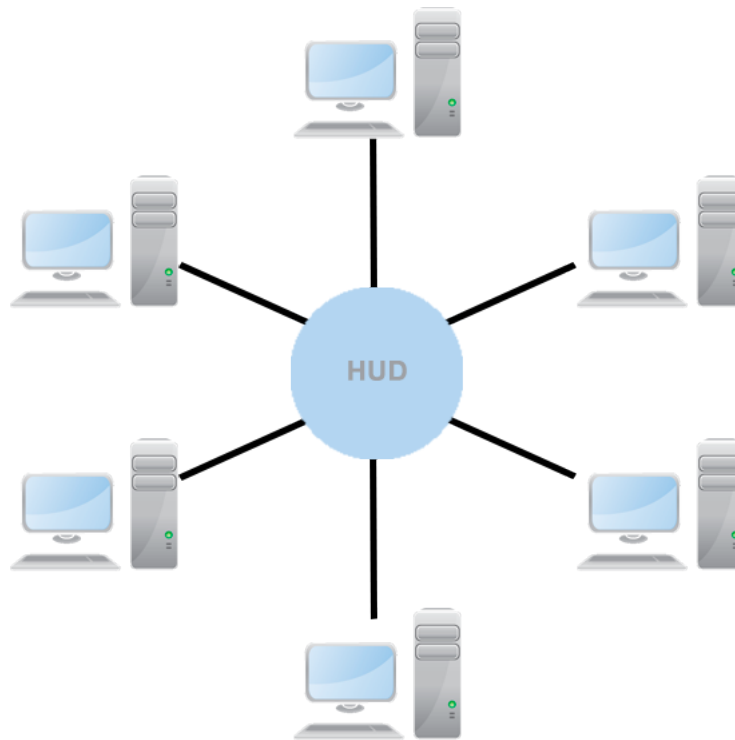


Figura 3.2: Topología estrella usada por el simulador

¹Comma Separated Values. Formato de archivos de texto estructurados para ser fácilmente leídos por filas, por ejemplo por programas de hojas de cálculo.

3.3. Parámetros

3.3.1. Archivo de configuración

El simulador está diseñado como *caja negra*, de manera que no es necesario que el usuario interactúe con el código fuente o haga modificaciones a éste a menos que desee agregar funcionalidades extra o definir nuevos comportamientos. De tal manera que para configurar los parámetros de las distintas simulaciones se usa un archivo de texto donde se define línea por línea los valores de dichos parámetros.

El archivo permite el uso de comentarios por línea, los cuales vienen dados por el uso del caracter `#` al comienzo de una línea, así mismo es válido dejar líneas en blanco entre parámetros. El simulador actualmente tiene definidos veintitrés (23) parámetros² para los cuales existen dos formatos en el archivo según corresponda:

1. *<nombre de parámetro> <valor>*
2. *<nombre de parámetro> <rango mínimo> <rango máximo> <valor>*

El primer caso es generalmente para parámetros globales, por ejemplo el tamaño en bytes de las hormigas o la cantidad de nodos, mientras que el segundo caso es para aquellos parámetros donde cada nodo requiere un valor independiente, por ejemplo el valor de *data-rate*.

Para aquellos parámetros que son variables aleatorias, *<valor>* equivale al nombre de alguna distribución de probabilidad seguido de los parámetros la misma. Cuando se lee un parámetro que hace uso de alguna distribución de probabilidad, es necesario identificar cuál es específicamente, ya que cada distribución requiere sus propios parámetros, por ejemplo una distribución uniforme requiere el intervalo en que está definida (mínimo y máximo), mientras que una distribución normal requiere conocer la media, la varianza y el valor límite para acotar los valores obtenidos.

²En el apéndice A se encuentra la descripción completa y detallada de cada uno de los parámetros

NS3 cuenta con una clase abstracta llamada ***RandomStreamVariable***, de la cual se especializan otras clases para definir una gran cantidad de distribuciones de probabilidad. A pesar de que esté definida esa gran cantidad, para ser leídas desde el archivo de configuración sólo se han definido cinco (5), sin embargo es posible reconocer más distribuciones desde el archivo con pocas líneas de código. Las distribuciones reconocidas actualmente son:

- Constante.
- Uniforme.
- Triangular.
- Exponencial.
- Normal.

3.3.2. Configuración del simulador

Cuando se ejecuta el software, la primera tarea que realiza el simulador es leer el archivo de configuración. Esta tarea es realizada en una clase llamada ***ArapSimulator***, la cual está diseñada para manejar la lectura y asignación de los distintos parámetros, así como de configurar la red a simular y controlar y permitir el inicio y finalización de las simulaciones. Es una clase «*static*», es decir que no es necesario (ni posible en este caso, ya que el constructor se definió como un método *privado*) crear instancias de ella para poder usar sus métodos. Esta clase es usada por los demás componentes principales y posee métodos que funcionan como una interfaz que se usa para realizar las simulaciones desde el método principal.

3.4. ArapNode

Como se mencionó en la sección 2.5.2, el nodo es la abstracción básica de un computador, el cual requiere entre otras cosas de aplicaciones para que tenga alguna funcionalidad. NS3 provee una clase abstracta llamada ***Application***, la cual provee la interfaz necesaria para crear e integrar nuevas aplicaciones. En el simulador *ARAP* la aplicación usada para simular el protocolo está implementada en la clase ***ArapNode***, la cual hereda de la clase ***Application*** antes mencionada.

ArapNode tiene como tareas principales:

- Mantener la lista de *sockets* y conexiones hacia cada nodo de la red anónima.
- Controlar el envío y la recepción de hormigas artificiales.
- Mantener y actualizar la tabla de enrutamiento de las hormigas.

3.4.1. Envío de hormigas exploradoras

En la sección 2.4.5 se explica la forma en que el protocolo define el envío de exploradoras. En el simulador *ARAP* esta tarea está implementada mediante dos algoritmos cortos.

Datos: N Conjunto de nodos de la red anónima.

para todo $i \in N$ hacer

si (*obtenerIP*(*i*) == *ObtenerIP*(this)) **entonces**

```
continuar ;           // Para evitar que se envíe a si mismo
```

fin

EnviarExploradorasADestino(obtenerIP(i));

fin

Algoritmo 1: Selección de destinos

Entrada: Dirección IP *destino*

Datos: N Conjunto de nodos de la red anónima

```

para todo  $i \in N$  hacer
    si (obtenerIP( $i$ ) == (obtenerIP(this) || obtenerIP( $i$ ) ==
        (obtenerIP(destino)) entonces
        continuar ; // Para evitar que use al destino o al origen como
            nodo intermedio
    fin
    hormiga = crearHormigaExploradora();
    ProgramarEnvioDeHormigaExploradora(hormiga);
fin

```

Algoritmo 2: Envío de exploradoras a un destino específico

El algoritmo 1 se usa para realizar la selección de los distintos destinos de las hormigas exploradoras, mientras que el algoritmo 2 corresponde al método *EnviarExploradorasADestino* en el cual dado un destino, se envían exploradoras usando cada uno de los demás nodos como nodo intermedio.

3.4.2. Envío de hormigas de carga

Las hormigas de carga son las que transportan los mensajes que envía un usuario, por ejemplo una petición *HTTP* o un mensaje de correo. En el simulador *ARAP* esta interacción por parte del usuario está programada de manera que se envíen periódicamente hormigas durante la simulación, para lograr esto el simulador requiere de tres (3) parámetros correspondientes a distribuciones de probabilidad que se indican en el archivo de configuración:

- Tiempo de envío
- Cantidad de hormigas

- Destino

En cada uno de los tiempos de envío, se obtiene aleatoriamente el valor de un nodo destino (previniendo que no coincida con el mismo nodo actual), y se procede a enviar una cantidad de hormigas según lo indique el valor obtenido de la distribución correspondiente, en la sección 2.4.5 se explica que cuando el mensaje es más grande que el tamaño de la hormiga, éste debe ser fragmentado en varias partes, y dichos fragmentos se convierten en la carga de las hormigas, el simulador no implementa actualmente esta operación, sino que ésta se simula con la cantidad de hormigas a enviar en cada tiempo de envío.

Para cada hormiga de carga a enviar se debe crear un camino conformado de distintos nodos que actúan como nodos intermedios y el destino final, la longitud de dicho camino está indicada por el parámetro *cantidad de saltos*. La forma de generar los caminos se define en el método **CreatePath** de la clase **ArapPathManager**, el cual es un método abstracto. Una definición de este método (y otros que corresponden al manejo de estadísticas) se realizan en la clase especializada **SmartPathManager**, en esta definición se usa el *método de Montecarlo* [21] para obtener cada elemento del camino en conjunto con los valores de las tablas de probabilidad, tal como se muestra en el algoritmo 3.

Para garantizar la protección de los datos éstos deben estar encriptados, sin embargo en NS3 el reloj de simulación avanza de igual manera así se haya o no aplicado alguna técnica de encriptación, razón por la que los datos se envían sin aplicar estas técnicas. Para hacer más precisas las simulaciones se usa un parámetro llamado *tiempo de cómputo*, el cual es un valor en milisegundos que simula el tiempo usado en encriptar y/o desencriptar los datos en cada nodo, es por esto que cuando se realizan envíos de hormigas (exploradoras o de carga) éstos no se realizan de inmediato, sino que se programa el envío para que se haga luego que transcurre el tiempo de cómputo en la simulación.

Entrada: Dirección IP *destino*

Salida: *camino* lista que contiene de forma ordenada las IP de los nodos que conforman el camino

Datos: *fila* Fila de la tabla de probabilidades para el nodo destino

valores = CrearDistribucionUniforme(0,1);

mientras *camino.longitud* < *cantidadDeSaltos()*-1 **hacer**

 valor = valores.obtenerValorReal();

 acumulado=0;

para todo $i \in \textit{fila}$ **hacer** ; // Recorrer la fila e ir obteniendo los valores de cada probabilidad

si (*obtenerIP(i)* == (*obtenerIP(destino)*)) **entonces**
 | continuar

fin

si (*valor* >= *acumulado* *∧* *valor* < *acumulado* + *i.probabilidad*)

entonces

si *obtenerIP(i)* no está en *camino* **entonces**

 Agregar *obtenerIP(i)* a *camino*;

 break ; // Dejar de recorrer la fila

sinó

acumulado = *acumulado* + *i.probabilidad*;

fin

fin

fin

fin

Agregar *destino* a *camino*;

devolver *camino*

Algoritmo 3: Creación de caminos con método de Montecarlo

3.4.3. Tabla de enrutamiento

Una de las funciones principales del protocolo simulado es ofrecer enrutamiento a nivel de capa aplicación, para lo cual es necesario que cada nodo mantenga y actualice su tabla de enrutamiento³, esta tabla indica desde y hacia qué nodos se envían las hormigas, y es la que permite que una vez que las mismas lleguen a sus destinos correspondientes, puedan realizar el camino de regreso al nodo origen.

En el simulador *ARAP* estas tablas se implementan como un mapa de objetos *RoutingTableRow*, una clase que modela cada fila de la tabla. Esta clase posee tres atributos:

- IP origen
- IP destino
- ID de la hormiga

El campo ID de la hormiga, corresponde a un valor que identifica inequívocamente cada hormiga creada, cuando un nodo recibe una hormiga se actualiza la tabla, sea agregando o eliminando filas según sea el caso.

Si se asume que la tabla 3.1 pertenece a un nodo **A** se pueden interpretar sus valores, por ejemplo la primera fila de la siguiente manera: **A** recibió una hormiga con ID 54 que provenía de un nodo **B**, y **A** actuando como nodo intermedio reenvió dicha hormiga a otro nodo **C**.

Origen	Destino	ID de hormiga
B	C	54
E	C	23
A	D	42

Tabla 3.1: Tabla de enrutamiento de nodo **A**

³No confundir con el enrutamiento de capa de red

3.4.4. Recepción de hormigas

Cuando una hormiga llega a un nodo se debe identificar si ésta llegó al destino o si se trata de un nodo intermedio, también si se trata de una petición (*request*) o una respuesta (*response*). La identificación de cada uno de estos casos se hace con ayuda de la tabla de enrutamiento y de un elemento en la estructura de las hormigas que indica si el nodo en que se encuentra es final o intermedio.

Según la definición del protocolo, cuando una hormiga llega a un nodo, éste se encarga de descryptar y remover una capa de la estructura cebolla que la conforma, debido a la forma que está creada, el nodo sólo podrá leer la información correspondiente a la capa recién descryptada, de esta manera no puede conocer qué otros nodos forman el camino e incluso si el nodo desde el cual llegó la hormiga es el nodo origen, o si al que la enviará será el nodo final. Entre los valores que puede leer el nodo se encuentran: El ID de hormiga y el identificador de tipo de nodo.

3.4.4.1. Peticiones

Si la tabla de enrutamiento no posee una fila para el ID de hormiga leído, significa que ésta es una petición (el camino corresponde a la *ida* hacia el destino).

Cuando se trata de un nodo intermedio otro de los datos que puede leer el nodo es la IP del siguiente salto, la cual se usa para hacer *forward* de la hormiga, y al hacerlo se agrega la fila correspondiente a la tabla de enrutamiento. En el caso de tratarse del nodo final éste puede leer la *carga* y luego crear la respuesta correspondiente y proceder a enviarla para que ésta vuelva al nodo donde se creó la petición; para este caso no se realiza ninguna acción en la tabla.

3.4.4.2. Respuestas

Si la tabla de enrutamiento posee una fila para el ID de hormiga leído, significa que ya ha pasado antes por el nodo y por ende esta vez corresponde a una respuesta

(el camino corresponde al *regreso* hacia el nido).

En este caso el nodo no verifica el campo tipo de nodo, sino que estudia los campos de la fila correspondiente a la hormiga recibida, si la IP del nodo actual es la misma que la del campo *origen*, significa que se encuentra en el nodo donde se creó la hormiga y ya ha terminado su recorrido, en este punto se lee la *carga* que posee y se pasa al manejo de estadísticas, el cual se explica en la sección 3.6. En caso que no se cumpla la condición antes mencionada, se trata de un nodo intermedio, en este caso se procede a reenviar la hormiga al nodo indicado en el campo *origen*. En ambos casos de respuesta, una vez se termina de procesar la hormiga, se elimina la fila correspondiente de la tabla.

Cada vez que una hormiga realiza un salto se elimina una capa de la estructura tipo cebolla, la cual reduce el tamaño de éstas, es por esta razón que para mantener el tamaño acorde se debe realizar un proceso de *relleno*, el cual consiste en agregar ceros al final de la estructura hasta que ésta tenga el tamaño adecuado. Esto se hace fácilmente en el simulador ya que no se están aplicando técnicas de encriptado (sino que se está simulando el tiempo de estas operaciones), en un caso de uso real sería necesario considerar otras técnicas para mantener la consistencia del tamaño de las hormigas entre saltos.

3.5. Estructura de las hormigas

Como se ha mencionado antes, el protocolo usa dos tipos de hormigas: exploradoras y de carga. Las hormigas artificiales son la parte fundamental que provee la *inteligencia* del protocolo, permitiendo en el caso de las exploradoras adaptarse y converger a soluciones sin interacción externa de algún otro agente.

En términos del protocolo las hormigas permiten encapsular la información y proveer la estructura cebolla requerida. La estructura de las hormigas en el simulador *ARAP* está definida en la clase ***ArapAnts***, ésta posee métodos para leer cada uno de los campos y para crear ambos tipos de hormigas según los parámetros que correspondan.

A nivel de programación las hormigas se implementan como un arreglo en el cual se escriben de forma ordenada los campos definidos en su estructura, tal como se muestra en la figura 3.3, la cual corresponde a una hormiga que requiere de tres (3) saltos para llegar al destino. Los primeros cuatro (4) campos mostrados (en negrita) son estructuralmente iguales a los siguientes cuatro, éstos corresponden a la *cabecera intermedia* los cuales son los valores a los que puede acceder cada nodo cuando es usado como nodo intermedio en el camino de una hormiga. Los últimos campos en la estructura de la hormiga (mostrados en itálica en la figura), corresponden a la *cabecera final* y similarmente al caso anterior, éstos son los que pueden leerse en el nodo cuando actúa como destino.

ID de hormiga		Nodo MEDIO		IP salto 2		Tamaño de capa 2		...
...	ID de hormiga	Nodo MEDIO		IP salto 3		Tamaño de capa 3		...
...	<i>ID de hormiga</i>	<i>Nodo FINAL</i>		<i>DATA</i>				

Figura 3.3: Estructura de hormiga con tres (3) capas

La principal diferencia entre ambas cabeceras aparte de ciertos campos, es que en el nodo final se puede acceder a *DATA*, el cual es un conjunto de campos con información que va dirigida al nodo destino, razón por la cual es sólo este nodo el que tiene acceso a ella. La estructura de *DATA* se puede ver en la figura 3.4.

Tipo de hormiga	Tiempo de envío	IP de destino	<Tamaño de mensaje>	<Mensaje>
-----------------	-----------------	---------------	---------------------	-----------

Figura 3.4: Estructura genérica de *DATA* en las hormigas del simulador ARAP

Los campos <Tamaño de mensaje> y <Mensaje> corresponden sólo a las hormigas de carga. Los campos *Tiempo de envío* e *IP destino* son usados para el manejo de estadísticas en el simulador, aparte de esto no tendrían necesidad de utilizarse en el caso de uso real.

Cuando se crea una hormiga, se empieza desde la capa más externa que corresponde a los primeros bytes del arreglo, según la cantidad de saltos se crean iterativamente los campos intermedios y al final se agrega la estructura *DATA*, la cual para el caso de las hormigas de carga puede variar según el tamaño del mensaje. Se puede apreciar que entre los campos se encuentra uno llamado *Tamaño de capa siguiente*, para calcular el valor de este campo en cada capa se usa la ecuación 3.1.

$$tamañoCapa_{i+1} = T_{cm}(n - i - 1) + T_{cf} \quad (3.1)$$

Donde:

- $i \in [0, n - 1]$
- n : cantidad de capas (cantidad de saltos)
- T_{cm} : Tamaño (en bytes) de cabecera intermedia
- T_{cf} : Tamaño (en bytes) de cabecera final

La estructura de las hormigas definida en el simulador *ARAP* está basada en la que usa TOR en sus celdas⁴, sin embargo pueden haber otras formas de definir las según la implementación que se realice del protocolo.

3.6. Manejo de estadísticas

La principal razón para desarrollar el simulador es validar la hipótesis propuesta en el protocolo que plantea que al usar los algoritmos ACO para la selección de caminos se logran reducir los tiempos de respuesta en las comunicaciones anónimas, para validar esto el simulador genera una serie de estadísticas con los valores del *RTT*⁵ de las

⁴En TOR se llama celda a la estructura usada para empaquetar los mensajes y proveer el enrutamiento cebolla

⁵Round Trip Time. Tiempo que tome obtener una respuesta desde el momento que se envía la petición correspondiente

diferentes hormigas que viajan en la red anónima. Cada tipo de hormiga, al cumplir una función específica es tratada de forma diferente según sea el caso.

3.6.1. Hormigas exploradoras

Las hormigas exploradoras pertenecen al modelo del algoritmo ACO que se ha adaptado para ser usado en el simulador *ARAP*, éstas proveen los valores necesarios para actualizar las tablas de probabilidad. Las fórmulas usadas para tal fin están basadas en las descritas para el algoritmo AntNet en [6], allí se proporciona una justificación detallada de las mismas y los valores usados en los distintos parámetros.

La clase encargada de realizar estas operaciones en el simulador *ARAP* se llama *SmartPathManager*, la cual es una especialización de *ArapPathManager*, en esta clase se implementa la tabla de probabilidad de cada nodo y se definen los métodos para actualizar dicha tabla, a su vez posee los modelos locales de estadísticas que se mantienen para cada destino.

Cuando un nodo recibe una exploradora de vuelta al nido, se obtiene el RTT de la misma, con este valor se procede a actualizar el modelo estadístico local del destino para el cual se exploró el camino según las fórmulas 3.2 para la media y 3.3 para la varianza.

$$\mu = \mu + \varsigma(rtt + \mu) \quad (3.2)$$

$$\sigma^2 = \sigma^2 + \varsigma(rtt - \mu)^2 - \sigma^2 \quad (3.3)$$

Donde:

- μ : Media estadística de los tiempos de respuesta.
- σ^2 : Varianza de los tiempos de respuesta.

- ς : Factor que mide la cantidad de muestras más recientes que afectan el promedio.
- rtt : Tiempo de viaje de la hormiga.

Como se mencionó en la sección 2.4.3 cada fila de la tabla es una distribución de probabilidad, es por esto que cada vez que se actualiza un valor, se debe normalizar el resto de los valores para que se sigan manteniendo las propiedades de la distribución. El proceso de aumentar la probabilidad se denomina *aumentar feromona* y consiste en calcular el valor del campo T_{fm} según la fórmula 3.4, mientras que el proceso de normalización de los demás campos se conoce como *decrementar feromona* y consiste en calcular el valor de los demás campos en la distribución según la fórmula 3.5.

$$T_{fm} = T_{fm} + r(1 - T_{fm}) \quad (3.4)$$

$$T_{fj} = T_{fj} + r(1 - T_{fj}) \quad \forall j \in N, \quad j \neq f, \quad j \neq m \quad (3.5)$$

Donde:

- T_{fm} : Campo con la probabilidad del camino recorrido por la exploradora.
- T_{fj} : Campo con la probabilidad de un camino no recorrido por la exploradora para el mismo destino.
- r : Valor de refuerzo calculado.
- N : Nodos de la red anónima.

El valor de r se conoce como el *refuerzo* o *recompensa*, y es el que regula qué tanto incrementa o aumenta una probabilidad según el valor de RTT dado, en la implementación realizada r se calcula con la fórmula 3.6.

$$r = C_1 \frac{W_{best}}{RTT} + C_2 \frac{I_{sup} - I_{inf}}{I_{sup} + RTT - 2I_{inf}} \quad (3.6)$$

Donde:

- C_1 y C_2 son constantes definidas para la implementación actual.
- I_{sup} e I_{inf} son estimados de los límites de confianza para μ .
- W_{best} es el mejor tiempo obtenido en la ventana de observación actual.

Al tratarse de un protocolo para proveer Anonimato es ideal que un mismo nodo o camino no sea seleccionado frecuentemente, lo cual puede ocurrir si un camino resulta siempre mejor que otros, ya que su probabilidad tendería a 1. Para prevenir este comportamiento se usa un valor de probabilidad máxima que puede tener una celda de la tabla de probabilidad, luego de actualizar la tabla se verifica que el nuevo valor de probabilidad no supere el máximo, y de hacerlo se realiza una normalización en la cual se distribuye el excedente de manera uniforme entre los demás campos, tal como lo describe el algoritmo 4.

Entrada: p La probabilidad nueva calculada

Datos: N nodos de la red anónima

si $p > MAX_PROB$ **entonces** ; // MAX_PROB es una constante
definida en la implementación

```

|    $resto = p - MAX\_PROB$ ;
|    $p = MAX\_PROB$ ;
|   para todo  $p_i$  en la fila, excepto el  $p$  actual hacer
|       |    $p_i = p_i + \frac{resto}{N-2-1}$  ;
|   fin
fin

```

Algoritmo 4: Normalizar probabilidades si se supera el valor máximo

3.6.2. Estadísticas de hormigas de carga

Cada vez que se recibe una hormiga de carga de vuelta al nodo donde se creó es necesario actualizar el modelo estadístico de éstas en dicho nodo. El modelo permite obtener la media y la varianza de los tiempos de respuesta obtenidos por las hormigas de carga enviadas a los demás nodos de la red anónima, estos valores se obtienen a partir de las fórmulas 3.7 y 3.8.

$$E(x) = \mu = \frac{\sum_{i=1}^n x_i}{n} \quad (3.7)$$

$$\sigma^2 = E(x^2) - [E(x)]^2 \quad (3.8)$$

Donde:

- x : Tiempos de viaje de las hormigas.
- $E(x)$: Esperanza o media estadística de x .
- σ^2 : Varianza de x
- n : Cantidad de muestras (tiempos) recibidas.

Para calcular los valores requeridos el modelo almacena tres (3) datos, los cuales actualiza al momento de recibir la hormiga según se indica a continuación:

1. Se incrementa el contador n que lleva la cantidad de muestras (los tiempos) recibidas en uno ($n++$).
2. Se incrementa la suma de los tiempos:
 $sumaMuestras = sumaMuestras + rtt$, donde $sumaMuestras$ equivale a $\sum_{i=1}^n x_i$, con la cual se calcula la media.

3. Se incrementa la suma de los cuadrados de los tiempos:

$sumaCuadrados = sumaCuadrados + rtt^2$, la cual se usa para calcular el valor de $E(x^2)$ y luego σ^2 .

Al finalizar la simulación los valores de los modelos estadísticos de cada nodo de la red anónima se exportan a un archivo, donde también se indica cuántas hormigas de carga se enviaron en total y cuántas envió cada nodo, así como la media y la varianza global de todos los tiempos registrados.

3.7. Arquitectura

A lo largo del capítulo se han ido mencionando las distintas clases que conforman el simulador, en esta sección se mencionan características sobre la arquitectura y ciertas decisiones de diseño pensadas con el fin de expandir la funcionalidad del software creado.

Para empezar, en la figura 3.5 se muestra el diagrama relacional de las clases que conforman el simulador *ARAP*, la mayoría de éstas ya han sido mencionadas en las secciones anteriores de este capítulo y explicada la tarea que cumplen.

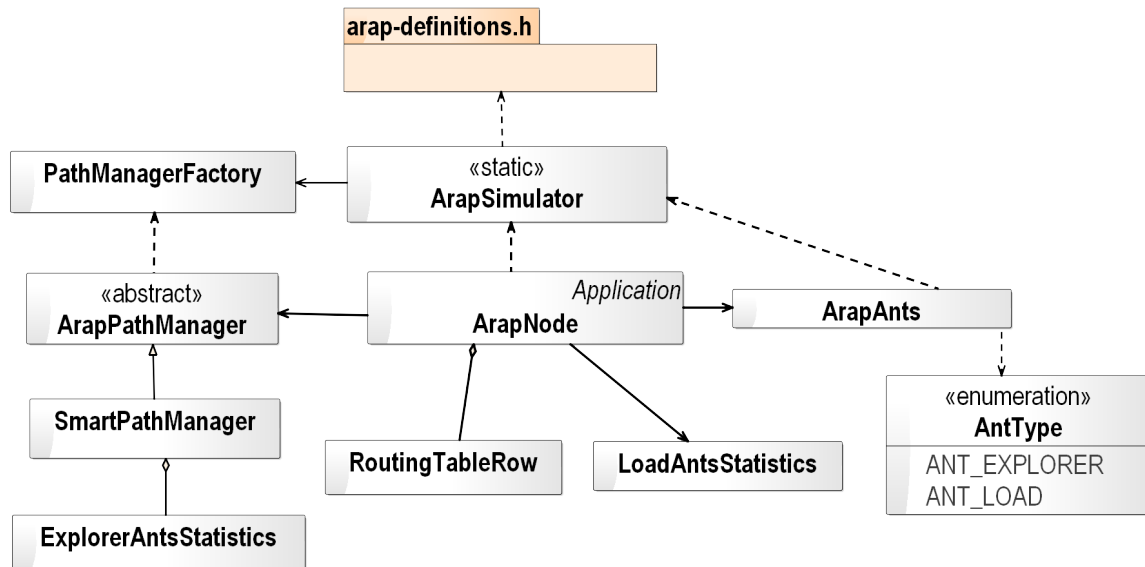


Figura 3.5: Diagrama relacional de clases del simulador ARAP

A continuación se muestran los diagramas detallados de cada una de las clases:



Figura 3.6: Diagrama de clase ArapSimulator



Figura 3.7: Diagrama de clase ArapNode

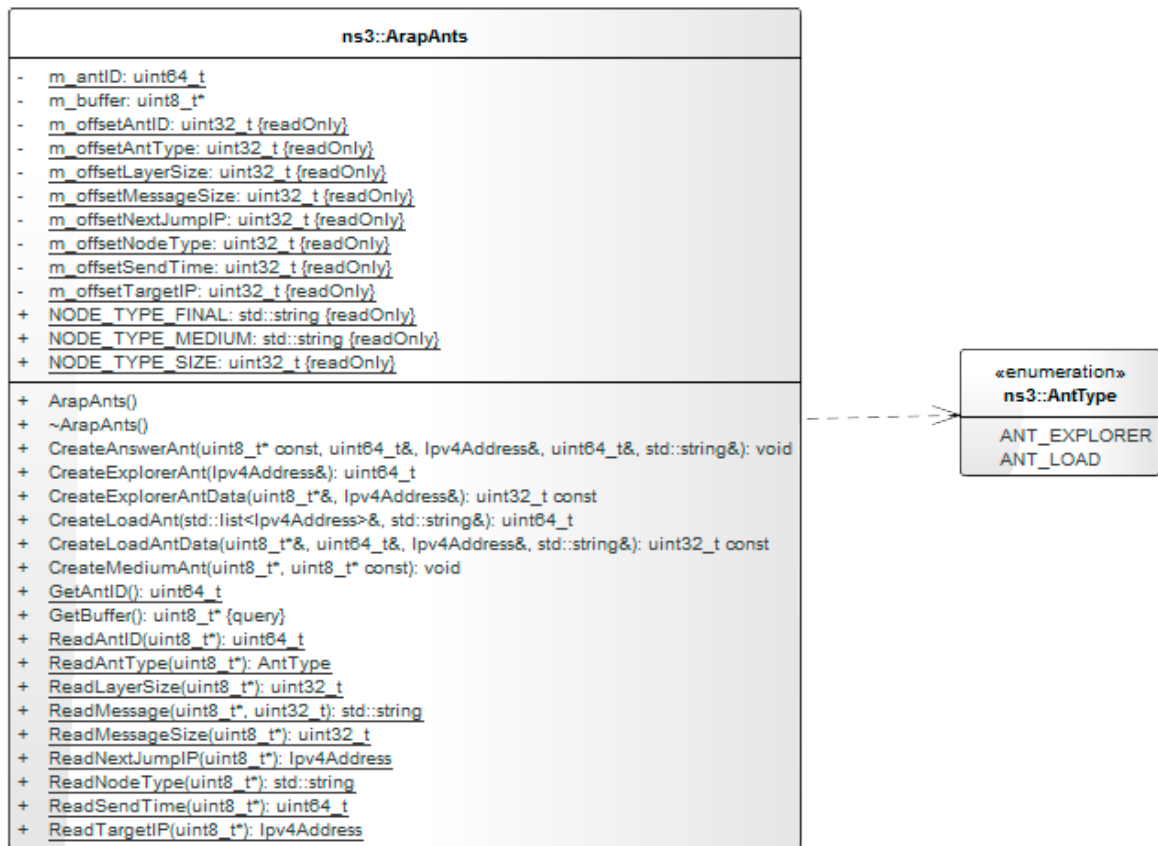


Figura 3.8: Diagrama de clase ArapAnts

ns3::RoutingTableRow
<ul style="list-style-type: none"> - m_antID: uint84_t - m_sourceIP: Ipv4Address - m_targetIP: Ipv4Address
<ul style="list-style-type: none"> + GetAntID(): uint84_t {query} + GetSourceIP(): Ipv4Address& {query} + GetTargetIP(): Ipv4Address& {query} + operator==(RoutingTableRow&): bool {query} + RoutingTableRow() + RoutingTableRow(Ipv4Address&, Ipv4Address&, uint84_t) + ~RoutingTableRow() + SetPacketID(uint84_t): void + SetSourceIP(Ipv4Address&): void + SetTargetIP(Ipv4Address&): void

ns3::LoadAntsStatistics
<ul style="list-style-type: none"> - m_numSamples: uint84_t - m_samplesCount: std::map<Ipv4Address,uint84_t> - m_sumDelay: double - m_sumDelaySquare: double
<ul style="list-style-type: none"> + GetMean(): double {query} + GetNumSamples(): uint84_t {query} + GetSampleCountsMap(): std::map<Ipv4Address,uint84_t>& {query} + GetVariance(): double {query} + InitializeTable(): void + LoadAntsStatistics() + ~LoadAntsStatistics() + UpdateModel(double, Ipv4Address&): void

Figura 3.9: Diagrama de las clases RoutingTableRow y LoadAntsStatistics

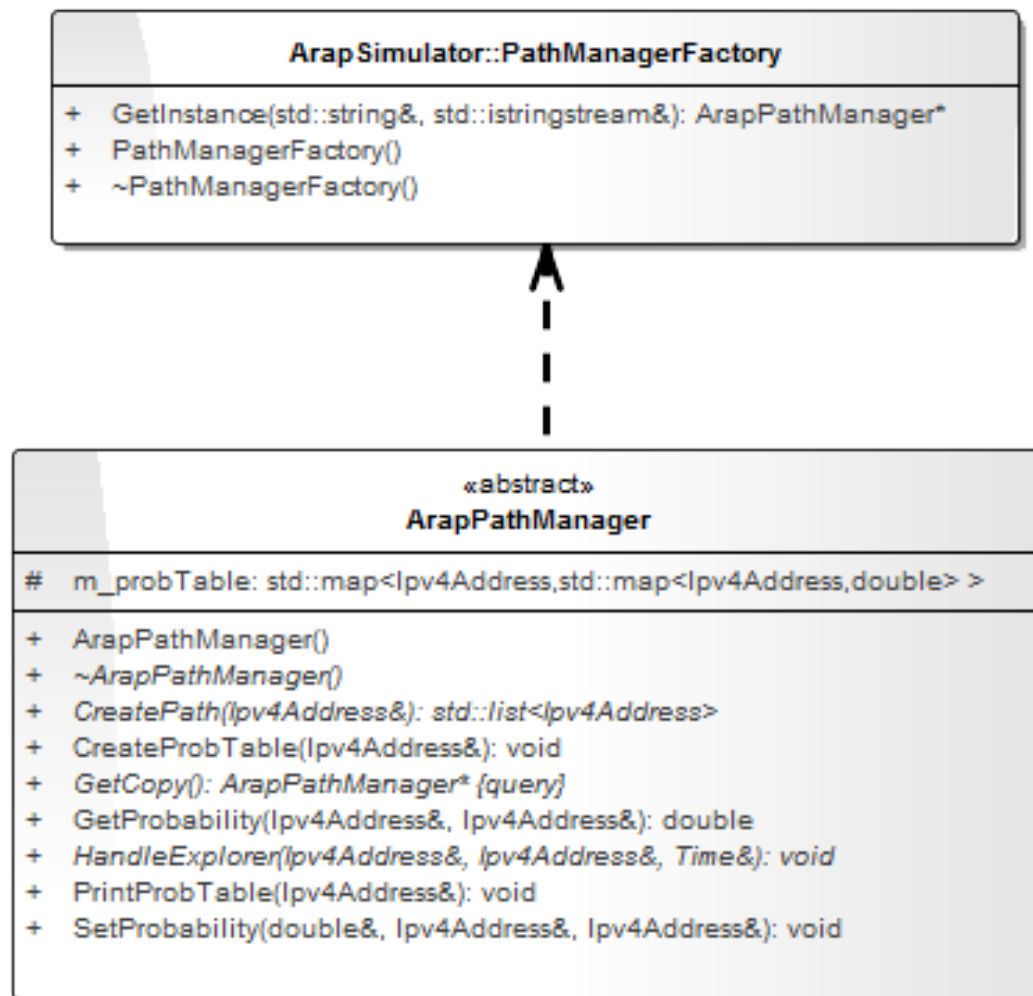


Figura 3.10: Diagrama de las clases ArapPathManager y PathManagerFactory

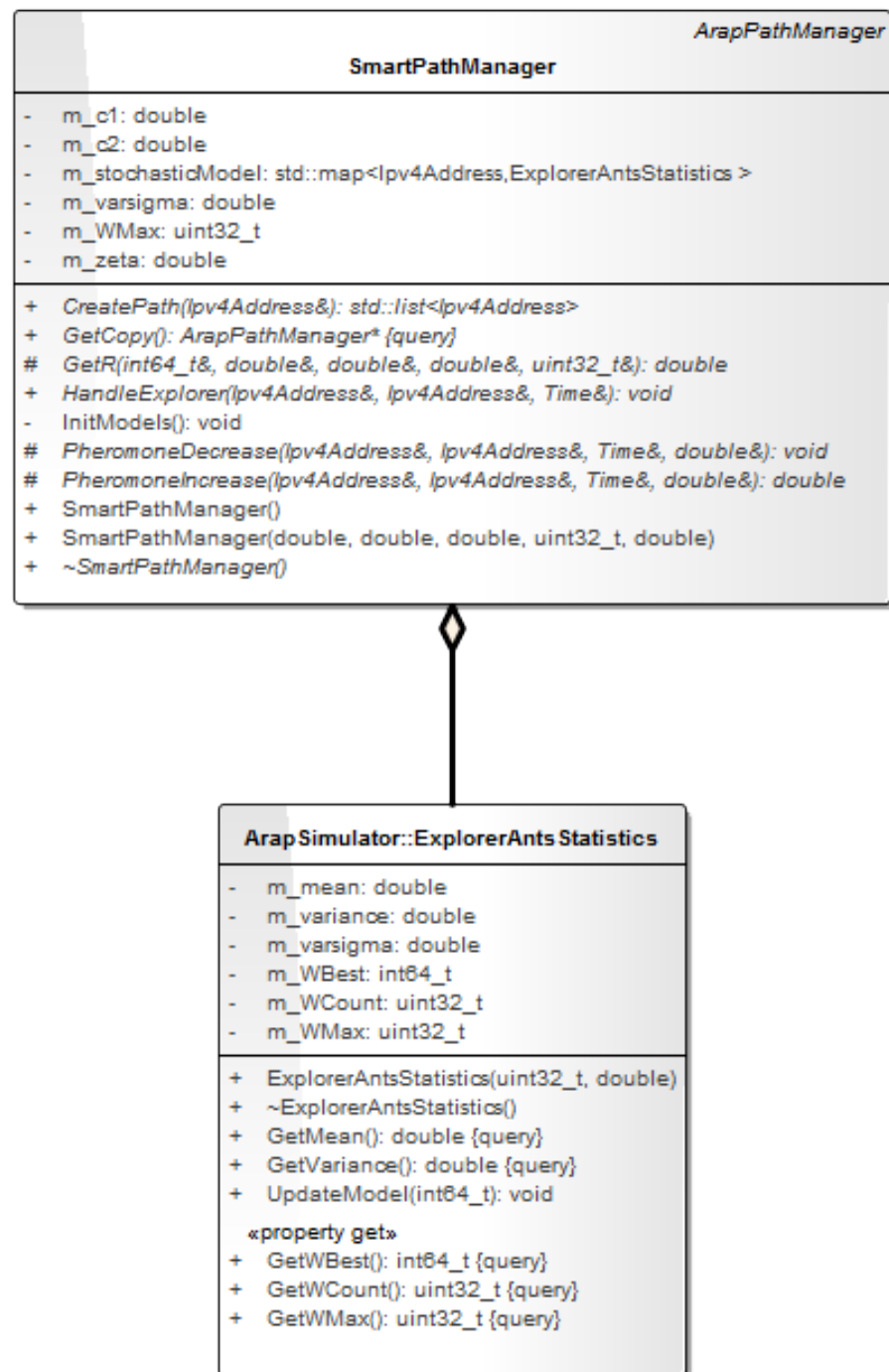


Figura 3.11: Diagrama de las clases SmartPathManager y ExplorerAntsStatistics

Como se ha mencionado, la clase *ArapPathManager* es una clase totalmente abstracta⁶, su función en el diseño usado es principalmente proveer un conjunto de métodos abstractos cuya definición sea responsabilidad de las clases que hereden de ella, la finalidad de esto es poder integrar fácilmente distintas implementaciones de los métodos usados para actualizar los valores de la tabla de probabilidad.

Los métodos abstractos a los que se hace referencia son:

- *HandleExplorer*
- *CreatePath*
- *GetCopy*

HandleExplorer se encarga de recibir la información necesaria para la actualización de la tabla, lo cual se hace según la implementación definida, por otra parte *CreatePath* se encarga de la generación de los caminos cada vez que se crea una hormiga de carga.

El método *GetCopy* se usa para poder crear múltiples copias del objeto de la clase especializada con los mismos parámetros, ya que todos los nodos de la red anónima deben usar los mismos métodos de actualización de las tablas de probabilidad.

Otra clase que cabe destacar es *PathManagerFactory*, ésta se usa como fábrica de objetos de clases que heredan de *ArapPathManager*. Se usa esta clase para proveer a los usuarios que deseen expandir el simulador con sus propias especializaciones, de una interfaz que permita leer los parámetros de dicha especialización desde el archivo de configuración, en el *apéndice C* se explica cómo realizar este proceso.

⁶No se pueden instanciar objetos de dicha clase

3.8. Ejecución

NS3 incluye un archivo *Makefile* que usa para simplificar la ejecución de ciertos comandos *waf*, este archivo fue modificado para permitir configurar el simulador *ARAP* según opciones específicas del mismo y también para permitir automatizar las diferentes ejecuciones del simulador al usar un mismo conjunto de parámetros, dichas modificaciones constan de tres comandos que simplifican el uso del simulador *ARAP* desde la línea de comandos.

3.8.1. `configure` y `debug`

En la sección 2.5.4 se menciona el comando `configure` de *waf* que permite realizar las configuraciones iniciales de NS3, así como alternar entre los modos de desarrollo y de producción. Los comandos `debug` y `configure` del *Makefile* se usan para permitir cambiar entre ambos modos respectivamente. En el modo *debug* NS3 habilita los *log* que muestran información durante la simulación, en el caso del simulador *ARAP* éstos archivos se guardan en la carpeta *arap-dir*, en un archivo de nombre: *arap-logs-`<run>`.log*. Donde `<run>` es el número de ejecución de la simulación a la que corresponde el archivo. El modo *optimized* deshabilita los *log* y permite que las simulaciones se ejecuten hasta cuatro veces más rápido que en el modo *debug*.

A continuación se muestra el extracto del archivo *Makefile* con ambos comandos mencionados y su expansión usando *waf*.

`configure:`

```
./waf --build-profile=optimized --enable-examples --enable-tests  
--out=build/optimized configure
```

`debug:`

```
./waf --build-profile=debug --enable-examples --enable-tests  
--out=build/debug configure
```

3.8.2. run

Este comando se usa para ejecutar las simulaciones, éste realiza de manera automática las distintas ejecuciones de una misma simulación sin intervención del usuario. A continuación se muestra su definición en el archivo Makefile.

```
run:
    @n=0; \
    while [ $$n -lt $(RUNS) ]; \
    do \
        n='expr $$n + 1'; \
        run=$$RANDOM; \
        date; \
        ./waf --cwd="arap-dir" --run="scratch/simulador  
    $$run $(PARAMETROS)" 2>arap-dir/arap-logs-$$run.log; \
    done; \
```

Hace uso de dos (2) parámetros: *RUNS* y *PARAMETROS*, los cuales por defecto tienen los valores *1* y *parametros.txt* correspondientemente. *RUNS* usa un valor numérico para indicar cuantas repeticiones de la simulación se deben realizar, cambiando siempre el *run number* para poder obtener diferentes números de los generadores de números aleatorios de NS3. Por otra parte *PARAMETROS* se usa para indicar el nombre del archivo de configuración del cual se leen los parámetros de la simulación a realizar, usando como directorio raíz para tal archivo la carpeta *arap-dir*. Por ejemplo si se desea ejecutar una simulación con un mismo conjunto de parámetros cinco (5) veces, y que éstos sean leídos de un archivo llamado *prueba.txt*, se debe usar el siguiente comando:

```
make PARAMETROS="prueba.txt" RUNS=5 run
```

Aparte de los dos anteriores, también se puede usar el parámetro *RANDOM*, que es propio de la consola, para cambiar el valor de la semilla con que se generan los *run number* para cada simulación.

3.9. Resultados

Luego de cada simulación realizada se crean una serie de archivos de formato **.csv** que contienen información sobre las variables de estudio. En principio se trata de tres (3) tipos de archivos:

- Modelo estadístico de las hormigas de carga para cada nodo
- Tablas de probabilidad de cada nodo en tiempos determinados de la simulación
- Caminos generados para las hormigas de carga

Las salidas se guardan en carpetas dentro del directorio *arap-dir*, los modelos estadísticos de las hormigas de carga en la carpeta *load-ants-model*, y las tablas en la carpeta de nombre *tablas*.

Los modelos de las hormigas de carga se guardan al momento de finalizar la simulación en archivos cuyo nombre sigue el siguiente formato:

<exploradoras habilitadas>_<semilla>_<ejecución>.csv

Donde:

- *<exploradoras habilitadas>* Indica si la simulación se realizó con el uso de hormigas exploradoras o no
- *<semilla>* Valor de la semilla usada para la generación de números aleatorios
- *<ejecución>* Valor de ejecución usado en conjunto con la semilla para la generación de números aleatorios

Por otra parte, los valores de las tablas de probabilidad se exportan a archivo de manera periódica durante la simulación según el intervalo de tiempo definido en

el archivo de configuración, ésto se hace con el fin de apreciar la manera cómo van cambiando los valores de éstas a lo largo de la simulación. Los archivos se nombran según el siguiente formato:

$$\langle IP\ local \rangle_ \langle tiempo \rangle_ \langle semilla \rangle_ \langle ejecución \rangle. \mathbf{csv}$$

Donde:

- $\langle IP\ local \rangle$ es la dirección IP del nodo al que pertenece la tabla
- $\langle tiempo \rangle$ El valor de tiempo de simulación en que se creó el archivo
- $\langle semilla \rangle$ y $\langle ejecución \rangle$ mantienen el mismo significado que en el caso de los modelos de hormigas de carga.

Capítulo 4

Pruebas y resultados

4.1. Pruebas

Al finalizar el desarrollo del simulador *ARAP* fueron realizadas pruebas piloto con el fin de validar su comportamiento y comprobar que se realizan las tareas que se plantean en el protocolo, las cuales incluyen: Creación aleatoria de caminos, enrutamiento correcto de las hormigas, actualización adecuada de las tablas de probabilidad y de enrutamiento.

Las pruebas realizadas consistieron de un análisis comparativo de los tiempos de respuesta de las hormigas de carga cuando se hace uso de las hormigas exploradoras y los algoritmos ACO, y cuando no se usan éstos, para lo cual se ejecutaron treinta (30) simulaciones para cada caso (con y sin hormigas exploradoras) con un conjunto de parámetros definidos, cambiando el valor del número de ejecución para obtener números diferentes por parte del generador de números aleatorios de NS3 en cada prueba, al tratarse de pruebas comparativas, se usaron los mismos valores de los parámetros en ambos casos de estudio.

Las pruebas fueron ejecutadas en un equipo con las siguientes especificaciones:

- Procesador Intel I7 4770 3.4 Ghz

- 8 GB de RAM DDR3
- Disco duro de 500 GB Sata 3.0 (7200 RPM)
- S.O. Ubuntu 14.04.3 de 64 bits
- Tarjeta gráfica Nvidia Geforce GT430

4.1.1. Parámetros de las simulaciones

Como se mencionó en la sección 3.3 los parámetros se definen de dos maneras: Por valor único o por rango. En la tabla 4.1 se muestran los valores usados en las pruebas realizadas para los parámetros que consisten de un valor único, los cuales son globales en la simulación.

Parámetro	Valor
Nodos	40
Salto de hormigas de carga	3
Tamaño de hormigas	512 (bytes)
Intervalo de envío de exploradoras	444.6 (Segundos)
Tiempo de simulación	86400 (segundos)
Distribución de retardo en los enlaces	Uniforme $\in [50, 100]$ (milisegundos)
Cambios del retardo en los enlaces	1 (segundo)
Imprimir tablas de probabilidad	8640 (segundos)

Tabla 4.1: Valores de parámetros globales usados en las pruebas

La tabla 4.2 muestra los parámetros que se definen según rangos, cuyos valores son locales a cada nodo. La tabla muestra el rango de nodos al que se le asigna un valor dado de un parámetro, por ejemplo la primera línea indica que el valor de *data-rate* para los nodos 0 a 7 (inclusive) es de 10,485,760 bps (10 Mbps).

Nombre	Rango mínimo	Rango máximo	Valor
Data-rate	0	7	10485760 (bps)
Data-rate	8	15	5242880 (bps)
Data-rate	16	23	2097152 (bps)
Data-rate	24	31	8388608 (bps)
Data-rate	32	39	1048576 (bps)
Iniciar envío de hormigas	0	39	1.0 (segundos)
Distribución de tiempo de computo	0	39	Uniforme $\in [30, 50]$ (milisegundos)
Incremento de tiempo de computo	0	7	0
Incremento de tiempo de computo	8	15	0.1
Incremento de tiempo de computo	16	23	0.2
Incremento de tiempo de computo	24	31	0.3
Incremento de tiempo de computo	32	39	0.4
Tiempo de envío de hormigas de carga	0	39	Normal $\mu = 105, \sigma^2 = 25 \in [90, 120]$ (Segundos)
Cantidad de hormigas de carga por envío	0	39	Uniforme $\in [3, 6]$
Destinos de hormigas de carga	0	39	Normal $\mu = 19,5, \sigma^2 = 42,25 \in [0, 39]$

Tabla 4.2: Valores de parámetros locales usados en las pruebas

Los valores mostrados son sólo para los parámetros que afectan los resultados de la simulación, por ejemplo el parámetro *número de puerto* se omite ya que no es relevante para los resultados obtenidos. Cada valor usado fue seleccionado ya sea por el estudio de antecedentes donde se usaban valores similares o por observación y estudio de las tecnologías actuales. A continuación se detallan las justificaciones de los valores usados en cada parámetro:

- La cantidad de *nodos* es la misma que poseía la red TOR en sus inicios, mientras que el *número de saltos* y el *tamaño de las hormigas* son valores usados en la implementación actual de TOR[14].
- En [6] se menciona que en AntNet cada hormiga exploradora es enviada con una diferencia de 0.3 ms, calculando la cantidad de hormigas creadas en cada exploración y usando el valor anterior como referencia se obtiene el tiempo usado para el *intervalo de envío de exploradoras*.
- El valor de *tiempo de simulación* que se usó es el tiempo suficiente para obtener una cantidad de muestras (rtt de hormigas de carga) que permite aproximar los

resultados obtenidos a una distribución normal mediante la ley de los grandes números.

- Para simular la dinámica cambiante de las redes, se optó por cambiar cada segundo el valor del *retardo en los enlaces*, el rango de la distribución que define dichos valores (*distribución de retardo en los enlaces*) viene dada por la realización de distintas pruebas de *ping* a varios sitios web.
- Los archivos con los valores de las tablas de probabilidad se crean según intervalos de una hora en tiempo de simulación para así poder apreciar cómo éstas van cambiando a lo largo de la misma, este valor es relevante sólo en el caso de estudio en que se habilitan las exploradoras, ya que en el otro caso los valores de las tablas no cambian luego de ser inicializadas y por ende los archivos creados serán siempre iguales.
- Los valores de *data-rate* corresponden a las velocidades de Internet que ofrecen los proveedores en el país actualmente, y los rangos usados equivalen a distribuciones uniformes de dichas velocidades en los distintos nodos de la red anónima a simular.
- Todos los nodos inician el envío de hormigas al mismo tiempo, para generar tráfico inicial.
- El *tiempo de cómputo* y los *incrementos* respecto al mismo corresponden a los valores que se muestran en los bancos de prueba de la biblioteca Crypto++ [22] que implementa diferentes algoritmos de encriptación, como el RSA.
- Para el tiempo de envío de hormigas de carga y la cantidad de hormigas de carga por envío se observó a distintos usuarios y la frecuencia con que realizaban peticiones HTTP en un tiempo dado y los tamaños de dichas peticiones y respuestas obtenidas. Se eligió como ejemplo tráfico HTTP ya que es el tráfico más común en Internet y es uno de los que utilizará el protocolo.
- Para los *destinos de hormigas de carga* se definió una distribución que permite que se hagan peticiones a todos los nodos de la red anónima, sin embargo no con

la misma probabilidad de seleccionar cada nodo como destino, sino que algunos destinos son más probables que otros, de esta manera se acerca más la simulación al comportamiento real de las comunicaciones, donde un nodo se comunica con más frecuencia con una serie de nodos que con otros.

4.2. Resultados

Es importante resaltar que los resultados obtenidos aunque no se basan en un diseño de experimento formal, tienen la intención de mostrar los resultados que el simulador es capaz de arrojar.

Al finalizar las pruebas se obtuvieron los distintos archivos con los resultados de las mismas. Para cada ejecución se genera un archivo que contiene: La cantidad de hormigas que envió cada nodo a los demás, la media y varianza de los tiempos de respuesta de cada nodo; también al final de cada archivo se encuentran datos globales de la simulación que indican la cantidad total de hormigas de carga creadas, la media y la varianza de todos los tiempos promedios de respuesta de cada nodo.

Los archivos obtenidos son bastante extensos y por dicha razón no se muestran en su totalidad. A continuación se muestran una serie de tablas con los datos más relevantes obtenidos de las pruebas ejecutadas, si se desea estudiar a detalle dichos archivos, éstos se encuentran en el CD que acompaña este documento y se pueden descargar desde el repositorio público¹ junto con el código del proyecto.

La tabla 4.3 muestra la cantidad de hormigas de carga creadas en las pruebas realizadas para ambos casos de estudio (con y sin exploradoras), se puede ver que en la mayoría de las simulaciones el número es el mismo en ambos casos. Por otro lado las tablas 4.4 y 4.5 muestran de la misma manera los valores de los promedios globales y la varianza de los tiempos de respuesta en cada simulación realizada.

¹<https://github.com/david-sca/simulador-arap-v1.0>

N° de simulación	Con Exploradoras	Sin exploradoras
1	147998	147998
2	147987	147987
3	148311	148311
4	147659	147659
5	148422	148422
6	148087	148087
7	148018	148018
8	148163	148163
9	148033	148033
10	148048	148048
11	147804	147804
12	147965	147967
13	147827	147827
14	147774	147774
15	148209	148209
16	148304	148304
17	147988	147988
18	148167	148167
19	148033	148033
20	147938	147938
21	147557	147557
22	147707	147707
23	148170	148170
24	147947	147947
25	147713	147713
26	148180	148180
27	148283	148283
28	148063	148063
29	147977	147977
30	148058	148058

Tabla 4.3: Cantidad de hormigas de carga creadas en las pruebas realizadas

N° de simulación	Con Exploradoras	Sin exploradoras
1	1.6732	1.18646
2	1.67453	1.18691
3	1.66041	1.18653
4	1.63941	1.1866
5	1.67647	1.18707
6	1.67469	1.18602
7	1.67149	1.18664
8	1.69334	1.18649
9	1.68864	1.18708
10	1.68637	1.18675
11	1.65275	1.18657
12	1.64873	1.18629
13	1.65321	1.18669
14	1.67014	1.1861
15	1.66462	1.1865
16	1.6779	1.18624
17	1.64645	1.18668
18	1.67717	1.18662
19	1.6686	1.18629
20	1.67336	1.18639
21	1.67504	1.18693
22	1.67912	1.18647
23	1.68483	1.18644
24	1.65677	1.18699
25	1.66253	1.18652
26	1.69143	1.18678
27	1.66354	1.18695
28	1.67543	1.18686
29	1.64996	1.18664
30	1.67504	1.18607

Tabla 4.4: Promedio de las medias (en segundos) de los tiempos de respuesta de las hormigas de carga en las pruebas realizadas

N° de simulación	Con Exploradoras	Sin exploradoras
1	0,0311218315	0,0002404625
2	0,0344003704	0,0001419613
3	0,0248197419	0,0001252945
4	0,026442625	0,0002205
5	0,0369078485	0,0002728448
6	0,0744439698	0,0001229312
7	0,0162991513	0,0001374482
8	0,0543345613	0,0002933042
9	0,1593470605	0,0002050313
10	0,0432150601	0,0002026085
11	0,0346344881	0,000345845
12	0,0295828488	0,000163624
13	0,033359445	0,0002234498
14	0,060378125	0,000167445
15	0,0198821741	0,0001486088
16	0,0894983432	0,000271678
17	0,0720936392	0,0002302658
18	0,1624101025	0,0001875985
19	0,0445750082	0,0001491265
20	0,0970509625	0,0001942421
21	0,0063777218	0,0000824328
22	0,0825317192	0,0002232385
23	0,1356111121	0,000163624
24	0,0808944865	0,0001571765
25	0,0680030321	0,0001465472
26	0,0616005	0,0002238728
27	0,0473304145	0,0001630818
28	0,0348981781	0,000184512
29	0,0418241042	0,0001776613
30	0,0639639145	0,0001992008

Tabla 4.5: Varianza de las medias (en segundos cuadrados) de los tiempos de respuesta de las hormigas de carga en las pruebas realizadas

Además de las estadísticas obtenidas, para cada ejecución también se crea un archivo que contiene los caminos recorridos por cada hormiga de carga, un extracto de este archivo se puede ver en la tabla 4.6, en la cual se pueden ver los caminos desde el origen al destino, y el tiempo de simulación en que se generan éstos.

Tiempo	Origen	Salto 1	Salto 2	Destino
97,534	10.1.18.2	10.1.36.2	10.1.40.2	10.1.20.2
97,534	10.1.18.2	10.1.36.2	10.1.38.2	10.1.20.2
97,534	10.1.18.2	10.1.40.2	10.1.37.2	10.1.20.2
97,534	10.1.18.2	10.1.33.2	10.1.37.2	10.1.20.2
97,534	10.1.18.2	10.1.33.2	10.1.38.2	10.1.20.2
98,4031	10.1.31.2	10.1.34.2	10.1.39.2	10.1.13.2
98,4031	10.1.31.2	10.1.35.2	10.1.36.2	10.1.13.2
98,4031	10.1.31.2	10.1.39.2	10.1.33.2	10.1.13.2
100,324	10.1.36.2	10.1.34.2	10.1.39.2	10.1.14.2
100,324	10.1.36.2	10.1.34.2	10.1.38.2	10.1.14.2
100,324	10.1.36.2	10.1.39.2	10.1.34.2	10.1.14.2
100,752	10.1.35.2	10.1.39.2	10.1.36.2	10.1.20.2
100,752	10.1.35.2	10.1.37.2	10.1.34.2	10.1.20.2
100,752	10.1.35.2	10.1.34.2	10.1.39.2	10.1.20.2
100,752	10.1.35.2	10.1.39.2	10.1.34.2	10.1.20.2
100,752	10.1.35.2	10.1.37.2	10.1.36.2	10.1.20.2

Tabla 4.6: Extracto de archivo que contiene caminos de hormigas de carga

Con los resultados obtenidos se puede concluir que:

- Los caminos se generan aleatoriamente y son diferentes para cada hormiga, como se puede apreciar en el archivo que contiene los caminos generados. Existen casos donde un mismo nodo es seleccionado más seguido que otros pero aún así los caminos son diferentes.

- Las tablas de probabilidad se actualizan a lo largo de la simulación y las filas siempre mantienen las propiedades de una distribución de probabilidad, respetando los valores máximos definidos en la implementación. Con ésto también se verifica que las hormigas exploradoras recorren la red correctamente.
- Los valores de las estadísticas obtenidas permiten verificar que las hormigas de carga recorren de manera correcta los caminos creados para ellas, ya que los modelos estadísticos solo se actualizan cuando la hormiga de carga vuelve al nodo origen.

Capítulo 5

Conclusiones y recomendaciones

5.1. Conclusiones

El desarrollo del simulador *ARAP* marca un punto de partida para continuar la investigación del protocolo simulado, así mismo se espera que sirva como aporte para solventar el problema de la carencia de herramientas especializadas para sistemas anónimos con las características del protocolo planteado y como recurso para la creación de otras similares en el área.

El simulador fue implementado como un módulo de NS3 y funciona mediante los principios de las simulaciones por eventos discretos, está diseñado de manera que con las funcionalidades que posee actualmente pueda ser fácilmente usado por usuarios sin conocimientos especializados, requiriendo sólo definir los parámetros de cada simulación en un archivo de texto sin necesidad de manipular el código, a su vez permite agregar nuevas características que se integran a la arquitectura definida para el mismo. Posee características que ayudan a acercar los modelos simulados a la realidad al incluir factores de interés en la abstracción de éstos, como lo es la simulación de los *tiempos de cómputo* en cada nodo. Los diferentes parámetros que definen cada simulación permiten variar las condiciones de la red desde la cantidad de nodos que la conforman hasta la cantidad de hormigas de carga que envía cada nodo en un tiempo dado.

Luego de implementar las funcionalidades requeridas por el simulador se realizaron pruebas para comprobar que el sistema funciona de manera correcta, es decir que el comportamiento de las hormigas sea el esperado, que se realice el enrutamiento de las mismas, se actualicen las tablas y se generen las estadísticas correspondientes al final de cada simulación. Las pruebas fueron diseñadas para llevar al máximo las capacidades del simulador respecto a las limitantes de cómputo con que se contaba para las mismas; al completarse dichas pruebas y obtener los resultados se pudo observar que todos los módulos cumplían su función y se generaban los resultados acorde a lo esperado.

Cabe decir que los resultados de las pruebas obtenidas no son en realidad significativos para efectos de la hipótesis que plantea el protocolo y no permiten ofrecer conclusiones respecto a la misma, ya que para esto es necesario realizar pruebas que estén enfocadas a dicha tarea y que estén regidas por la teoría de diseño de experimentos.

5.2. Recomendaciones

Al tratarse de la primera versión del simulador, se puede decir que aún existen muchas oportunidades de trabajo por realizar con él, y que hay nuevas características y mejoras por agregar. Entre las cuales cabe destacar las siguientes:

- Creación de interfaz gráfica para el manejo de los parámetros de simulación y otras operaciones.
- Creación de módulo para graficar resultados de las simulaciones obtenidas.
- Implementación de la fragmentación de mensajes en nodo origen, y reensamblamiento en nodo destino, esta función permite acercar aún más los modelos a la realidad y de esta manera obtener resultados más precisos de las simulaciones.

- Realizar pruebas formales sobre el protocolo, basadas en la teoría de diseño de experimentos para verificar o rechazar la hipótesis propuesta y/o formular nuevas según los resultados de las mismas.

Como recomendación al diseñar los componentes gráficos, se sugiere que se provean los mismos de métodos de accesibilidad para facilitar su uso por parte de personas con alguna discapacidad, para dicha tarea se recomienda el uso de las bibliotecas GTK o alguna otra que posea soporte a dicha característica.

Apéndice

Apéndice A

Descripción de parámetros

A continuación se especifican los formatos y dominios de los diferentes parámetros que posee el simulador *ARAP* actualmente, los cuales se usan para configurar las distintas simulaciones a realizar.

A.1. Cantidad de nodos

- Constante que lo define: *PARAM_NODES*
- Descripción: Cantidad de nodos. Este valor debe ser definido antes que cualquier otro parámetro en el archivo.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_NODES (nodos)
- valor: entero ≥ 4

A.2. Tamaño de hormigas

- Constante que lo define: *PARAM_ANT_SIZE*
- Descripción: Tamaño en bytes de las hormigas enviadas a través de la red.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE *PARAM_ANT_SIZE* (tamano-hormiga)
- valor: entero ≥ 128

A.3. Habilitar exploradoras

- Constante que lo define: *PARAM_ENABLE_EXPLORER_ANTS*
- Descripción: Parámetro (booleano) que indica si se deben o no enviar hormigas exploradoras a través de la red.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE *PARAM_ENABLE_EXPLORER_ANTS*
(habilitar-exploradoras)
- valor: booleano (0 = falso, 1 = verdadero)

A.4. Habilitar trazas

- Constante que lo define: *PARAM_ENABLE_TRACING*
- Descripción: Parámetro (booleano) que indica si se deben o no crear archivos **.pcap** y **.tr** de cada uno de los nodos que forman la red simulada. Estos archivos contienen información sobre cada paquete que viaja en la red simulada y son leídos por el programa *wireshark*.

- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_ENABLE_TRACING (usar-trazas)
- valor: booleano (0 = falso, 1 = verdadero)

A.5. Intervalo de envío de exploradoras

- Constante que lo define: *PARAM_EXPLORER_INTERVAL*
- Descripción: Intervalo de tiempo (segundos) en que se envían las hormigas exploradoras.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_EXPLORER_INTERVAL
(intervalo-exploradoras)
- valor: real > 0

A.6. Saltos de las hormigas

- Constante que lo define: *PARAM_HOPS*
- Descripción: Número de saltos que realizan las hormigas de carga.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_HOPS (saltos)
- valor: entero $\in [2, \text{cantidad de nodos}-1]$

A.7. Distribución de retardo en los enlaces

- Constante que lo define: *PARAM_LINK_DELAY_DIST*
- Descripción: Distribución de probabilidad que provee los valores de delay (milisegundos) en los enlaces de la red.
- Formato en el archivo: *<nombre> <nombre-distribución> <parametros-distribución>*

Donde:

- nombre: VALOR DE *PARAM_LINK_DELAY_DIST* (delay-enlaces-dist)
- nombre-distribución: Nombre de la distribución de probabilidad a usar
- parametros-distribución: Parámetros requeridos por la distribución de probabilidad a usar

A.8. Intervalo de cambio de retardo en los enlaces

- Constante que lo define: *PARAM_LINK_DELAY_INTERVAL*
- Descripción: Intervalo de tiempo (segundos) con que se cambia el delay de los enlaces.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE *PARAM_LINK_DELAY_INTERVAL* (delay-enlaces-intervalo)
- valor: real > 0

A.9. Tamaño máximo de segmento

- Constante que lo define: *PARAM_MAXIMUM_SEGMENT_SIZE*

- Descripción: Valor del MSS (Maximum Segment Size) a ser usado por el protocolo TCP en los sockets cliente de los nodos pertenecientes a la red.

- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_MAXIMUM_SEGMENT_SIZE (tamano-segmento)
- valor: entero ≥ 0

A.10. Puerto

- Constante que lo define: *PARAM_PORT*
- Descripción: Número de puerto por el que se reciben las hormigas.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_PORT (puerto)
- valor: entero $\in [1, 65535]$

A.11. Intervalo de impresión de tablas de probabilidad

- Constante que lo define: *PARAM_PRINT_PROB_TABLES_INTERVAL*
- Descripción: Intervalo de tiempo (segundos) con que se exportan a archivo las tablas de probabilidad de los nodos.

- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_PRINT_PROB_TABLES_INTERVAL (imprimir-tablas-intervalo)
- valor: real > 0

A.12. Tamaño de cola

- Constante que lo define: *PARAM_QUEUE_SIZE*
- Descripción: Capacidad máxima de paquetes que pueden haber en cada una de las colas de los nodos pertenecientes a la red.
- Formato en el archivo: *<nombre> <valor>*
Donde:

- nombre: VALOR DE PARAM_QUEUE_SIZE (tamano-cola)
- valor: entero $\Rightarrow 0$

A.13. Número de ejecución

- Constante que lo define: *PARAM_RUN_NUMBER*
- Descripción: Número de ejecución a usar en conjunto con la semilla para la generación de números aleatorios en la simulación. Este valor se usa principalmente para cambiar los valores de un experimento (también es posible cambiar la semilla, pero para diferentes ejecuciones con una misma configuración se recomienda cambiar el número de ejecución).
- Formato en el archivo: *<nombre> <valor>*
Donde:

- nombre: VALOR DE PARAM_RUN_NUMBER (ejecucion)
- valor: entero $\Rightarrow 0$

A.14. Semilla

- Constante que lo define: *PARAM_SEED_NUMBER*
- Descripción: Valor de la semilla a usar para la generación de números aleatorios en la simulación.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_SEED_NUMBER (semilla)
- valor: entero > 0

A.15. Tiempo de simulación

- Constante que lo define: *PARAM_SIMULATOR_STOP_TIME*
- Descripción: Tiempo (segundos) de duración de la simulación.
- Formato en el archivo: *<nombre> <valor>*

Donde:

- nombre: VALOR DE PARAM_SIMULATOR_STOP_TIME
(tiempo-detener-simulador)
- valor: real > 0

A.16. Tiempo de inicio de aplicaciones

- Constante que lo define: *PARAM_APP_START_TIME*
- Descripción: Tiempo (segundos) en que se inicia el envío de hormigas exploradoras y de carga en los nodos especificados en el rango.

- Formato en el archivo: $\langle nombre \rangle \langle rangoMin \rangle \langle rangoMax \rangle \langle valor \rangle$

Donde:

- nombre: VALOR DE PARAM_APP_START_TIME (app-tiempo-iniciar)
- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- $rangoMin \leq rangoMax$
- valor: real ≥ 1

A.17. Factor de incremento de tiempo de cómputo

- Constante que lo define: *PARAM_COMPUTING_DELAY_INCREMENT*
- Descripción: Factor de incremento de tiempo de cómputo, respecto a la referencia fijada según sea la distribución de probabilidad seleccionada para cada nodo.
- Formato en el archivo: $\langle nombre \rangle \langle rangoMin \rangle \langle rangoMax \rangle \langle valor \rangle$

Donde:

- nombre: VALOR DE PARAM_COMPUTING_DELAY_INCREMENT (delay-computo-incremento)
- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- $rangoMin \leq rangoMax$
- valor: real $\in [0, 1]$

A.18. Data-rate

- Constante que lo define: *PARAM_DATA_RATE*

- Descripción: Data-rate (bits por segundo) a asignar a los nodos especificados en el rango.

- Formato en el archivo: $\langle nombre \rangle \langle rangoMin \rangle \langle rangoMax \rangle \langle valor \rangle$

Donde:

- nombre: VALOR DE PARAM_DATA_RATE (data-rate)
- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- $rangoMin \leq rangoMax$
- valor: entero > 0

A.19. Distribución de tiempo de cómputo

- Constante que lo define: *PARAM_COMPUTING_DELAY_DIST*
- Descripción: Distribución de probabilidad que provee los valores del retraso generado por el tiempo de cómputo (milisegundos) en los nodos especificados.

- Formato en el archivo: $\langle nombre \rangle \langle rangoMin \rangle \langle rangoMax \rangle \langle nombre-distribución \rangle \langle parametros-distribución \rangle$

Donde:

- nombre: VALOR DE PARAM_COMPUTING_DELAY_DIST (delay-computo-dist)
- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- $rangoMin \leq rangoMax$
- nombre-distribución: Nombre de la distribución de probabilidad a usar
- parametros-distribución: Parámetros requeridos por la distribución de probabilidad a usar

A.20. Cantidad de hormigas de carga por envío

- Constante que lo define: *PARAM_LOAD_ANT_QUANTITY_DIST*
- Descripción: Distribución de probabilidad de la cual se obtiene la cantidad de hormigas de carga a enviar en cada tiempo indicado por la distribución de *PARAM_LOAD_ANT_TIME_DIST*. Este valor sirve para simular en cuántas partes es dividido un mensaje creado por el usuario. Se recomienda que el rango de valores que usen las distribuciones sean valores pequeños, adaptados a la realidad de un mensaje p.e. una petición HTTP.
- Formato en el archivo: *<nombre> <rangoMin> <rangoMax> <nombre-distribución> <parametros-distribución>*

Donde:

- nombre: VALOR DE *PARAM_LOAD_ANT_QUANTITY_DIST* (hormigacarga-cantidad-dist)
- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- *rangoMin* \leq *rangoMax*
- nombre-distribución: Nombre de la distribución de probabilidad a usar
- parametros-distribución: Parámetros requeridos por la distribución de probabilidad a usar

A.21. Distribución para destinos de hormigas de carga

- Constante que lo define: *PARAM_LOAD_ANT_TARGET_DIST*
- Descripción: Distribución de probabilidad de la cual se obtiene el nodo destino al cual se envían las hormigas de carga en cada tiempo indicado por la distribución

de `PARAM_LOAD_ANT_TIME_DIST`. El rango de valores que usen las distribuciones DEBEN ser valores $\in [0, \text{cantidad de nodos}-1]$.

- Formato en el archivo: `<nombre> <rangoMin> <rangoMax>`
`<nombre-distribución> <parametros-distribución>`

Donde:

- nombre: VALOR DE `PARAM_LOAD_ANT_TARGET_DIST`
(hormigacarga-destino-dist)
- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- $\text{rangoMin} \leq \text{rangoMax}$
- nombre-distribución: Nombre de la distribución de probabilidad a usar
- parametros-distribución: Parámetros requeridos por la distribución de probabilidad a usar

A.22. Distribución para tiempos para envíos de hormigas de carga

- Constante que lo define: `PARAM_LOAD_ANT_TIME_DIST`
- Descripción: Distribución de probabilidad de la cual se obtienen los distintos tiempos de envío (segundos) de hormigas de carga en los nodos indicados según el rango dado.
- Formato en el archivo: `<nombre> <rangoMin> <rangoMax>`
`<nombre-distribución> <parametros-distribución>`

Donde:

- nombre: VALOR DE `PARAM_LOAD_ANT_TIME_DIST`
(hormigacarga-tiempo-dist)

- rangoMin: entero $\in [0, \text{cantidad de nodos}-1]$
- rangoMax: entero $\in [0, \text{cantidad de nodos}-1]$
- $\text{rangoMin} \leq \text{rangoMax}$
- nombre-distribución: Nombre de la distribución de probabilidad a usar
- parametros-distribución: Parámetros requeridos por la distribución de probabilidad a usar

A.23. PathManager

- Constante que lo define: *PARAM_PATH_MANAGER*
- Descripción: Parámetro que indica la especialización de *ArapPathManager* a usar para el manejo de las tablas de probabilidad y la creación de caminos. Los distintos valores de *<nombre>* dependen de los definidos en *PathManagerFactory::GetInstance()*¹.
- Formato en el archivo: *<nombre> <identificador de caso> [<parámetros de caso>]*

Donde:

- nombre: VALOR DE PARAM_PATH_MANAGER (path-manager)
- identificador de caso: cadena identificadora del caso a usar para crear la especialización
- parámetros de caso: Lista de parámetros definida según el caso en el método *PathManagerFactory::GetInstance()*

Para los parámetros que hacen uso de distribuciones de probabilidad, a continuación se listan aquellas que son válidas junto con sus parámetros y el orden de éstos en el archivo de configuración.

¹Véase también el apéndice **C**

- UniformRandomVariable $\langle rangoMin \rangle \langle rangoMax \rangle$
- ConstantRandomVariable $\langle valor \rangle$
- TriangularRandomVariable $\langle rangoMin \rangle \langle rangoMax \rangle \langle media \rangle$
- ExponentialRandomVariable $\langle media \rangle \langle límite \rangle$
- NormalRandomVariable $\langle media \rangle \langle varianza \rangle \langle límite \rangle$

Todos los parámetros de las distribuciones pueden ser valores enteros o decimales, y su dominio viene dado por la distribución específica, por ejemplo una distribución exponencial sólo está definida para valores positivos. El parámetro $\langle límite \rangle$ se usa para acotar los valores posibles en distribuciones que tienden al infinito.

Para más información sobre los valores, y sobre otras distribuciones de probabilidad que pudieran ser programadas dentro del simulador se puede verificar la documentación de NS3 sobre las clases que heredan de *RandomVariableStream* [23].

Apéndice B

Cómo agregar nuevos parámetros

El simulador *ARAP* posee una gran cantidad de parámetros, sin embargo es posible agregar nuevos siguiendo los pasos siguientes.

1. En el archivo **arap-definitions.h** se agrega una constante que indique el nombre que se usará para identificar el parámetro en el archivo de configuración, este nombre debe ser una cadena de caracteres sin espacios, si el nombre a usar consta de varias palabras se recomienda separarlas con guiones para mantener consistencia con el formato del resto de los parámetros. Por ejemplo:

```
#define PARAM_COMPUTING_DELAY_INCREMENT "delay-computo-incremento"
```

2. Dentro del enum *StringValue* agregar un nuevo valor referente al nuevo parámetro, luego en el método ***ArapSimulator::InitializeValuesMap*** asigna al mapa *s_mapStringValue* el par clave, valor formado por el valor numérico del parámetro en el enum, y la cadena que lo identifica definida en el paso anterior. Este paso se hace para permitir la conversión de la cadena a un valor numérico al momento de leer los parámetros y manejar los casos específicos de cada uno.
3. Si el parámetro requiere el uso de rangos, se debe definir en el mismo archivo *arap-definitions.h* una estructura de datos que sirva para almacenar cada rango y los valores asociados. por ejemplo:

```

struct RandomStreamFormat {
    Ptr<RandomVariableStream> stream;
    int min;
    int max;

    RandomStreamFormat():stream(0),min(0),max(0){};
    RandomStreamFormat(Ptr<RandomVariableStream> s, int mi, int ma):
        stream(s),min(mi),max(ma){};
};

```

4. Dentro de la clase ***ArapSimulator*** agregar un atributo que será usado para almacenar el valor del parámetro, si el parámetro es por rangos el atributo corresponde a una lista que almacena instancias del *Struct* definido en el paso anterior.
5. En la función ***ArapSimulator::HandleParameter*** agregar un nuevo *case* en la estructura *switch* definida ahí, la clave del nuevo caso corresponde al valor numérico definido en el enum *StringValue* y dentro de éste se define la forma de leer y validar el nuevo parámetro.
6. Una vez que se ha leído el parámetro, se asigna éste donde corresponda, ya sea en la función ***ArapSimulator::ConfigNetwork*** o ***ArapSimulator::ConfigNodes***.

Si lo que se desea es agregar una nueva distribución de probabilidad de las que están definidas en NS3, se debe ir también a la función ***ArapSimulator::ReadRandomStream*** y agregar el caso correspondiente dentro de la estructura *if-else* que ahí se encuentra. En la definición del caso se leen los parámetros de la nueva distribución con el mismo orden que en el archivo de configuración.

Apéndice C

Especializar ArapPathManager

Esto se hace con el fin de cambiar el comportamiento y/o definición de los algoritmos ACO usados, para integrar una nueva implementación se deben seguir los siguientes pasos:

1. Dentro de la carpeta *model* del módulo *anonymity* (donde se encuentra el código del simulador *ARAP*), crear una clase que herede de ***ArapPathManager***, e implementar en ésta los métodos virtuales:

- ***HandleExplorer***.
- ***CreatePath***.
- ***GetCopy***.

HandleExplorer se usa para manejar la recepción de una exploradora cuando llega de vuelta al nido, allí se actualiza la tabla de probabilidad y los modelos estocásticos en caso de tenerlos. ***CreatePath*** indica el algoritmo usado para la selección de los nodos que conforman los caminos de las hormigas de carga, y ***GetCopy*** es usado para crear múltiples instancias de la clase, requeridas por cada nodo, por lo cual la definición más sencilla de dicho método es retornar una copia del objeto *this*.

2. Luego de crear la clase, es necesario indicarle a NS3 que la detecte al momento de compilar. Para esto se debe ir al archivo *wscript* que se encuentra en la carpeta *anonymity* y agregar dentro de los array *headers.source* y *module.source* la entrada correspondiente a la ruta desde la carpeta *anonymity* hasta los nuevos archivos **.h** y **.cc** respectivamente.
3. Para finalizar es necesario reconfigurar el simulador para que detecte la nueva clase que fue agregada al módulo, lo cual se puede hacer facilmente con los comandos `make debug` o `make configure`.

C.1. PathManagerFactory

La clase *PathManagerFactory* se usa en el simulador *ARAP* como una herramienta para distinguir entre las varias especializaciones de *ArapPathManager* y que éstas puedan ser leídas desde el archivo de configuración junto con los parámetros definidos para las mismas.

Para usarlo se debe primero crear la especialización, y luego dentro del método *PathManagerFactory::GetInstance* se agrega un nuevo caso dentro de la estructura *if-else* ahí definida.

Una vez en el método, el valor de *caseName* corresponde a la cadena *<identificador de caso>* usada para identificar el caso en el archivo de configuración según se indica en la sección [A.23](#). En la definición del caso se indica el orden en que serán leídos los parámetros de la especialización desde el archivo (de tenerlos). Una vez se haya creado y configurado la nueva instancia sólo es necesario retornarla y de esta manera queda integrada la especialización al archivo de configuración.

Referencias bibliográficas

- [1] H. C. A. van Tilborg y S. Jajodia, *Encyclopedia of Cryptography and Security*, vol. 1. Springer, 2 ed., 2011.
- [2] A. Cecil, “A summary of network traffic monitoring and analysis techniques”, tech. rep., Washington University in St. Louis School of Engineering and Applied Science, Department of Computer Science and Engineering, diciembre 2006.
- [3] “Declaración universal de derechos humanos”, diciembre 1948. Art. 12.
- [4] R. L. Sumoza Matos, A. L. Sandoval Orozco, L. J. García Villalba, y T. Kim, “Collective intelligence for anonymous systems”, julio 2012.
- [5] G. A. Di Caro, “Ant colony optimization and its application to adaptive routing in telecommunication networks”. Université Libre de Bruxelles. Faculté des Sciences Appliquées, septiembre 2004.
- [6] M. Dorigo y T. Stützle, *Ant Colony Optimization*. Massachusetts Institute of Technology, 2004.
- [7] E. Granados, “Un metodo para el ajuste de la velocidad de transmisiones cortas de datos en redes tcp/ip: Un enfoque por simulación”, Proyecto de Grado, Universidad de Los Andes, septiembre 2014.
- [8] M. Puente Pomposo, “Mejora del comportamiento del protocolo udp sobre redes inalámbricas multi-salto a través de técnicas de network coding”, Proyecto Fin de Carrera, Universidad de Cantabria, octubre 2013.

- [9] G. A. Wainer, “Introducción a la simulación de sistemas de eventos discretos”, septiembre 2003.
- [10] H. Hoeger, “Apuntes de modelado y simulación 1”.
<http://webdelprofesor.ula.ve/ingenieria/hhoeger/simulacion/PARTE1.pdf>,
febrero 2005.
- [11] R. Coss Bú, *Simulación: Un enfoque práctico*. Grupo Limusa, 2003.
- [12] R. Sumoza, “Sistemas anónimos en escenarios globales”, Proyecto Fin de Máster en Investigación en Informática, Universidad Complutense de Madrid, septiembre 2008.
- [13] A. Pfitzmann, “A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management”, agosto 2010.
- [14] R. Dingledine, N. Mathewson, y P. Syverson, “Tor: The second-generation onion router”, mayo 2004.
- [15] G. Danezis y C. Diaz, “A survey of anonymous communication channels”, Tech. Rep. MSR-TR-2008-35, Microsoft Research, enero 2008.
- [16] J. Kurose y K. Ross, *Redes de computadoras 5 ed.* Pearson Addison-Wesley, 2010.
- [17] C. Adams y S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Technology series, Addison-Wesley, 2003.
- [18] “What is ns3”, enero 2016. <https://www.nsnam.org/overview/what-is-ns-3/>.
- [19] “Instalación paso a paso de ns3”, enero 2016.
<https://www.nsnam.org/wiki/Installation>.
- [20] “Tutorial de ns3”, febrero 2016. <https://www.nsnam.org/docs/tutorial/html/>.
- [21] D. P. Sánchez de Rivera, *Fundamentos de Estadística*. Alianza Editorial, 2001.
- [22] “Crypto++ 5.6.0 benchmarks”, enero 2016. <http://www.cryptopp.com/benchmarks.html>.

-
- [23] “Documentación de ns3”, enero 2016. <https://www.nsnam.org/docs/release/3.24/dxygen>.