DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA, REDES Y COMUNICACIÓN DE DATOS

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN ELECTRÓNICA, REDES Y COMUNICACIÓN DE DATOS

TEMA: ESTUDIO COMPARATIVO BASADO EN LA SIMULACION DE ESCENARIOS ENTRE LOS PROTOCOLOS DE ENCAMINAMIENTO SAODV Y AODV UTILIZADOS EN REDES AD-HOC

AUTOR: ROJAS RIVERA DIEGO PAÚL

DIRECTOR: ING. AGUILAR DARWIN

SANGOLQUÍ

2015

CERTIFICADO

CERTIFICADO

DEPARTAMENTO DE ELECTRICA Y ELECTRÓNICA

CARRERA DE INGENIERIA EN REDES Y COMUNICACIÓN DE DATOS

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***ESTUDIO COMPARATIVO BASADO EN LA SIMULACION DE ESCENARIOS ENTRE LOS PROTOCOLOS DE ENCAMINAMIENTO SAODV Y AODV UTILIZADOS EN REDES AD-HOC***" realizado por el señor ***ROJAS RIVERA DIEGO PAÚL***, ha sido revisado en su totalidad y analizado por el software anti-Plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***ROJAS RIVERA DIEGO PAÚL*** para que lo sustente públicamente.

Sangolqui, 1 de Diciembre del 2015

ING.AGUILAR SALAZAR DARWIN LEONIDAS

DIRECTOR

AUTORIA DE RESPONSABILIDAD

AUTORÍA DE RESPONSABILIDAD

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELECTRICA Y ELECTRÓNICA

CARRERA DE INGENIERIA EN REDES Y COMUNICACIÓN DE DATOS

AUTORÍA DE RESPONSABILIDAD

Yo, ***ROJAS RIVERA DIEGO PAÚL*** , con cédula de identidad N° 1717947186, declaro que este trabajo de titulación "***ESTUDIO COMPARATIVO BASADO EN LA SIMULACION DE ESCENARIOS ENTRE LOS PROTOCOLOS DE ENCAMINAMIENTO SAODV Y AODV UTILIZADOS EN REDES AD-HOC***" ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 1 de Diciembre del 2015

ROJAS RIVERA DIEGO PAÚL

C.C.1717947186

AUTORIZACION

AUTORIZACIÓN

DEPARTAMENTO DE ELECTRICA Y ELECTRÓNICA

CARRERA DE INGENIERIA EN REDES Y COMUNICACIÓN DE DATOS

AUTORIZACIÓN

Yo, ***ROJAS RIVERA DIEGO PAÚL***, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "***ESTUDIO COMPARATIVO BASADO EN LA SIMULACION DE ESCENARIOS ENTRE LOS PROTOCOLOS DE ENCAMINAMIENTO SAODV Y AODV UTILIZADOS EN REDES AD-HOC***" cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Sangolqui, 1 de Diciembre del 2015

ROJAS RIVERA DIEGO PAÚL

C.C.1717947186

DEDICATORIA

Deseo dedicar este trabajo de titulación en primer lugar a Dios ya que sin él no hubiera sido posible este logro tan importante, a mis padres que me brindaron su apoyo y dedicación incondicionalmente para que día a día pueda cumplir esta gran meta, a mi hermana que con su ejemplo me ha brindado una enorme ayuda en busca de mis sueños, y a todas las personas que estuvieron en todo momento apoyándome tanto en mi vida personal como en este gran paso a la vida profesional.

Diego Paúl Rojas Rivera

AGRADECIMIENTO

Quiero agradecer principalmente a la Universidad de las Fuerzas Armadas por haberme formado más que como profesional como una persona correcta con valores y conocimientos, a todas las personas que me apoyaron incondicionalmente en el proceso de elaboración de este gran sueño, y de manera especial agradecer al Ing. Darwin Aguilar por haberme guiado durante toda la carrera y en este proyecto de titulación.

Diego Paúl Rojas Rivera

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

RESUMEN

Las redes inalámbricas surgieron como complemento de las redes cableadas, ya que la verdadera ventaja de las redes inalámbricas son las diferentes opciones de accesibilidad y movilidad que ofrecen frente a las cableadas. Debido al desarrollo que han tenido los dispositivos inalámbricos, la tecnología inalámbrica, ha ido evolucionando con mayor fuerza en los últimos años, alcanzando avances importantes en cuanto a las velocidades de transmisión, seguridad y cobertura, y que han permitido que se presente un nuevo escenario de operación, el enmarcado dentro de lo que se conoce como Redes Ad Hoc, Las redes Ad-Hoc nacen bajo el concepto de autonomía e independencia, al no requerir el uso de infraestructura pre-existente ni la necesidad de soportar su administración en esquemas centralizados como lo hacen las redes actuales, entre otras de sus características.

Para la implementación de una Red Ad-Hoc es necesario utilizar un nivel de seguridad en la trasmisión de datos, es aquí donde entran los protocolos de encaminamiento SAODV y AODV, ya que las redes ad hoc al ser inalámbricas son naturalmente vulnerables a diversos tipos de ataques, y con la seguridad incorporada por el protocolo SAODV se disminuirán la vulnerabilidad de la red y se brindará un mayor nivel de seguridad al momento de la comunicación entre equipos.

PALABRAS CLAVE

- PROTOCOLO-SAODV
- PROTOCOLO-AODV
- RED-AD HOC
- CONFIDENCIALIDAD
- INTEGRIDAD

ABSTRACT

Wireless networks emerged to complement wired networks, because the real advantage of wireless networks are different accessibility and mobility options offered against wired. Because of the development that have wireless devices, wireless technology has evolved more strongly in recent years, achieving significant advances in transmission speeds, security and coverage and that have allowed this new scenario of operation, framed within what is known as Ad Hoc networks, ad-hoc networks are born under the concept of autonomy and independence, it does not require the use of pre-existing infrastructure and the need to support centralized administration schemes as do today's networks, among other characteristics.

To implement a Network Ad-Hoc is necessary to use a level of security in data transmission, this is where routing protocols SAODV and AODV come as ad hoc networks to be wireless are naturally vulnerable to various types of attacks, and built by the SAODV security protocol vulnerability of the network will decrease and a higher level of security will be provided at the time of communication between computers.

KEYWORDS

- PROTOCOL-SAODV
- PROTOCOL-AODV
- NETWOTK-AD HOC
- CONFIDENTIALITY
- INTEGRITY

CAPíTULO 1

1. Marco teórico

1.1 REDES AD-HOC

1.1.1 Historia de las redes Ad-Hoc

En este capítulo se empezará con un pequeño resumen del nacimiento de las Redes Ad-Hoc, como en muchos nacimientos de las tecnologías en electrónica y otros campos, las redes inalámbricas nacen en el campo militar por la necesidad de comunicarse entre las diferentes zonas militares y así en los años 70 el Departamento de Defensa de los Estados Unidos (DARPA) inicia una investigación de una comunicación mediante paquetes de radio (PRNET), que básicamente utilizaba los protocolos ALOHA y CSMA.

La facilidad para montar terminales inalámbricos y el hecho de que estos tenían una facilidad y rapidez de despliegue, tuvo una gran acogida en el campo militar naciendo así el concepto de red inalámbrica.

El grupo DARPA es el precursor de los inicios de Internet, y en los años 90 recién se obtuvo un crecimiento importante, en la actualidad el objetivo principal de este proyecto es que todo mundo pueda acceder a Internet.

En los últimos años ha existido demanda para contar con tecnología inalámbrica en diferentes ámbitos por lo que se necesita de acceso móvil más rápido , fiable y alcanzar una reducción de costo de los equipos , la ventaja de accesibilidad y movilidad y que no se requiere de una infraestructura que ofrecen las redes inalámbricas frente a las redes cableadas han ocasionado un gran auge en las comunicaciones inalámbricas , debido al desarrollo que han tenido los dispositivos, la tecnología inalámbrica ha ido evolucionando con mayor fuerza, alcanzando avances importantes en cuanto a las velocidades de transmisión, seguridad y cobertura, y que han permitido que se presente un nuevo escenario de operación, el enmarcado dentro de lo que se conoce como Redes Ad Hoc.

Las redes Ad-Hoc nacen bajo el concepto de autonomía e independencia, al no requerir el uso de infraestructura pre-existente ni la necesidad de soportar su administración en esquemas centralizados como lo hacen las redes actuales, entre otras de sus características.

1.1.2 Características

Las redes Ad-Hoc inicialmente fueron creadas para campos hostiles e irregulares, pero con el paso del tiempo estas se fueron implementando en diferentes tipos de campos como aeropuertos, universidades, entornos comerciales, entre otros.

Las redes Ad-Hoc inalámbricas tienen un sin número de características pero entre las más destacadas e influyentes frente a otras tecnologías tenemos las siguientes:

- **Topología Dinámica**

En este tipo de redes los nodos están sujetos a cambiar de lugar de una forma rápida y fiable moviéndose en cualquier dirección y en cualquier momento comportándose de forma autónoma.

Pueden desaparecer del área de cobertura de un nodo y formar una nueva ruta o enlace con otro nodo que se encuentre dentro de su nuevo rango de cobertura.

- **Variabilidad del canal radio**

Esta característica hace referencia a la potencia que necesitan los dispositivos, debido a que es una red inalámbrica presenta mucha variación en las condiciones que propagan al canal de radio.

- **No usan una infraestructura para su red**

Este tipo de redes no necesitan una infraestructura existente de red disponible para poder funcionar, se comportan de forma autónoma, contando con protocolos propios de redes inalámbricas, así como programas de gestión de red que ayudan al control y administración de las redes.

- **Ancho de banda limitado**

Debido a que se trata de una red inalámbrica el ancho de banda será menor frente a una red cableada, este parámetro depende del número de mensajes necesarios que se

utiliza tan para establecer la comunicación entre nodos y por ende la comunicación total en la red.

- **Uso de Baterías**

Debido a que los nodos en la red tienden a moverse en cualquier momento y dirección estos se manejan con baterías, estas tienen capacidad de entregar energía en un tiempo limitado que dependerá de la movilidad y actividad que tenga el nodo al momento de establecer una comunicación.

1.1.3 Arquitectura de un nodo

Sabemos que un nodo dentro de una red Ad-Hoc funciona simultáneamente como receptor y como emisor , lo más común dentro de un nodo es que utilice una torre de protocolos TCP/IP y constantemente actualicen la tabla de rutas mediante un protocolo de encaminamiento que detallaremos más adelante.

A continuación mostraremos como está conformada la arquitectura de cada nodo Ad-Hoc:

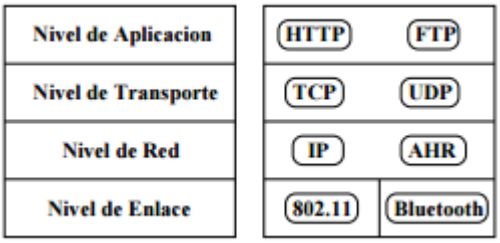


Figura 1 Arquitectura en capas de un nodo de una manet

(Ros, Evaluacion de Propuestas de Interconexion a Internet, 2004)

- **Nivel de enlace**

En la capa de enlace lo más relevante es el control de acceso al medio (MAC), como la redes Ad-Hoc utiliza el aire como medio compartido en donde se transmiten los datos mediante ondas electromagnéticas, es indispensable que cada nodo de alguna forma se ponga de acuerdo con el resto de la red para que no exista colisiones al momento de realizar la trasmisión de datos, para esto se han creado dos métodos para coordinar la comunicación.

- **Centralizado**

Existe un controlador o ´arbitro que va asignando el turno de palabra a cada uno de los interlocutores. Ningún nodo puede transmitir hasta que le llegue su turno.

- **Por contienda**

En este caso las redes Ad-Hoc utilizan el método por contienda, ya que no depende de ningún dispositivo central que organice la red, aunque también es conveniente que los nodos no tengan que esperar tiempo para transmitir, adicional en esta capa lo más adecuado es que cada nodo Ad-Hoc cuente con una antenas omnidireccional que le permitirá establecer comunicación con cualquier otro dispositivo siempre y cuando este se encuentre en el mismo rango de cobertura.

En esta capa las redes Ad-Hoc utilizan dos tecnologías propias del nivel para establecer la comunicación como son la tecnología 802.11 y Bluetooth.

- **IEEE 802.11**

Este estándar es el responsable del gran crecimiento que han tenido las redes WLANs, esta tecnología ha incursionado en el mercado de las redes inalámbricas con gran éxito y se ha iniciado varias investigaciones impulsando el uso de este protocolo en las redes Ad-Hoc.

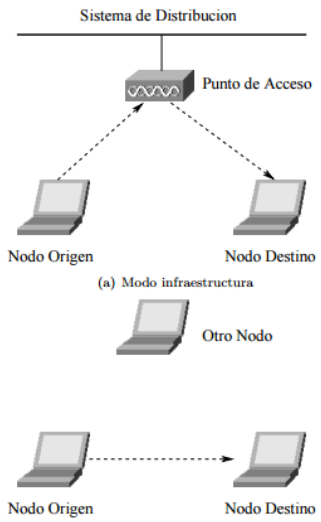


Figura 2 WLAN 802.11 en modo infraestructura e independiente

(Ros, Evaluacion de Propuestas de Interconexion a Internet, 2004)

El protocolo 802.11 puede ejecutarse de dos maneras: en modo infraestructura en donde se necesita de un punto de acceso para la comunicación entre nodos como se

muestra en la parte superior de la figura 2, y el modo independiente en donde los nodos se comunican entre si como se muestra en la parte superior de la figura 2, por lo que nos enfocaremos en el modo independiente que es el que emplea las redes Ad-Hoc.

- **Bluetooth**

Esta tecnología es más usada en redes de áreas personales (PAN), su creación se enfocó en comunicaciones de corta distancia y el consumo de energía tiene que ser limitado.

- **A nivel de red**

Es en este nivel donde los nodos de las redes Ad-Hoc se encargan de enviar los paquetes a su destino correspondiente, con esto se logra que las redes Ad-Hoc sea independiente de la tecnología de acceso al medio, en este nivel se utilizan los mismos protocolos que en el modelo IP.

Como los nodos Ad-Hoc actúan simultáneamente como receptor y transmisor o es lo mismo decir como host y router , entonces es indispensable que se ejecute algún algoritmo de enrutamiento, los protocolos tradicionales de las redes fijas (RIP, BGP, OSPF) no son adecuados para este tipo de entornos móviles por lo que es importante que el protocolo de enrutamiento para este tipo de redes se adapte a los continuos cambios de la red.

- **Nivel de transporte**

Como sabemos el protocolo UDP (User Datagram Protocol), es un protocolo de transporte no confiable y no orientado a la conexión, esto quiere decir que se envía paquetes sin saber si llegarán a su destino, son las capas superiores las que se encargan de verificar esta información, como UDP es sencillo se adapta fácilmente a las redes Ad-Hoc.

Algunas investigaciones de varios autores han sugerido la creación de un nuevo protocolo denominado TPA (Transport Protocol For Ad Hoc Networks), es un protocolo exclusivo para redes Ad Hoc que incluye algoritmos que detectan la caída del enlace y es capaz de recuperar la ruta perdida

A nivel de aplicación

Los protocolos a nivel de aplicación que utilizan las redes Ad-Hoc son los mismos que se utilizan en las redes habituales (DNS, HTTP, POP, IMAP, SNMP), estos protocolos son similares ya que los usuarios utilizaran la comunicación en las redes Ad-Hoc para usar las mismas aplicaciones que en redes tradicionales, aplicaciones como navegación web, correo electrónico entre otros.

1.1.4 Aplicaciones

Una de las características más importantes de las redes Ad-Hoc es la flexibilidad, y por tener la capacidad de operar en cualquier lugar en cualquier momento sin ningún tipo de infraestructura las hace muy interesantes al momento de poder implementarlas, es por eso que aunque nació de una necesidad militar hoy en día se utilizan en varios campos y los servicios que ofrecen, como se muestra en la siguiente tabla:

Tabla 1

Campos de aplicación de las tecnologías inalámbricas ad hoc

Aplicaciones	Servicios ofrecidos
Redes tácticas	Comunicaciones en operaciones militares Campos de batalla automatizados
Redes de sensores	Recogida de datos en tiempo real, generalmente altamente correlados en espacio y tiempo
Servicios de salvamento y emergencia	Operaciones de búsqueda y rescate Sustitución de redes con infraestructuras en caso de catástrofes naturales
Entornos comerciales	Comercio electrónico Acceso remoto a los registros de los cliente desde una base de datos centralizada Oficina móvil Servicios vehiculares
Redes para particulares y empresas	Redes de área local inalámbricas (WLAN, *Wireless Local Area Network*) para hogares u oficinas Redes de área personal (PAN, *Personal Area Network*)
Aplicaciones educativas	Configuración de clases virtuales Configuración de comunicaciones *ad hoc* en reuniones, conferencias y congresos
Ocio	Juegos multi-usuario Robots mascota Accesso a Internet en exteriores
Servicios de localización	Servicios de seguimiento Servicios de información

Fuente:(Campos de aplicación de las tecnologías inalámbricas ad hoc)

1.1.5 Ventajas y desventajas

Debido a que se trata de una red inalámbrica y uso un medio compartido para comunicarse que es el aire, se tendrán ventajas al momento de usarlas asi como también desventajas frente al uso de otras tecnologías de redes inalámbricas y redes cableadas.

A continuación vamos a mencionar las principales ventajas y desventajas de las redes Ad-Hoc.

VENTAJAS

- La flexibilidad de estas redes son la principal ventaja al momento de implementarlas, ya que no requiere una infraestructura previa ni de un punto de acceso o central que administre la red para poner en funcionamiento la red, adicional se puede añadir nuevos nodos sin que esto implique un alto coste como en una red cableada.
- La topología dinámica hace que los nodos puedan moverse arbitrariamente en cualquier sentido y cualquier lugar, por lo que la topología de la red puede cambiar en cualquier momento sin obstaculizar el rendimiento de la red.
- El costo de implementación sin duda es otro aspecto importante en las redes Ad-Hoc, en comparación a redes cableadas y otro tipo de redes inalámbricas las redes Ad-Hoc eliminan costos en infraestructura fija y reduciendo el consumo de energía en las estaciones
- La reutilización del espectro hace referencia a una disminución de recursos, ya que la comunicación es punto a punto y no es necesario comunicarse primero con un punto de acceso o una estación central base para comunicarse con otros nodos como se hace en otros tipos de redes.

DESVENTAJAS

- Debido a que se trata de una red inalámbrica móvil se utilizaran equipos móviles y por ende el uso de baterías, lo cual afecta directamente el rendimiento del nodo al momento de comunicarse con el resto de nodos

perdiendo área de retransmisión de la señal o a la vez perder conectividad con la red por no contar con suficiente energía para continuar operando.

- El tema de la seguridad es muy importante ya que se trata de una red inalámbrica y en el medio compartido que utiliza para transmitir pueden tener acceso cualquier intruso, se debe cuidar la confidencialidad de los datos para que ningún intruso pueda acceder a la información de la red.
- El ancho de banda limitado es otra de las desventajas de este tipo de redes, a comparación con redes cableadas es mucho menor debido a las atenuaciones e interferencias del medio.
- La capacidad de proceso también es una limitante ya que varios equipos en este tipo de redes no tienen una gran capacidad de procesamiento y es por eso que las funciones de enrutamiento y trasmisión de datos deben reducir al máximo la capacidad de energía que tiene cada uno para evitar complicaciones al momento de poner en funcionamiento la red.
- Latencia alta es la que se percibe cuando un nodo que está inactivo por ahorrar energía empieza a transmitir, al recuperarse y ponerse activo puede tardar cierto tiempo para cumplir su función.
- Errores de transmisión en una red inalámbrica es común, adicional en esta red se usa un protocolo de transporte poco confiable (UDP) que no garantiza una entrega correcta de paquetes al destino.

1.2 Seguridades en redes Ad-Hoc

1.2.1 Introducción

Una red Ad-Hoc al ser una red inalámbrica es vulnerable a distintas clases de ataques, se han creado varios protocolos de encaminamiento que también son vulnerables a estos tipos de ataques, sin embargo existen distintos planteamientos para implementar un nivel máximo de seguridad a las redes inalámbricas y específicamente a las redes Ad-Hoc, uno de los protocolos que implementa seguridad a una red Ad-Hoc es el SAODV que detallaremos su funcionamiento más adelante.

Para entender mejor este tema se debe tener claro los diferentes tipos de nodos que intentarán violar la seguridad de las redes Ad-Hoc.

- Un nodo malicioso será un atacante que no puede autenticarse a sí mismo como nodo legítimo debido a la falta de información criptográfica válida.
- Un nodo comprometido será un atacante interno quien se comporta de forma malintencionada, sin embargo puede ser autenticado por la red como un nodo legítimo y obtener la confianza de los otros nodos.

1.2.2 Ataques a redes Ad Hoc

Para empezar a mencionar los diferentes ataques a las redes inalámbricas de tipo Ad-Hoc, en primer lugar describiremos ciertos requisitos de seguridad que deberían cumplir estas redes:

- **Autenticación**

La autenticación es un aspecto fundamental en la seguridad de toda red, ya que permite verificar que la información que se transmite es auténtica y no es adulterada, para esto se necesita de la colaboración de los nodos demostrando que su identidad es verdadera, sin una previa autenticación un intruso podría hacerse pasar por un equipo de la red y acceder a la información dentro de la red.

- **Disponibilidad**

Otra característica importante en toda red es la disponibilidad, esto significa que cada nodo de la red debería estar operativo en todo momento independiente de algún ataque que se pueda dar como denegación de servicio, falta de energía entre otros.

- **Confidencialidad**

La confidencialidad verifica que un nodo malicioso no pueda acceder a información que se está transmitiendo en la red entre dos nodos, una red inalámbrica cuenta con mecanismos para prevenir los datos y además asegurarse que la información no sea mostrada a ningún nodo que no pertenezca a la red.

- **Integridad**

En una comunicación segura debe existir una integridad de la información, esta característica nos indica que la información que se envía es la misma que se recibe sin haber sido manipulada durante la transmisión, adicional se verifica que todos y cada uno de los paquetes de los datos sean los mismos al momento de su envío y recepción por parte de los nodos, estos paquetes pueden ser manipulados por nodos maliciosos o a su vez por decaimiento del medio de propagación

- **No Repudio**

No repudio significa que un nodo que envía cierta información no puede cancelar el envió de la misma manera el nodo receptor no podrá cancelar la recepción, este método es muy importante al momento de detectar un nodo malicioso en la red, ya que si envía una información errónea no podrá detenerla en el transcurso de la transmisión y el nodo receptor al recibirla informará al resto de nodos que se trata de un intruso dentro de la red.

- **Autorización**

La autorización define un reglamento para indicar lo que un nodo tiene o no permitido hacer dentro de la red como por ejemplo a cierta información que determinado nodo no tienen acceso.

Teniendo en cuenta ciertos parámetros de seguridad que deben tener las redes Ad-Hoc a continuación se detalla dos niveles de ataques en general que pueden sufrir estas redes

- Ataque a los algoritmos de funcionamiento de la red como es el encaminamiento
- Ataque a los algoritmos de seguridad, afectando directamente a los algoritmos de administración de claves que usan estas redes.

Y los ataques pueden clasificarse en otros dos grupos:

Pasivos

Son los ataques que afectan solo a la transmisión de datos, se dedican a escuchar el envió de datos e introducen entre sus mecanismos de ataque a canales secretos, análisis de tráfico, sniffer entre otros.

Activos

Este tipo de ataque ya utiliza acciones realizadas por el nodo malicioso entre ellas modificar, eliminar, repetir los datos a través de la red, los nodos maliciosos intentan una y otra vez cambiar el comportamiento del protocolo y afectar su funcionamiento,

El protocolo puede revelar sin darse cuenta la información a los atacantes pasivos cuando intenta lanzar mensajes de alertas a los atacantes activos.

Se podría añadir otra clasificación según la procedencia del atacante:

Externos

Los ataques externos son aquellos que ocasionan tráfico en la red, se dedican a generar información errónea, y a ocasionar danos en el funcionamiento de los servicio o aplicaciones de la red, una manera sencilla de contrarrestar estos ataques es usando mecanismos de seguridad comunes como cortafuegos, método de cifrado, entre otros.

Internos

Los ataques internos son más fuertes y peligrosos que los externos, ya que en estos ataques los nodos ya se encuentran formando parte de la red y por lo tanto son defendidos por los mecanismos de seguridad que la red ofrece, por ende pueden asociarse con otros nodos y hacer uso de los mecanismos comunes para operar.

1.2.3 Esquemas de seguridad

Existe un esquema de seguridad para redes Ad-Hoc llamado seguridad cooperativa que a su vez se dividen en 3 grupos:

Fundamentado en monedas virtuales

En este tipo de seguridad los nodos reciben un regalo por colaborar en las operaciones de la red, dentro de este tipo de seguridad se encuentran los mecanismos llamados nuglets.

Fundamentado en monitorizado local

Cada usuario realiza un seguimiento a sus vecinos evaluando su métrica para cada uno, con esto consiguen saber el prestigio con el que cuenta cada usuario, y los nodos que no quieran compartir información los van aislando poco a poco.

Híbridos

Este subtipo de seguridad es una combinación de los dos anteriores, cada usuario debe poseer un identificador para operar en la red, son los vecinos los que juzgan su comportamiento y decidirán si puede renovar el identificador para seguir operando o no.

En la siguiente tabla presentamos algunas características de seguridad cooperativa.

Tabla 2

Características de seguridad cooperativa

Nuglets	Confidant
Se limitan a interacciones uno-a-uno	Un mal comportamiento produce una mala reputación que se propaga a más de un nodo
Requiere un módulo de seguridad a prueba de falsificaciones	No requiere módulo de seguridad a prueba de falsificaciones

Fuente:(Estudio del rendimiento y la seguridaden redes ad hoc, 2006)

Nuglets

Los usuarios en esta red deben tener 2 características fundamentales, desinteresados a saturar la red con sobrecarga y deben estar motivados para cooperar con el funcionamiento de la red principalmente en el reparto de paquetes con el resto de usuarios.

Confidente

En este ejemplo de seguridad los nodos evitan vecinos egoístas creando rutas y monitoreando a cada vecino y a la vez detectan nodos maliciosos, adicional los nodos

egoístas tienen una desventaja ya que los nodos propios de la red ya no enviaran información y solo recibirán paquetes de los nodos maliciosos.

Si por causas como ser culpado por error, o cambian su conducta y decide ser un nodo honesto durante cierto periodo entonces la red se dará cuenta del cambio y decidirá integrarlo nuevamente para operar dentro de la red

1.2.4 Criptografía

Dentro de toda red y sistemas de seguridad existe un método fundamental que se implementa para evitar el acceso no autorizado a la información , y mucho más en una red inalámbrica que está expuesta a varios tipos de ataques por el medio compartido en el que transmiten la información , la criptografía es un método que consiste en transformar información clara en información o señales indescifrable por ajenos que no conozcan el secreto, adicional esta característica descubre cuando la información no es integra y preserva la confidencialidad de la información .

La seguridad a la que está sometida la criptografía se basa en las siguientes características:

- **Clase de algoritmos**

No es necesario que los algoritmos utilizados sean secretos, pero deben ajustarse a preguntas complicadas de resolver, a esto se lo conoce como clave.

- **Implementación de algoritmos**

Es preferible librarse de una implementación defectuosa que acabar uno mismo con el propio algoritmo

- **Administración de claves**

Existe un sistema en donde 2 nodos comparten un secreto, este sistema es llamado simétrico, en otro caso si el número de nodos aumenta entonces los secretos aumentan de acuerdo al número de nodos crecientes, para este caso existe un sistema en donde no es necesario compartir un secreto, este sistema es denominado asimétrico pero no garantiza la integridad de las claves ni tampoco los mecanismos que protegen la confidencialidad de los secretos de los nodos.

- **Clave publica**

La clave pública es utilizada en el sistema asimétrico, este sistema usa una única clave o un único secreto para descubrir la información de la red, además este sistema utiliza una clave para la emisión de mensajes y otra clave para su recepción, estas dos claves constan de dos partes, una fracción es privada que es secreta, y otra fracción es publica que no necesariamente tiene que ser confidencial.

- **Firma Digital**

La criptografía que consta de clave pública tiene 2 claves, la privada y la pública, la clave privada es la encargada de generar una firma y debe permanecer secreta, mientras que la clave pública es la encargada de verificar una firma, esta puede ser difundida pero debe permanecer integra.

El dueño de las 2 claves (pública y privada) es el único que tiene acceso y el único que puede crear las firmas digitales, en cambio la clave pública puede ser usada por varios receptores para que puedan comprobar la firma.

En la criptografía existe un algoritmo importante denominado **función hash**, esta función básicamente realiza una concentración o compresión de un mensaje denominado resumen, este mensaje a ser comprimido debe firmarse pero al ser comprimido y convertirse en un resumen tiene la función de verificar claramente si existió alguna modificación del mensaje inicial por más mínima que sea y será casi muy difícil que se pueda descifrar un mensaje inicial a partir de un resumen.

Para **firmar digitalmente** un texto se debe realizar un resumen del mensaje utilizando una clave privada, este proceso es la firma digital que servirá para el resto de nodos de la red para que en cada salto que vaya dando la información por cada usuario estos puedan verificar la integridad del mensaje y vayan firmando digitalmente el mensaje, y asi pueda llegar al usuario el texto completo sin ningún tipo de perdida de integridad o adulteración del mensaje inicial.

1.3 Protocolos de Encaminamiento

En general el encaminamiento en las redes Ad-Hoc es el procedimiento en el que se transmite la información desde un nodo origen hasta un nodo destino, es el proceso mediante el cual los nodos de la red seleccionan el camino más apto para transmitir los mensajes dentro de la red.

Ya entrando a lo que son los protocolos de encaminamiento , estos son los encargados de organizar la red y su principal función es trazar la ruta más eficiente entre un usuario inicial y final, y garantizar que la información se entregue integra y a tiempo, para establecer las rutas los protocolos deben procurar que los nodos optimicen el uso de energía y ancho de banda ya sea modificando las cabeceras de los paquetes o el algoritmo que cada protocolo implemente para llevar a cabo una comunicación optima dentro de la red .

Entre las principales características que deberán tener los algoritmos de cada protocolo para poner en funcionamiento la red Ad-Hoc tenemos los siguientes:

- Rutas convergentes, bajo tráfico, capacidad de establecer rutas alternativas para evitar congestión en la red.
- Algoritmos para optimizar el consumo de ancho de banda y energía en los nodos al momento de posibles cambios de topología o congestión en la red.
- Escalabilidad en la red, de tal forma que si la red crece con un número elevado de usuarios, no afecte en el rendimiento de la misma
- Seguridad en la red enfocándose en aspectos como confidencialidad, integridad, control de acceso.
- Tener destreza para brindar la máxima calidad de servicio, que es indispensable en ciertas aplicaciones.

1.3.1 Técnicas de encaminamiento

Las técnicas de encaminamiento son ciertas características o métodos que cumplen los protocolos y se dividen en tres grupos:

- **Vector Distancia**

Este método utiliza tablas en todos los nodos vecinos de la red que divulgan dicha información, con esta información cada nodo podrá descubrir la ruta más cercana al destino y entregar la información a tiempo.

- **Estado de Enlace**

Los protocolos que utilizan este método cuentan con una tabla de encaminamiento que contiene la topología de la red, cada nodo identifica cuáles son sus vecinos y a que distancia se encuentra de ellos, con esta información va trazando el mapa o topología de la red, cada nodo actualiza su tabla de encaminamiento con la información del estado o coste del enlace, esta técnica no es recomendada ya que suele crear lazos en redes que modifican su topología rápidamente.

- **Encaminamiento Fuente**

Este tipo de método utiliza información de encaminamiento en las cabeceras de los paquetes, esta información la proporciona el remitente, una ventaja de esta técnica es que evita los lazos, pero la sobrecarga del protocolo es alta, adicional no es recomendable para redes con topologías cambiantes ya que las rutas de un paquete se pueden anular en un determinado trayecto de la información.

1.3.2 Clasificación de los protocolos de encaminamiento

Dentro de las redes Ad-Hoc podemos clasificar los protocolos de encaminamiento de acuerdo a varios criterios, pero los más comunes son los protocolos proactivos y reactivos.

1.3.2.1 Protocolos de encaminamiento proactivos

Los protocolos de encaminamiento proactivos cada cierto tiempo cambian los mensajes de control entre los nodos de la red Ad-Hoc, de esta manera consigue mantener las rutas entre nodos actualizadas , adicional este tipo de protocolos tienen la facilidad de adaptarse a los cambios de topología descubriendo así cuando un enlace deja de operar y cuando un nodo nuevo se introduce a la red se envía un mensaje tipo broadcast con esta información para advertir que hay nuevos integrantes, con estas características tienen la capacidad de elegir la mejor ruta en cada momento, mientras la información de control se actualice con más frecuencia las rutas también se

actualizarán rápidamente , pero esto conlleva a un mayor trabajo del protocolo y se produce una sobrecarga, que disminuye el ancho de banda a la red.

Este tipo de protocolos utilizan dos algoritmos, el vector distancia y el estado de enlace a continuación ejemplos de protocolos que utilizan dichos algoritmos.

Vector Distancia

- DSDV (Destination Sequenced Distance Vector)
- WRP (Wireless Routing Protocol)

Estado de Enlace

- OSPF (Open Shorest Path First)
- OLSR (Optimized Link State Routing)
- STAR (Source Tree Adaptive Routing)
- GSR (Global State Routing)

En resumen la principal ventaja de este tipo de protocolos es que responde rápidamente cuando se requiere de una ruta actualizada, la desventaja es que por la actualización de información periódica existe sobrecarga principalmente cuando la topología de la red es cambiante de manera frecuente.

1.3.2.2 Protocolos de encaminamiento reactivos

Los protocolos de encaminamiento reactivos surgen con la necesidad de mejorar las limitaciones de ancho de banda que tienen las redes Ad-Hoc, esta limitación se genera cuando la red se actualiza mediante los mensajes de control y es ahí donde los protocolos reactivos o baja demanda realizan su parte, estos protocolos no actualizan periódicamente sus rutas únicamente se mantienen las rutas con el destinatario de los nodos activos de la red, cuando un nodo necesita establecer comunicación con otro envía un mensaje tipo broadcast requiriendo la ruta hacia el destino mediante un mensaje tipo RREQ , el nodo que tenga dicha ruta responde con un mensaje tipo RREP, la principal limitante de este tipo de protocolos es que responden tardíamente ante cambios de topología y tienen muchos retardos al momento de establecer la comunicación entre nodos.

Algunos ejemplos de protocolos reactivos son:

- AODV
- DSR
- LMR

1.3.3 Estudio comparativo

Para realizar un estudio comparativo entre estos protocolos se escogió los parámetros más importantes y los que más influyen dentro del rendimiento de una red Ad-Hoc

- **Latencia**

Es la acumulación de retardos en una red, en este caso en los protocolos proactivos se tiene menor latencia debido a la actualización de rutas frecuentemente, caso contrario en los protocolos reactivos existe una latencia elevada debido a que solo se buscan rutas cuando se requiere establecer una comunicación entre nodos.

- **Frescura de rutas**

Se refiere a las rutas actualizadas o nuevas de la red , en los protocolos proactivos estas rutas se refrescan cada cierto periodo evitando asi latencia al momento de establecer la comunicación , en los protocolos reactivos solo se busca una ruta cuando un nodo desea comunicarse con otro nodo.

- **Desperdicio de ancho de banda**

En el caso de los protocolos reactivos existe una utilización menor de ancho de banda debido a que no están constantemente actualizando la información de control en toda la red, cosa que no sucede en los protocolos proactivos.

- **Procesamiento**

El procesamiento de los equipos afecta directamente a las baterias de los equipos de una red Ad-Hoc, en los protocolos proactivos existe un alto procesamiento al momento de actualizar la informacion de control y la actualizacion de rutas frecuentemente.

En la siguiente tabla resumimos las ventajas y desventajas de los protocolos proactivos y reactivos, un + significa una caracteristica favorable para cada protocolo mientras que un – significa una desventaja.

Tabla 3

Ventajas e inconvenientes de los protocolos de encaminamiento proactivos y reactivos

	Proactivos	Reactivos
Latencia	+	-
Frescura rutas	+	-
Overhead	-	+
Procesamiento	-	+

Fuente:(Ros, 2004)

1.4 PROTOCOLO AODV

1.4.1 Introducción

El protocolo de encaminamiento para redes Ad-Hoc AODV es uno de los protocolos en el que se basa este estudio, AODV es un protocolo de tipo reactivo que como se vio en el capítulo anterior solo busca rutas cuando un nodo necesita establecer comunicación con otro nodo y evitar sobrecarga en la red.

Los mensajes que se envían desde los nodos origen no tienen información sobre el trayecto de la ruta, solo conocen el origen y el destino, adicional AODV emplea el algoritmo vector distancia, los mensajes utilizan un número de secuencia por cada destino y verificar si una ruta es antigua o nueva y así evitar la formación de bucles, para un funcionamiento óptimo del protocolo cada nodo deben mantener su número de secuencia actualizado.

1.4.2 Funcionamiento

El protocolo AODV usa mecanismos combinados de dos protocolos antecesores, DSDV y DSR usando de manera eficaz el uso del ancho de banda en la red, y tiene la capacidad de responder rápidamente ante los cambios de topología frecuentes.

Con el objetivo de mantener la información más fresca utiliza el número de secuencia, cada nodo o estación se encarga de mantener su propio número de secuencia, cada nodo incrementa dicho número después de enviar la información de control, al igual que el número de secuencia cada nodo también cuenta con un identificador único dentro de la red, la combinación de ambos hace posible verificar la información verdadera de la obsoleta.

Por ejemplo si un nodo obtiene dos paquetes con un mismo identificador pero con distintos números de secuencia entonces la información más fresca será insertada en el paquete con mayor número de secuencia.

Cada nodo de la red guarda toda la información mediante tablas que tiene las siguientes entradas:

- **Dirección Ip destino**

Cada nodo en su tabla contiene la dirección IP destino, sin embargo no conoce cómo llegar hacia él.

- **Numero de secuencia del destino**

Este valor se obtiene de la información de control y hace referencia al número de secuencia que pertenece al destino.

- **Identificador de validez del nodo destino**

Si por cualquier razón el nodo origen no logra comunicarse con el nodo destino ya sea porque un enlace dejo de operar o la ruta ha expirado, entonces el número de secuencia del nodo destino se descartará ya que es inválido.

- **Número de saltos**

Se refiere al número de saltos que se necesitan para llegar desde el nodo origen hacia el destino

- **Siguiente estación**

Cada tabla contiene una entrada que proporciona información del siguiente salto o el siguiente nodo para alcanzar al destino.

- **Antecesores**

Los nodos contienen la información del historial de estaciones que han pasado en la búsqueda del trayecto para llegar al destino, esta búsqueda se la realiza mediante el algoritmo de descubrimiento de ruta que utiliza el protocolo AODV.

- **Tiempo de vida de ruta**

Cada ruta almacenada en una tabla tiene un tiempo de vida útil, cuando dicho tiempo expira la ruta es descartada y se borra de la tabla, esto con el fin de no sobrecargar la red.

- **Otros indicadores**

Esta entrada muestra indicadores adicionales, como por ejemplo si una ruta dejó de operar este campo nos indica si la ruta puede ser restablecida o hay que buscar otra ruta alternativa.

1.4.3 Descubrimiento de ruta

El proceso de descubrimiento de ruta se realiza cuando una estación o nodo desea comunicarse con otra estación pero no tiene la información en su tabla de rutas para llegar hacia dicha estación, para este tipo de proceso existen dos tipos de mensajes, el mensaje de petición (RREQ) y el mensaje de respuesta de ruta (RREP).

Adicional se pueden también definir dos tipos de rutas , el camino que se forma de regreso desde el destino hacia el origen proyectados por los mensajes de respuesta (RREP), y el camino de ida que es el que recorrerán los mensajes desde el origen hacia el destino, a continuación explicaremos como se forman cada uno de estos caminos.

Camino de regreso

Cuando un nodo desea comunicarse con un nodo destino pero ignora cómo alcanzarlo envía un mensaje de petición (RREQ) de tipo broadcast , el mensaje contiene las direcciones IP y números de secuencia del origen y destino, previo al envió del mensaje el nodo origen incrementa su número de secuencia para evitar que se procesen peticiones anteriores, y en el campo del número de secuencia de destino se introduce el valor final que se le entregó, para evitar que se haya solicitado esta ruta anteriormente o el destino haya sido desconocido.

Si un nodo no es el destino y desconoce la ruta hacia el destino entonces reenvía el mensaje a los siguientes nodos de la red hasta alcanzar al nodo destino, para evitar un uso excesivo de ancho de banda el mensaje de petición (RREQ) tiene un tiempo de vida limitado que cuando expira el mensaje es descartado, si no se tiene respuesta del destino por parte de los nodos de la red entonces el nodo origen envía nuevamente un (RREQ) y el tiempo de vida del mensaje (RREQ) se duplica, generalmente el número de intentos para descubrimiento a través del mensaje de petición (RREQ) de rutas es dos.

Los nodos intermedios anotan de donde viene la solicitud y la almacenan, y con esto van trazando las posibles rutas hacia el destino, el proceso que traza el camino de vuelta se termina cuando un nodo es el destino o tiene la ruta para acceder al mismo.

En la figura 3 se observa 4 rutas que son el trayecto de los mensajes RREQ, 2 de ellas van directamente al origen y las otras 2 son descartadas.

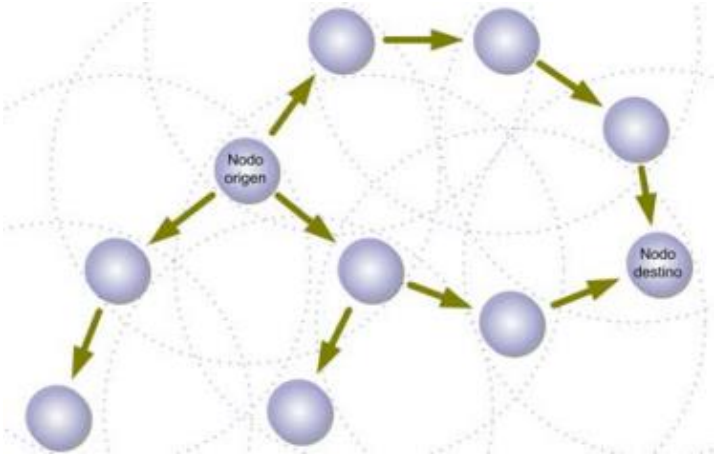


Figura 3 Inundación con mensajes RREQ

(Gil M. E., Estudio de eficiencia del protocolo AODV, 2009)

Camino de ida

Cuando un nodo que recibe el mensaje de petición es el nodo destino o tiene un ruta para acceder al destino, entonces se envía un mensaje de repuesta de ruta (RREP), se considera que un nodo intermedio puede acceder al destino si su número de secuencia del destino es mayor al número de secuencia del nodo solicitante, si el nodo destino es quien genera le mensaje de respuesta de ruta (RREP) entonces emitirá el número de

secuencia máximo entre los números de secuencia solicitantes y su propio número de secuencia, el mensaje (RREP) se envía de forma unicast desde el destino hacia el origen cop3iando el mismo camino de los mensajes (RREQ).

En la figura 4 se observa los posibles caminos de ida que se han formado, en este caso es el nodo destino quien envía el mensaje de respuesta , el nodo origen recibe dos mensajes RREP con un mismo número de secuencia debido a los dos caminos de ida generados , entonces se elegirá la ruta que contenga el menor número de saltos en este caso se eligió la ruta de tres saltos como se indica en la figura #2, para finalizar cuando el nodo origen recibe el mensaje de respuesta almacena la ruta hacia el destino y empieza a transmitir los paquetes .

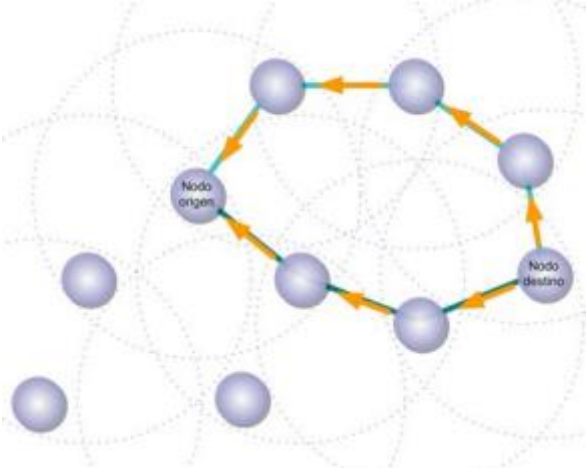


Figura 4 Reenvió de mensajes RREP

(Gil M. E., Estudio de eficiencia del protocolo AODV, 2009)

Si los nodos intermedios reciben la información de la ruta definitiva entonces deberían actualizar su tabla de rutas, un nodo decide actualizar su tabla si el nuevo número de secuencia del nodo destino es mayor al que tenía en su tabla para dicho nodo, o también cuando los dos números de secuencia son iguales y el número de saltos es menor al que figuraba en su tabla.

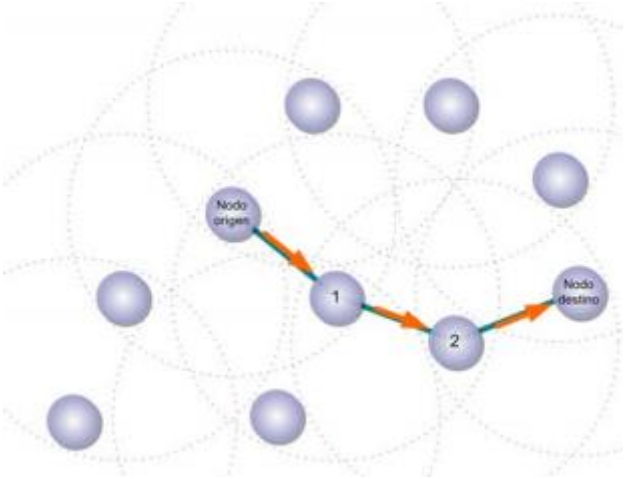


Figura 5 Envió de paquetes de datos sobre el camino de ida

(Gil M. E., Estudio de eficiencia del protocolo AODV, 2009)

1.4.4 Mantenimiento de ruta

La función principal del mantenimiento de rutas en el protocolo es el de informar a un nodo origen cuando un enlace falle, de esta manera puede buscarse una nueva ruta, por ejemplo si un nodo intermedio cambie de posición deberá informar a todos sus vecinos que necesitan de esta estación para transmitir, el mensaje se lo envía al resto de saltos y la ruta antigua será eliminada.

Para detectar la falla de un enlace el protocolo AODV utiliza los mensajes HELLO , estos mensajes son enviados periódicamente por todos los nodos hacia sus vecinos para anunciar su presencia en la red, si un nodo deja de recibir un mensaje HELLO de algún vecino puede considerar que el enlace con dicho vecino ha dejado de operar, e inmediatamente envía un mensaje de error de ruta (RERR) a sus vecinos y estos a su vez lo envían a los nodos que tengan rutas con el nodo afectado, adicional cuando el nodo origen recibe el mensaje de error de ruta puede iniciar nuevamente el proceso de descubrimiento de ruta.

1.4.5 Formato de mensajes AODV

El protocolo AODV contiene 4 tipos de mensajes de control principales, en la siguiente figura se observa cómo se intercambian los mensajes de control sobre la red, con las flechas se indica el sentido en que se envían los mensajes desde un nodo origen a un nodo destino.

El mensaje tipo HELLO se envían periódicamente para que cada nodo conozca si los enlaces con los nodos vecinos están operativos por ese motivo son bidireccionales, los mensajes tipo RREQ se envían para descubrir rutas hasta encontrar al nodo destino es por eso que solo va en la ruta de ida, los mensajes tipo RREP nos indican que el destino ha sido encontrado y van en sentido contrario a la trayectoria de los mensajes RREQ, y finalmente los mensajes RERR son de error de ruta y se envían hacia el destino cuando detectan que algún enlace ha fallado es por eso que van en la ruta de vuelta.

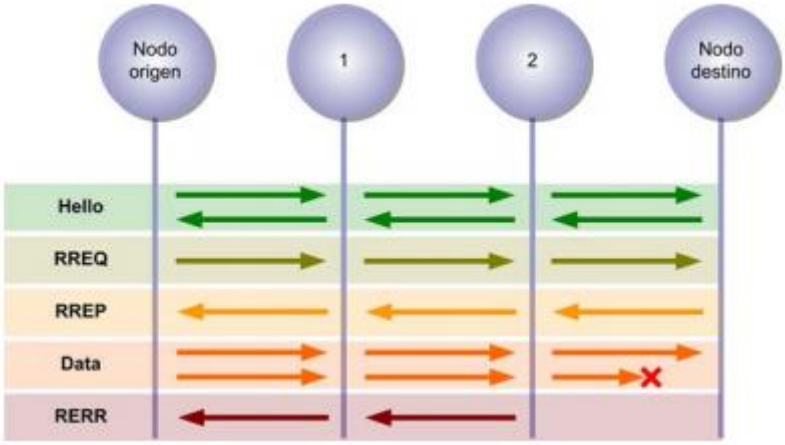


Figura 6 Mensajes de encaminamiento en AODV

(Gil M. E., Estudio de eficiencia del protocolo AODV, 2009)

Cuando la red está compuesta por nodos unidireccionales, los nodos pueden canjear otros mensajes de certificación de respuesta de rutas pero este tipo de mensajes no son usuales debido a la comunicación en una sola dirección de la red, estos mensajes son de tipo RREP-ACK.

Mensajes RREQ

Como se mencionó anteriormente para que un nodo se pueda comunicar con otro nodo por primera vez se debe empezar el proceso de descubrimiento de ruta, el nodo origen debe enviar un mensaje de petición a toda la red de tipo RREQ no sin antes incrementar su número de secuencia, este mensaje almacenará durante un periodo de

tiempo la dirección del origen y su identificador para no volver a procesarlo en caso de recibirlo nuevamente.

Este tipo de mensaje contiene la dirección IP y el número de secuencia tanto del origen como del destino, un contador de saltos total hacia el destino, cuando un nodo recibe este tipo de mensajes almacena en su tabla la información del nodo que la envió para establecer la ruta de vuelta hacia el origen, si un nodo recibe un RREQ con un mismo número de secuencia de otro mensaje RREQ enviado anteriormente lo descarta.

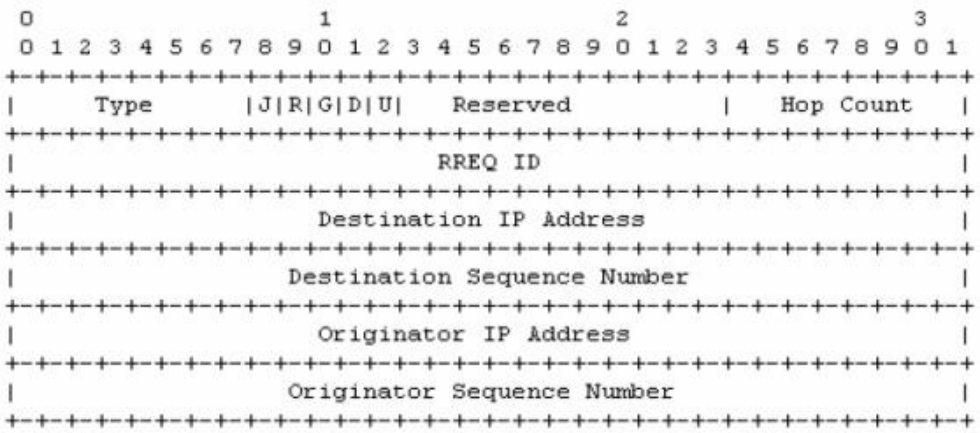


Figura 7 Formato de mensaje Route Request

(Monroy, 2012)

Mensajes RREP

Si el nodo destino recibe el RREQ envía un mensaje de respuesta de ruta RREP al vecino que le envió el RREQ, y este a su vez lo reenvía al nodo anterior , así sucesivamente hasta llegar al nodo origen, en el caso de que un nodo intermedio tenga guardada en su tabla la ruta del nodo destino entonces él será el encargado de enviar el RREP al origen y cada nodo intermedio guardará esta ruta encontrada en su tabla, si el origen recibe más de dos RREP elegirá el que contenga el menor número de saltos hacia el destino y la actualiza en su tabla para ser usada , si el origen transcurrido un tiempo no recibe una respuesta RREP entonces enviara nuevamente un RREQ a toda la red incrementando el número de secuencia como una petición nueva para que los nodos dentro de la red no lo descarten.

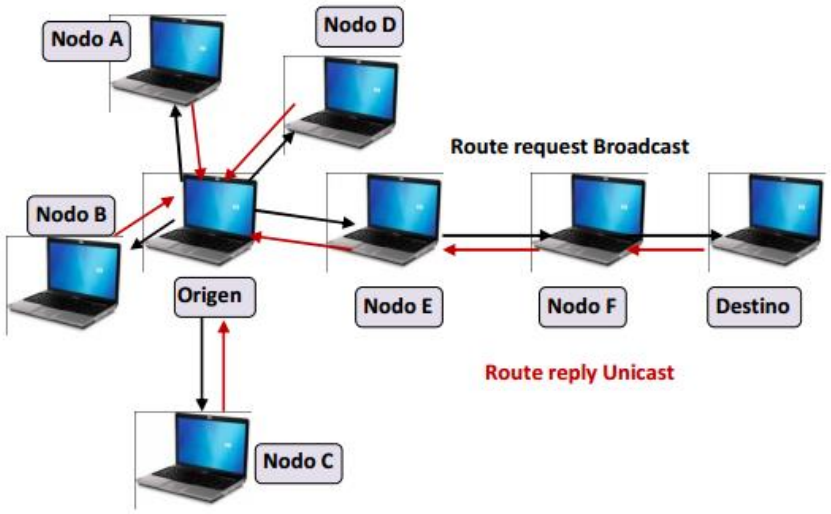


Figura 8 Funcionamiento de los mensajes Route Request y Route Reply

(Monroy, 2012)

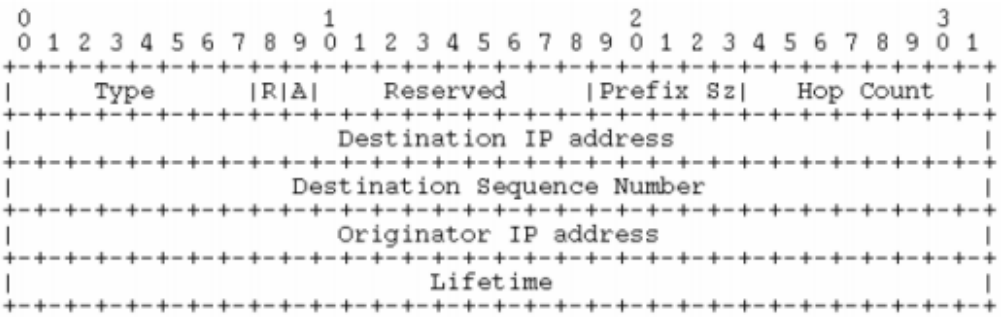


Figura 9 Formato de mensaje Route Reply

(Monroy, 2012)

Mensajes HELLO

El protocolo de encaminamiento AODV emplea los mensajes HELLO para el mantenimiento de rutas, los vecinos envían periódicamente este tipo de mensajes para anunciar su presencia en la red, pero únicamente participan las estaciones que se están comunicando esto con el fin de no sobrecargar la red.

Mensajes RRER

Los mensajes RERR son enviados de un nodo a sus vecinos cuando detectan que un enlace ha dejado de operar, los vecinos a su vez lo envían a los nodos que tengan rutas con el nodo afectado hasta llegar al origen, adicional cuando el nodo origen recibe el mensaje de error de ruta puede iniciar nuevamente el proceso de descubrimiento de ruta, este tipo de mensaje no contiene campos modificables.

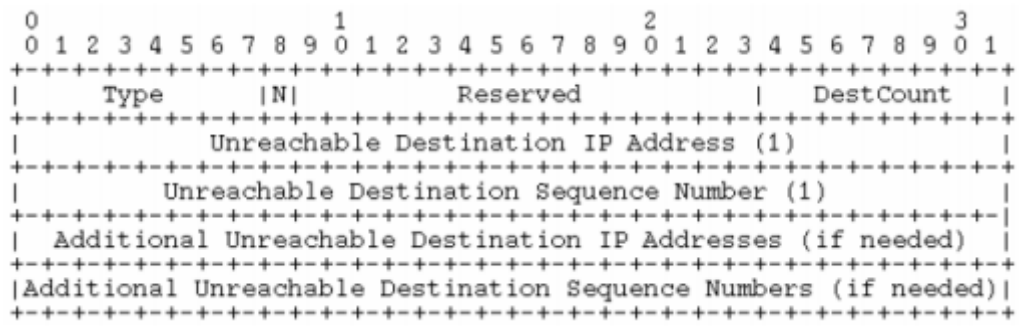


Figura 10 Formato del mensaje Route Error que no presenta campos mutables.

(Monroy, 2012)

1.4.6 Vulnerabilidades

El protocolo AODV no implementa ningún tipo de seguridad a pesar de que en los protocolos de encaminamiento es un objetivo principal, este protocolo está diseñado para confiar en la red y confiar en cada una de las estaciones dentro de ella, ya sea por el empleo de calves pre configuradas o también si se entiende que no existen nodos maliciosos en la red.

Es por este motivo que una red inalámbrica Ad-Hoc es vulnerable a varios tipos de ataques que se detallan a continuación:

- Un nodo malicioso podría usurpar el lugar de un nodo destino falsificando un mensaje tipo RREP incluyendo su dirección como nodo destino
- Un intruso podría hacerse pasar por un nodo destino de la red adulterando un RREP y para crear credibilidad en la red incluiría en el mensaje un numero de secuencia alto para que los vecinos asuman que es una información reciente.
- Si un nodo origen falsificaría un mensaje RERR con un numero de secuencia alto enviando esta información a sus vecinos, estos pensarían que el enlace con dicho nodo ha dejado de operar, entonces inundarían toda la red con esta

información y se volvería imposible volver a buscar rutas hacia este nodo ya que las peticiones tendrían un numero de secuencia menor al que envió el nodo malicioso.

- Para que la red Ad-Hoc funcione correctamente necesita la comunicación continua entre sus estaciones, si estas dejan de reenviar mensajes de control (RREQ, RREP, RRER) podría considerarse un ataque aunque sería complicado detectarlo ya que se puede confundir con errores de transmisión que tienen la misma consecuencia.
- Al momento de que un nodo envía un RRER podría reducir el campo del número de saltos hacia el nodo y aumentar el número de secuencia, esto con el fin de confundir al nodo origen qué pensará que existe una ruta con menor números de saltos y más actual que el resto de rutas que verdaderamente existen.

1.5 PROTOCOLO SAODV

1.5.1 Introducción

En una red Ad-Hoc existen principalmente dos tipos de mensajes, los de enrutamiento y los de datos, ambos tienen una naturaleza diferente por ende necesitan sistemas de seguridad distintos, los mensajes de datos realizan una entrega punto a punto y se puede utilizar cualquier sistema de seguridad de punto a punto, en cambio los mensajes de enrutamiento se envían a varios nodos vecinos de la red y al momento de procesar el mensaje de enrutamiento podría existir modificación en el mismo, por eso existe la necesidad de que los nodos intermedios puedan autentificar la información contenida en los mensajes de enrutamiento.

1.5.2 Funcionamiento

El protocolo SAODV utiliza un mecanismo de administración de claves en donde todos los nodos de la red pueden obtener las claves públicas del resto de nodos, de igual forma cada nodo tiene la capacidad de verificar la identidad de cualquier nodo y la correspondiente clave pública, este mecanismo se lo realiza mediante un algoritmo de administración de claves.

Este protocolo utiliza dos mecanismos de seguridad, las **cadenas hash** para verificar la información que van dentro del único campo modificable dentro del mensaje que es el contador de salto dentro de los mensajes RREQ y RREP, y las **firmas digitales** que caso contrario al mecanismo anterior certifica los campos que no se pueden modificar de los mensajes de encaminamiento, este tipo mecanismos se implementan en los mensajes del protocolo aodv en un campo denominado extensión de firma.

El algoritmo básico consiste en que el creador del mensaje incorpora una firma digital más el componente final de una cadena hash, el mensaje recorre todos los nodos intermedios en donde cada nodo verifica criptográficamente la firma y el valor hash, luego generan el n-simo componente de la cadena hash en donde n es el número total de saltos recorridos y lo insertan en el paquete, los mensajes RREP de respuesta de ruta también son firmados digitalmente.

Existe un mecanismo denominado verificación retrasada de firmas que se utiliza para reducir la sobrecarga en la red debido al uso de criptografía asimétrica, este mecanismo consiste en que cada nodo reenvía los mensajes de encaminamiento antes de examinarlos, para el proceso de descubrimiento de ruta cada nodo solo examinara el mensaje RREQ al recibirlo y luego al enviarlo, esto evita procesamiento ya que únicamente estarán implicados los nodos que están dentro del recorrido del mensaje.

Cadena Hash

Como se mencionó anteriormente, las cadenas hash son las encargadas de verificar la información que van dentro del único campo modificable dentro del mensaje que es el contador de salto de los mensajes RREQ y RREP, esto lo hace con el objetivo de que cada nodo de la red verifique que el contador de saltos no ha sido adulterado por un nodo malicioso.

Una cadena hash está compuesta por una función hash de una dirección que se repite sa un valor aleatorio inicial llamado seed.

Cuando un nodo genera un mensaje RREQ o RREP

En primer lugar crea un numero aleatorio (seed), en el campo Max_Hop_Count introduce el valor del Time To Live de la cabecera Ip , luego en el campo Hash el valor

del número aleatorio, después en el campo Hash Function introduce el identificador de la función hash a usarse, calcula el valor de Top Hash aplicando la siguiente formula

$$Top\_Hash = h^{Max\_Hop\_Count}(seed)$$

en donde h es la función hash y $h^i(x)$ es el desenlace de aplicar la función h a x i veces.

Cuando un nodo recibe un mensaje RREQ o RREP

Aplica la función hash al valor almacenado en el campo HASH (seed) la cantidad de veces especificada por la diferencia entre MAX HOP COUNT Y HOP COUNT como se muestra en la siguiente ecuación

Top_Hash_verificador = Max Hop Count – Hop_Count (campo HASH)

Los campos Hash_Function, Max_Hop_Count, Top_Hash y Hash son firmados con el fin de preservar la integridad, como el campo Hash Function se encuentra firmado entonces un nodo que reenvía una información únicamente lo hará con la misma función que uso el autor del mensaje, si el nodo no puede comprobar un mensaje porque no es compatible con la función hash seleccionada entonces descarta el paquete.

Tabla 4

Posibles valores del campo Hash_Function

Valor	Función Hash
0	Reservado
1	MD5HMAC96[14]
2	SHA1HMAC96[15]
3-127	Reservado
128-255	Dependiente de la implementación

Fuente:(Estudio del rendimiento y la seguridaden redes ad hoc, 2007)

Firmas Digitales

Las firmas digitales verifican los campos que no se pueden modificar de los mensajes de encaminamiento RREQ y RREP, firman todos los campos excepto el campo Hop Count (AODV) y el campo Hash (SAODV).

El proceso de descubrimiento de rutas del protocolo AODV permiten contestar a los nodos intermedios con un RREP si tienen en su tabla de enrutamiento una ruta fresca hacia el destino haciendo eficiente este tipo de protocolo, sin embargo es más difícil implementar seguridad, ya que un nodo intermedio puede almacenar las rutas en el proceso de descubrimiento de reversa después de recibir un RREQ de tal manera no podrá contar con la firma para el RREP.

Para este inconveniente el protocolo ha creado dos tipos de soluciones que se detallan a continuación:

Si un nodo intermedio no puede firmar de manera correcta el mensaje de respuesta de ruta RREP entonces actué como un nodo que no tiene la ruta en su tabla y continúe reenviando el RREQ a sus vecinos, la otra alternativa es que cada que un nodo crea un RREQ introduzca banderas, la firma y el tamaño de prefijo pueden ser usados por cualquier nodo intermedio que origine una ruta reversa al recibir un RREQ y para saber el nodo origen que origino el RREQ.

Por otro lado cuando un nodo intermedio genera un RREP el campo de tiempo de vida va a cambiar, entonces este nodo tendrá que generar el nuevo tiempo de vida e incluir los dos tiempos, el anterior que servirá para comprobar la firma del destino y adicional firmará el nuevo tiempo de vida, con esto se consigue que la información original sea firmada por el destino y el nuevo tiempo de vida por el nodo intermedio.

Cuando un nodo intermedio recibe un RREQ en primer lugar verifica la firma antes de actualizar su tabla de enrutamiento una vez realizado este proceso entonces actualiza su información, si un nodo recibe un RREQ con doble extensión de firma guardara la firma para el RREP y tiempo de vida, si el nodo cumple con los requisitos introduce el tiempo de vida anterior en el campo Signature y Old Lifetime del RREP-DSE , caso contrario sigue reenviando los RREQ a sus vecinos, si el nodo que recibe un RREQ es el destino entonces enviara un RREP-SSE (RREP con extensión de firma única).

Los mensajes de error RRER son protegidos de manera distinta al resto ya que posee algunos campos modificables, cada nodo firma todo el mensaje y sus vecinos verifican dicha firma, sin embargo no pueden firmar el campo números de secuencia del destino e solo lo puede hacer el nodo origen, esto evita que cualquier nodo intruso pueda acceder a información de la red.

1.5.3 Requerimientos de seguridad que satisface

Una característica principal de seguridad en las redes es la autorización que requiere de otros mecanismos de seguridad como la integridad y la autenticación que aplican dentro de sus algoritmos la utilización de las firmas digitales, la confidencialidad y el no repudio debería ser primordial en una red inalámbrica , el no repudio es de gran importancia por ejemplo si un nodo recibe información errónea de otro nodo puede dar aviso a toda la red que dicho nodo no se está comportando adecuadamente.

La disponibilidad no es un aspecto que este protocolo implemente, la denegación de servicios tampoco, aunque en una red inalámbrica no es posible impedir los ataques por denegación de servicios.

A continuación los principales aspectos de seguridad que el protocolo SAODV implementa

Autorización

Esta característica consiste en que cada nodo que envía un mensaje es el único que tiene autorización para solicitar información de su tabla de enrutamiento a otro nodo, de tal forma que si un nodo intruso falsifica su identidad el resto de nodos no le entregara ningún tipo de información.

Autenticación

Cada nodo de la red Ad-Hoc es capaz de comprobar si un nodo es quien dice ser, esta información la puede comprobar mediante un identificador y la firma que posee cada nodo en la red.

Integridad

La integridad consiste en verificar si la información que sale desde un nodo origen es la misma que llega a un nodo destino, es decir comprueba que no exista ningún tipo de alteración en la información.

1.5.4 Simple ad hoc key management (SAKM)

Este sistema es un administrador de claves y su principal función se basa en posibilitar a los nodos para que utilicen un sistema de criptografía asimétrico sin que necesite un sistema central, también protege los campos no modificables de los mensajes de encaminamiento.

En una red no siempre un nodo estará disponible todo el tiempo debido a los continuos cambios de topología de una red Ad-Hoc, es por esto que no pueden ejercer la función de servidores en la red por así decirlo.

En una red Ad-Hoc los nodos no contaran siempre con una dirección IP fija, en algunos casos tomaran una dirección dinámicamente y mediante un broadcast verificar si esta en uso , si no lo está hará uso de dicha dirección caso contrario escogerá otra dirección IP y volverá a enviar un mensaje broadcast hasta encontrar una dirección que no esté en uso, enfocándonos en este tipo de escenarios será difícil para un nodo identificar la dirección IP al nodo que va transmitir ya que no existen servidores como en las redes convencionales (DHCP), es por eso que se ha creado este sistema en donde se busca que la dirección pública dinámica asignada se asocie con una clave pública y se lo realice sin ningún clase de autoridad de certificación.

Este tipo de sistema (SAKM) genera una dirección generada criptográficamente (CGA) que será una dirección única , un nodo origen utiliza este tipo de direcciones y adicional inserta una clave pública firmada por su clave privada, así cualquier nodo de la red podrá comprobar la identidad mediante la información de su clave secreta.

1.5.5 Generación de direcciones IP para SAODV

El protocolo SAODV sugiere utilizar direcciones Ipv7 en vez de direcciones Ipv4 debido a su mayor longitud de dirección que garantiza la singularidad de la dirección Ip, la dirección se puede componer de un prefijo de red de 64 bits con un Identificador medio (SAODV HID) y 128 bits con un identificador full (SAODV FID).

Para la generación de direcciones se utiliza una bandera (H) en las extensiones de los mensajes de enrutamiento, esta bandera se colocara en `1` si la dirección IP corresponde a una tipo HID y en `0` si pertenece a una FID

La longitud de una dirección Ipv4 es muy corta para proporcionar una singularidad, para que exista distinción en este tipo de sistema requiere que el número de nodos sea elevado, caso contrario si existe un bajo número de nodos no es acertado esperar que la propiedad de singularidad se cumpla.

La dirección Ipv4 de SAODV contiene un prefijo de red de 8 bits (SAODV ID), el prefijo puede ser un número entre el rango 1 y 126 excepto 14,24 y 93, adicional el prefijo 10 se utiliza únicamente si la subred no será conectada a otra red.

Detección de Direcciones Duplicadas

Saodv puede resolver el inconveniente de utilizar direcciones duplicadas en las estaciones de la red, cuando una dirección que quiere usar un nodo X ya se encuentra en uso se envía un mensaje al nodo de Dirección Duplicada (DADD), adicional se envía un mensaje de Nueva Dirección (NADD) que indica que el nodo tendrá que generar nuevamente un par de claves (pública y privada) luego obtendrá su nueva dirección IP a partir de su clave pública e informara al resto de nodos mediante un mensaje particular indicando su anterior y nueva dirección IP y las dos firmas publicas la anterior y la nueva adicional su clave privada, este mensaje se lo enviara de manera unicast a los nodos que considere deban obtener esta aclaración , para finalizar se envía un mensaje de Reconocimiento de Dirección (Nadd-ACK) para comprobar que ha recibido el mensaje NAAD .

Para actualizar las tablas de enrutamiento del resto de nodos de la red , el nodo X generara un mensaje RERR con su dirección IP anterior y una extensión que indique la nueva dirección , con esto los nodos eliminaran las direcciones duplicadas y

almacenaran las nueva ruta con la nueva dirección .

Líderes de Red

El planteamiento inicial con el que se creó SAODV estableció que además de crear el sistema de distribución de claves que servirá para comprobar la identidad de un nodo también se lo puede identificar mediante la máscara de subred, esto servirá para identificar a un nodo intruso que quiere hacerse pasar por un nodo líder de la red.

En general un nodo de una red Ad-Hoc no es líder de una red, los líderes únicamente serán nodos fijos que brindan acceso a una red fija y el resto de estaciones de la red debe conocer su dirección IP , el tamaño del prefijo y sus correspondientes claves públicas .

Los nodos líderes de las redes no cambiarán su dirección IP cuando otro nodo quiera utilizar su dirección, un nodo en primer lugar al generar una dirección comprueba si pertenece a un nodo líder o a una subred correspondiente al tamaño de su prefijo.

Si un nodo descubre que otro nodo está usando o quiere generar la dirección Ip del líder de la red o cualquiera de estos correspondiente a la subred del líder, entonces informara al resto de nodos pero no al líder de la red.

1.5.6 Verificación de firmas

Como se indicó anteriormente la verificación de firmas puede ocasionar un poco de sobrecarga a la red, es por eso que SAODV implementa un método denominado verificación retrasada de firmas para solucionar este inconveniente y brindar un aspecto de seguridad muy importante a la red.

Exigencias de Seguridad

Las exigencias de seguridad que proporcionan la verificación retrasada de firma son la integridad, la autenticación de fuente y la autorización de importación retrasada.

La autorización de importación consiste en que cada nodo que envía un mensaje es

el único que tiene autorización para solicitar información de su tabla de enrutamiento a otro nodo, de tal forma que si un nodo intruso falsifica su identidad el resto de nodos no le entregara ningún tipo de información

La autorización de importación retrasada permite disponer de entradas de ruta en la tabla de enrutamiento de un nodo que se encuentran pendientes de verificar, la verificación se la realizara siempre y cuando una estación haya ahorrado un poco de energía para procesar o antes de que las entradas sean usadas para enviar paquetes de datos.

Este tipo de característica no incluirán aspectos como la confidencialidad ni el no repudio ya que no se las considera servicios críticos en el aspecto de encaminar, tampoco se incluirá la disponibilidad ya que un nodo malicioso puede enfocarse en la capa física sin analizar previamente el protocolo de encaminamiento, el problema de nodos comprometidos porque no se observa que es una exigencia de seguridad en ámbitos que no sean militares.

Alcanzar Autorización de Importación Retrasada

En los protocolos de encaminamiento reactivos la mayor parte de mensajes de enrutamiento que circula en la red son solicitudes de ruta, esto se debe a que las rutas solicitadas se envían a toda la red (broadcast) y las respuesta de ruta se envían de forma unicast de forma reversa al camino seleccionado, los mensajes de error de ruta también se envían de forma unicast a todos los nodos que tienen en su tabla una ruta del nodo inalcanzable.

Cuando un nodo recibe un mensaje de solicitud de ruta crea una nueva entrada en su tabla de enrutamiento llamada ruta inversa, por lo tanto después de emitir un mensaje de solicitud de ruta todos los nodos de la red crean supuestas rutas inversas hacia al autor de la petición de ruta, pero la mayoría de estas rutas expiraran pronto ya que solo se mantendrá la ruta seleccionada a través de la cual viajara la respuesta de ruta.

Seria innecesario realizar la verificación de firmas de estos mensajes de rutas cuando se sabe que la mayoría van a expirar, entonces no hay necesidad de verificarlos hasta

que la respuesta de ruta correspondiente regrese y el nodo sepa que es la ruta seleccionada, el resto de rutas inversas expiraran sin ser verificadas, con el mismo planteamiento los mensajes de error de ruta y en general cualquier mensaje de encaminamiento se puede verificar después de haber sido enviado.

Las rutas que estén pendientes de verificación no se usaran para el envió de paquetes, si un paquete llega a un nodo en el cual existe una ruta pendiente de verificación entonces el nodo tendrá que verificar la ruta antes de utilizarla, si la verificación falla el nodo eliminará la ruta y solicitará una nueva.

SAODV con verificación retrasada

Cuando un nodo requiere enviar o reenviar un paquete a un destino para el cual no tiene una ruta activa, empezará comprobando si tiene una ruta pendiente de verificación, si la tiene la verificará y si lo hace con éxito se establece la ruta como activa y se la podrá utilizar, si no existe una ruta activa el nodo empezara el proceso de descubrimiento de ruta.

En la figura 10 se muestra que una vez que la verificación se realiza la ruta es incorporada en la tabla de enrutamiento del nodo, los paquetes pueden ser enviados con normalidad, y solo cuando existe una consulta en la tabla de enrutamiento y esta no la puede resolver, la petición es capturada por el demonio de encaminamiento de SAODV

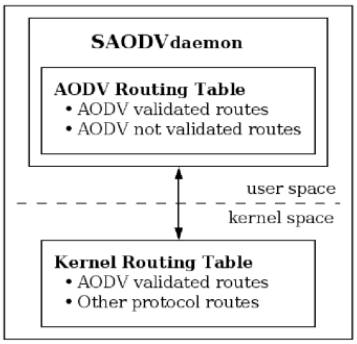


Figura 11 Demonio SAODV

(Zapata, 2006)

1.5.7 Formato de mensajes SAODV

RREQ (Single) Signature Extension

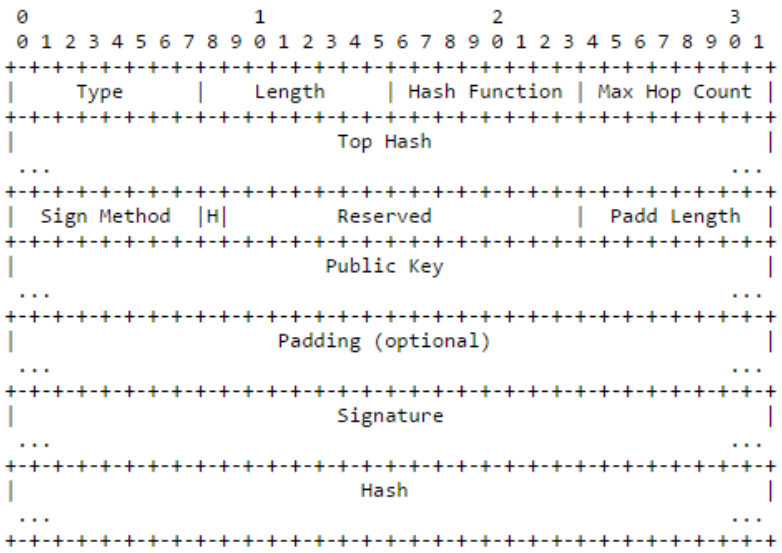


Figura 12 RREQ (Single) Signature Extension

(Zapata, 2006)

A continuación se detalla la función de cada campo:

- **Length**: indica la longitud del tipo específico de datos, no incluye los campos tipo y longitud en bytes
- **Hash Function**: es utilizado para calcular los campos Hash y Top Hash
- **Max Hop Count**: el número máximo de saltos es apoyado para la autenticación del número de saltos
- **Top Hash**: este campo tiene longitud variable pero debe ser de 32 bits alineados
- **Signature Method**: es utilizado para indicar el método de firmas que se utilizará en la red
- **H**: si se pone a '1' indica el uso de HID y si se ajusta a "0" el uso de la FID, se utiliza para la generación de IP's en SAODV
- **Reserved:** es enviado en '0' e ignorado en la recepción
- **Padding Length:** especifica la longitud del campo de relleno en unidades de 32 bits, si el campo se encuentra en '0' no habrá relleno

- **Public Key:** es la clave pública del remitente del mensaje. Este campo tiene longitud variable, pero debe ser 32-bits alineado
- **Padding:** El tamaño de este campo se establece en el campo Longitud de Relleno**.**
- **Signature:** La firma de todos los campos de los paquetes AODV irán antes de este campo **,** este campo tiene la longitud variable, pero debe ser de 32 bits alineado
- **Hash:** corresponde a la cuenta de salto real, este campo tiene la longitud variable, pero debe ser de 32 bits alineado

RREP (Single) Signature Extension

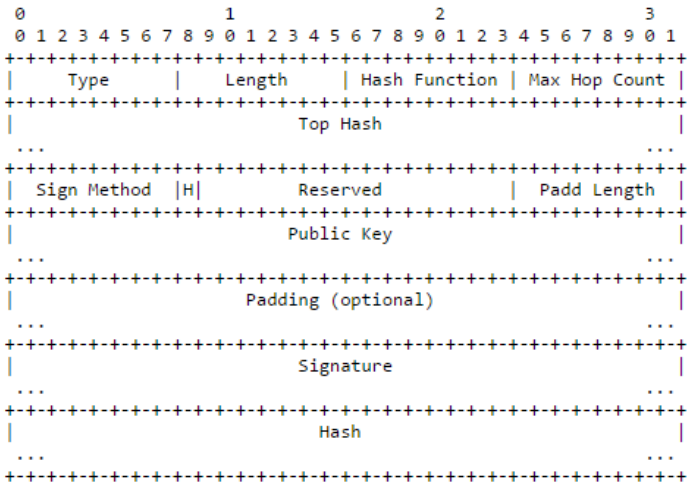


Figura 13 RREP (Single) Signature Extensión

(Zapata, 2006)

- **Length**: indica la longitud del tipo específico de datos, no incluye los campos tipo y longitud en bytes
- **Hash Function**: es utilizado para calcular los campos Hash y Top Hash
- **Max Hop Count**: el número máximo de saltos es apoyado para la autenticación del número de saltos
- **Top Hash**: este campo tiene longitud variable pero debe ser de 32 bits alineados

- **Signature Method**: en este campo se coloca el mismo valor que en el mensaje RREQ

- **Signature:** La firma de todos los campos de los paquetes AODV irán antes de este campo **,** este campo tiene la longitud variable, pero debe ser de 32 bits alineado
- **Hash:** corresponde a la cuenta de salto real, este campo tiene la longitud variable, pero debe ser de 32 bits alineado

RREQ Double Signature Extension

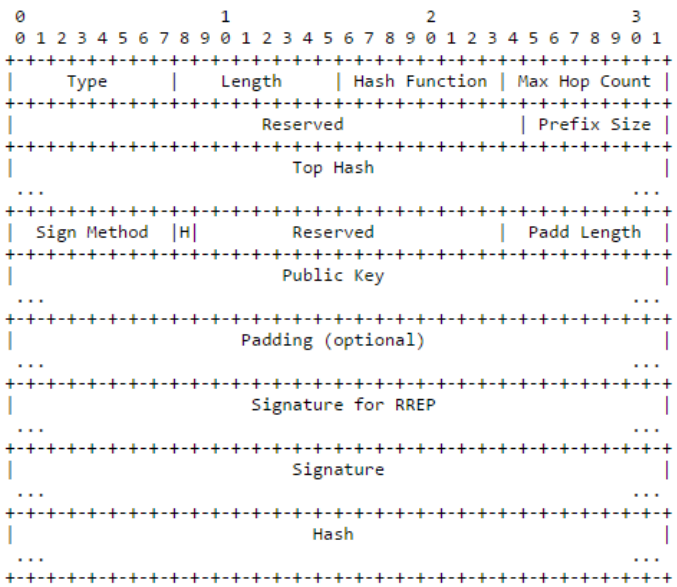


Figura 14 RREQ Double Signature Extensión

(Zapata, 2006)

- **Length**: indica la longitud del tipo específico de datos, no incluye los campos tipo y longitud en bytes

- **Hash Function**: es utilizado para calcular los campos Hash y Top Hash
- **Max Hop Count**: el número máximo de saltos es apoyado para la autenticación del número de saltos
- **Reserved:** es enviado en '0' e ignorado en la recepción
- **Prefix Size:** el tamaño de prefijo para el RREP es de 7 bits para permitir prefijos Ipv6
- **Top Hash**: este campo tiene longitud variable pero debe ser de 32 bits alineados
- **Signature Method**: en este campo se coloca el mismo valor que en el mensaje RREQ
- **Signature for RREP:** en este campo se encuentra la firma de un nodo que ha recibido un RREQ y ha creado una ruta inversa generando un RREP hacia la fuente que generó el mensaje de requerimiento de ruta RREQ.

- **Signature:** La firma de todos los campos de los paquetes AODV irán antes de este campo**,** este campo tiene la longitud variable, pero debe ser de 32 bits alineado, ambas firmas son generadas por el nodo solicitante.
- **Hash:** corresponde a la cuenta de salto real, este campo tiene la longitud variable, pero debe ser de 32 bits alineado

RREP Double Signature Extension

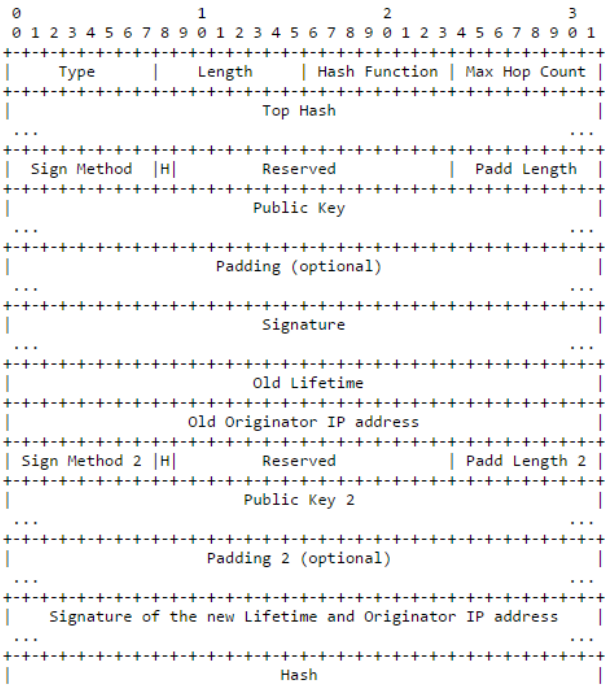


Figura 15 RREP Double Signature Extension

(Zapata, 2006)

- **Length**: indica la longitud del tipo específico de datos, no incluye los campos tipo y longitud en bytes
- **Hash Function**: es utilizado para calcular los campos Hash y Top Hash
- **Max Hop Count**: el número máximo de saltos es apoyado para la autenticación del número de saltos
- **Top Hash**: este campo tiene longitud variable pero debe ser de 32 bits alineados
- **Signature Method**: en este campo se coloca el mismo valor que en el mensaje RREQ
- **Signature:** La firma de todos los campos del paquete AODV son antes de este campo, y con el antiguo valor del tiempo de vida , esta firma es generada por el mensaje RREQ-DSE, este campo tiene longitud variable pero debe ser de 32 bits alineados
- **Old Lifetime:** El tiempo de vida que estaba en el RREP generada por el originador del RREQ-DSE)
- **Old originator IP Address:** el creador de la dirección IP que estaba

en el RREP generada por el creador del RREQ-DSE)

- **Signature Method 2:** todo los campos del bloque se repiten, esta vez para el campo “Signature of the New Lifetime and Originator IP address”
- **Signature of the New Lifetime and Originator IP address:** es la firma del RREP en tiempo real (el tiempo de vida de la ruta en el nodo intermedio) y con la dirección de IP del creador real. Esta firma es generada por el nodo intermedio. Este campo tiene longitud variable, pero debe ser 32-bits alineados.
- **Hash:** corresponde a la cuenta de salto real, este campo tiene la longitud variable, pero debe ser de 32 bits alineado

RERR Signature Extension

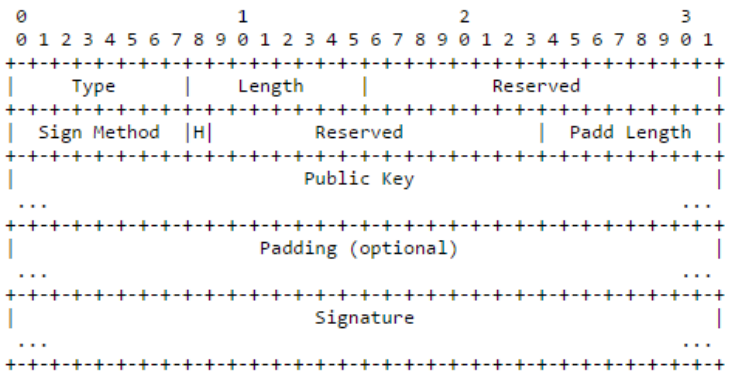


Figura 16 RERR Signature Extension

(Zapata, 2006)

- **Length**: indica la longitud del tipo específico de datos, no incluye los campos tipo y longitud en bytes
- **Reserved:** es enviado en ‘0’ e ignorado en la recepción
- **Signature Method**: en este campo se coloca el mismo valor que en el mensaje RREQ (Single)
- **Signature:** La firma de todos los campos de los paquetes AODV irán antes de este campo**,** este campo tiene la longitud variable, pero debe ser de 32 bits alineado.

RREP-ACK Signature Extension

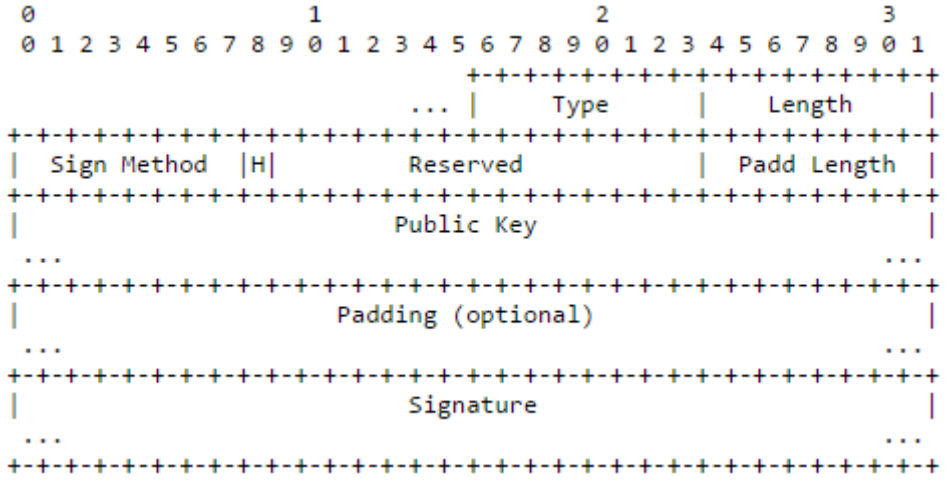


Figura 17 RREP-ACK Signature Extension

(Zapata, 2006)

- **Length**: indica la longitud del tipo específico de datos, no incluye los campos tipo y longitud en bytes
- **Reserved:** es enviado en '0' e ignorado en la recepción
- **Signature Method**: en este campo se coloca el mismo valor que en el mensaje RREQ (Single)
- **Signature:** La firma de todos los campos de los paquetes AODV irán antes de este campo**,** este campo tiene la longitud variable, pero debe ser de 32 bits alineado

CAPÍTULO 2

2.1 Simulador

2.1.1 Introducción

A través de los años ha surgido la necesidad de encontrar una manera de poder visualizar como podría ser el funcionamiento de varios protocolos implementados en una red y que se encuentran en etapa de investigación, NS2 es un herramienta de simulación poderosa que nos brinda dicha posibilidad además de ser un software con gran nivel de detalle, permite visualizar el funcionamiento de los protocolos de una manera profunda.

Entre varias de sus características se puede visualizar de manera gráfica las simulaciones en tiempo real visualizando el intercambio de paquetes entre los nodos y así comprender más a fondo el funcionamiento de cada protocolo, la idea de implementar este software de simulación comienza en los años 80 a partir del Real network simulator, años después fue complementado mediante ARPA y actualmente se encuentra a manos de grupos de investigación de Universidades de EEUU, la versión actual del simulador es la 2.35 que utilizaremos para el desarrollo de este proyecto.

Esta poderosa herramienta nos ayudará a simular diferentes tipos de redes ya sean terrestres, inalámbricas por satélite, cada uno con algunos protocolos de encaminamiento que vienen implementados en el software y otros que con la ayuda de programación y algoritmos de enrutamiento y seguridad se puede implementar por cuenta propia como en este proyecto.

2.1.2 Instalación de NS

Esta herramienta de simulación puede ser instalada en cualquier tipo de PC que tenga mínimo los siguientes requerimientos.

- 128 Mb de memoria RAM

- 4GB de disco duro
- Procesador Pentium (última generación)

Estos requisitos van a variar depende de la versión que se va a instalar, los protocolos que vamos a simular y los escenarios de simulación ya que algunas simulaciones ocupan gran capacidad de procesamiento y memoria.

Para nuestro proyecto usaremos la versión de Ubuntu 12.04 , también se puede realizar la instalación en el entorno de Windows con la herramienta cygwin, pero el simulador fue creado propiamente para trabajar en Linux.

A continuación los pasos para la instalación del simulador NS2 version 2.35 en nuestra PC.

- En primer lugar se dirige a la página de descarga http://sourceforge.net/projects/nsnam/ y se elige el archivo ns-allinone-2.35.tar.gz

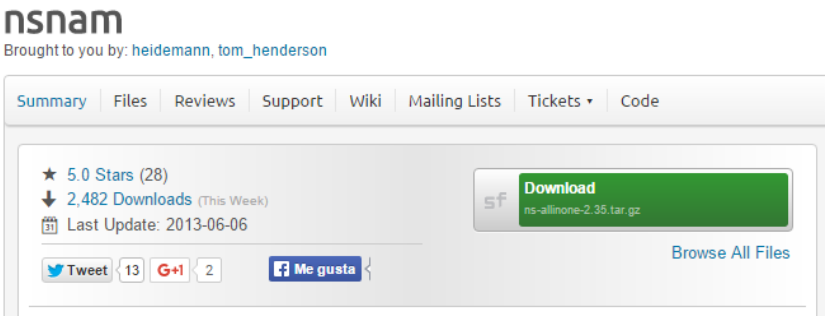


Figura 18 Página de descarga del NS2

- Una vez descargado el archivo procedemos a extraerlo en cualquier directorio de nuestro sistema operativo en nuestro caso lo hicimos en la carpeta “Home”

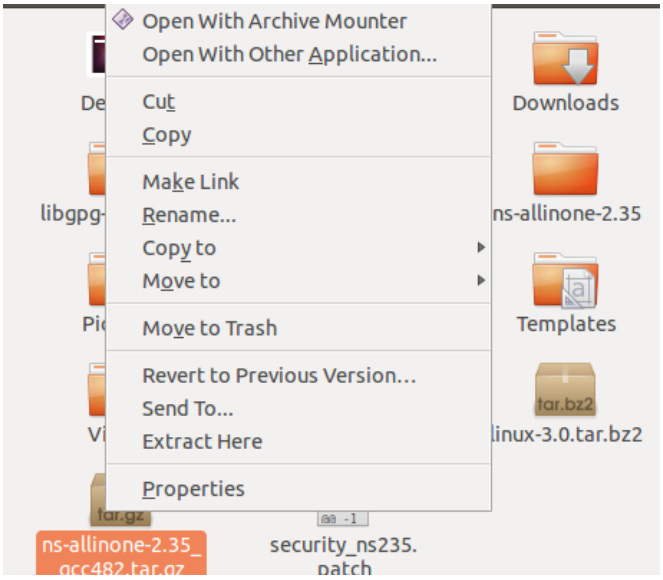


Figura 19 Directorio para extraer el archivo descargado

- A continuación se abre el terminal de Ubuntu y se ejecutan los siguientes comandos que sirven para instalar todos los paquetes básicos que requiere el NS2

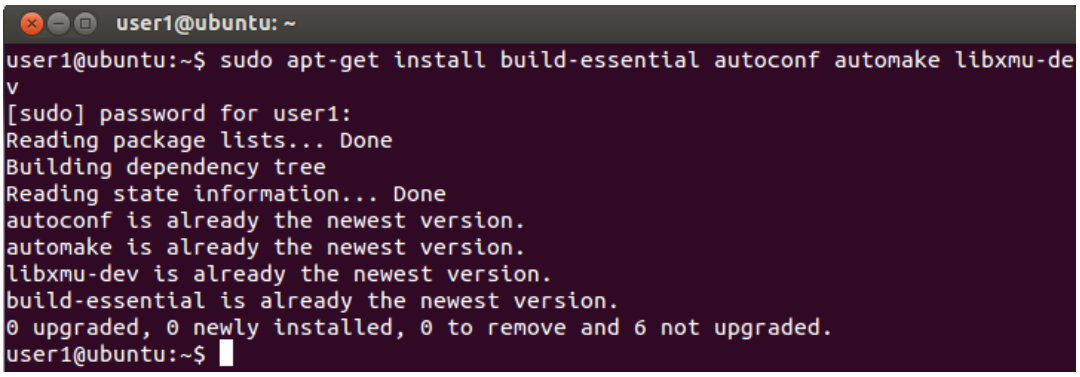


Figura 20 Paquetes básicos del NS2

- Una vez instalado todos los paquetes necesarios se procede a modificar la línea 137 del archivo ns-allinone-2.35/ns-2.35/linkstate/ls.h file as de la siguiente manera

```
        void eraseAll() {this> baseMap::erase(baseMap::begin(), baseMap::end
()); }
        T* findPtr(Key key) {
                iterator it = baseMap::find(key);
                return (it == baseMap::end()) ? (T *)NULL : &((*it).second);
        }
```

Figura 20 Archivo ls.h

- Se guardan los cambios y se abre nuevamente un terminal y nos dirigimos a la carpeta en donde se encuentra nuestro programa y se ejecutan el comando ./install para empezar la instalación .

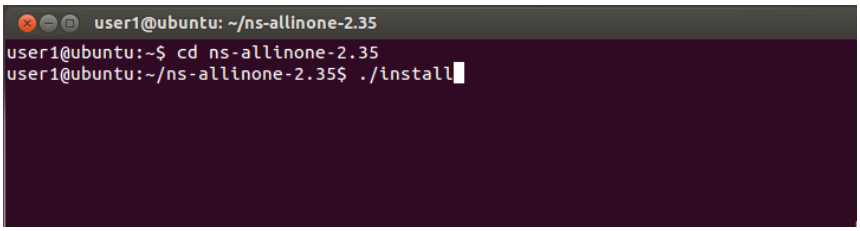


Figura 21 Comando para empezar la instalación

- Una vez finalizada la instalación se mostrará el siguiente mensaje

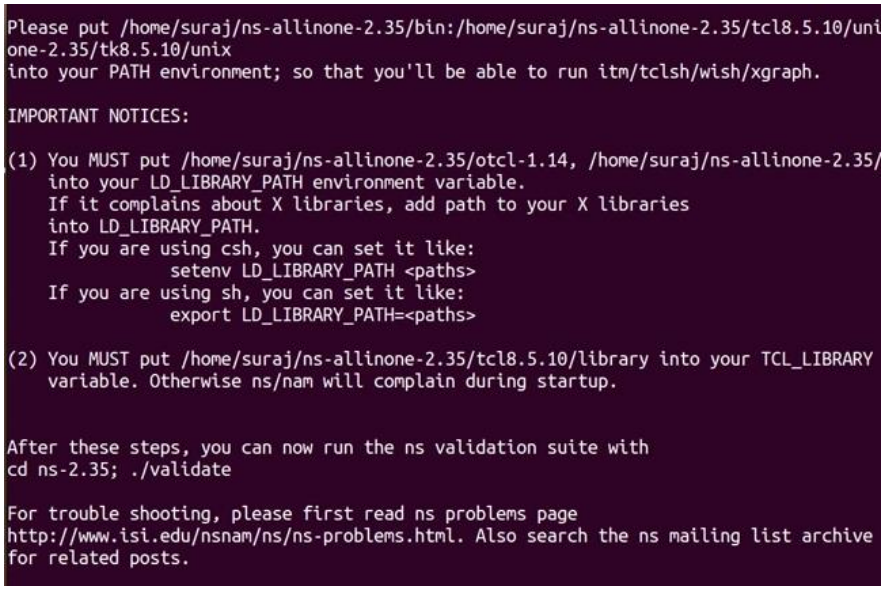


Figura 22 Mensaje al culminar la instalación

- Al finalizar la instalación insertamos el comando $gedit ~/.bashrc que generará crear un archivo path.

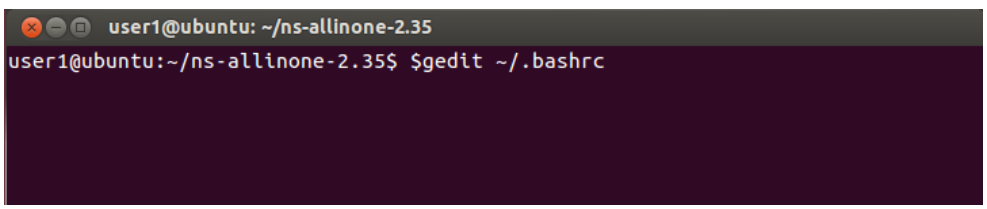


Figura 23 Comando para editar el archivo bashrc

Una vez que se genera el archivo path se modificarán los siguientes directorios como se muestra a continuación

```
 # LD_LIBRARY_PATH
OTCL_LIB=/home/user1/ns-allinone-2.35/otcl-1.14
NS2_LIB=/home/user1/ns-allinone-2.35/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB:$USR_LOCAL_LIB
# TCL_LIBRARY
TCL_LIB=/home/user1/ns-allinone-2.35/tcl8.5.10/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB
# PATH
XGRAPH=/home/user1/ns-allinone-2.35/bin:/home/user1/ns-allinone-2.35/tcl8.5.10/
unix:/home/user1/ns-allinone-2.35/tk8.5.10/unix
NS=/home/user1/ns-allinone-2.35/ns-2.35/
NAM=/home/user1/ns-allinone-2.35/nam-1.15/
export PATH=$PATH:$XGRAPH:$NS:$NAM
```

Figura 24 Archivo bashrc

- Se guardan los cambios y mediante el terminal se ejecuta el comando $source ~/.bashrc

Figura 25 Comando para ejecutar el archivo bashrc

- Para finalizar ejecutamos en el terminal el comando ns , se muestra un signo de % que nos indica que nuestro simulador ha sido instalado correctamente

Figura 26 Instalación exitosa del NS2

2.1.3 El lenguaje TCL

El simulador NS2 utiliza 2 lenguajes de programación, el lenguaje c y el tcl.

Su principal función es simular programas con sintaxis simple, de fácil aprendizaje, se emplea para realizar algunos scripts de interfaces gráficas, es un lenguaje

multiplataforma sin embargo al momento de compilar tardará un poco más que otros lenguajes ya que este es un lenguaje interpretado mas no compilado.

Para ejecutar un entorno de simulación en NS2 en primer lugar se elabora un archivo en lenguaje tcl en donde se configura el entorno de simulación como el número de nodos, la posición , el movimiento , el protocolo que se usará , los parámetros del protocolo , los agentes de tráfico, la topología , en si como se debe comportar la red , lo que viene a continuación es un trabajo interno que realiza el NS2, toda la simulación se resume en un archivo de traza (.trace) que es bastante dificultoso de leer, pero usando algunas aplicaciones especiales podemos mostrar los resultados deseados de manera gráfica, adicional también se puede generar un archivo (.nam) en donde se puede visualizar gráficamente la simulación en tiempo real.

2.1.4 Funcionamiento interno de NS

El simulador NS2 fue creado para simular redes, en el escenario de simulación se toman varios factores en cuenta como la transmisión de paquetes, la topología y el software nos mostrara el comportamiento de la red, con todas sus especificaciones.

A continuación se muestra una forma sencilla del funcionamiento de NS2 indicando las partes más comunes dentro de su descripción interna

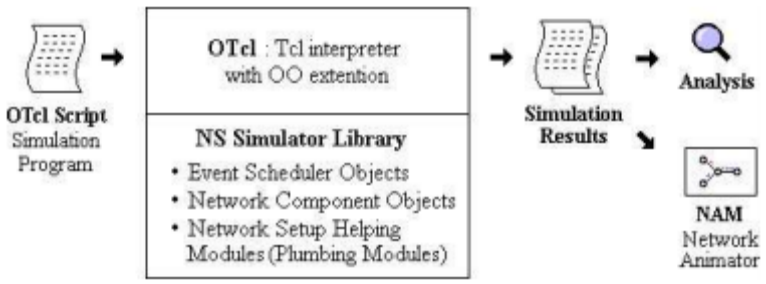


Figura 27 Funcionamiento interno del NS2

En la figura 27 se muestra que se empieza con un script escrito en lenguaje tcl, luego de eso viene el trabajo interno que ejecuta el NS2 en donde se encuentran las librerías, los módulos, la interpretación entre los dos lenguajes de programación (C y tcl) ,los paquetes , los protocolos de enrutamiento, entre otros.

A continuación se detalla rápidamente la función de cada componente para un mayor entendimiento

Programador de eventos de objetos

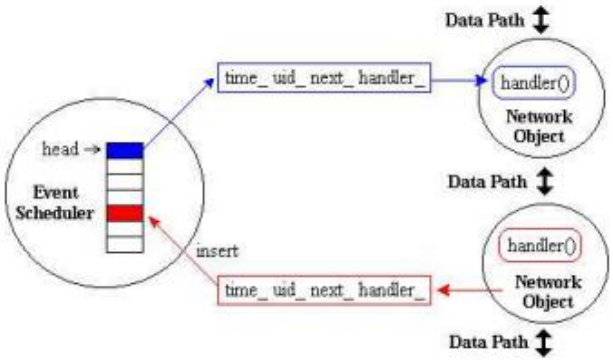


Figura 28 Planificador de eventos

La transmisión de paquetes entre nodos requiere de cierto retardo, si un usuario configura un tiempo demasiado alto de retardo o un bajo ancho de banda el planificador de eventos tiene que ver la manera de configurar la red para que la transmisión sea pausada, todos los objetos cuentan con un planificador de eventos para ajustarse a los parámetros que el usuario requiera.

Objeto de componente de red

Este objeto es el encargado de realizar una comunicación solida entre los componentes de la red, ejemplos de objetos de res son ancho de banda, retardo, entre otros.

A manera de ejemplo se muestra una red con dos nodos que cuentan con un enlace unidireccional, el objeto de componente de red debe calcular el tamaño del paquete en el caso de que exista una cola de espera, dependiendo de las circunstancias se decidirá si el paquete es descartado o se realizara un retardo y el paquete tendrá que esperar para ser transmitido.

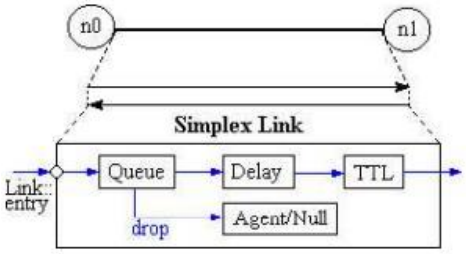


Figura 29 Componente de red

Módulo de ayuda de configuración de la red

El módulo de ayuda es el encargado de seleccionar las bibliotecas que se necesitan para ejecutar una simulación, como se sabe NS2 está escrito y compilado lenguaje C y se asocia a un intérprete Tcl a través de un linkage (vinculo) como se muestra en la figura a continuación

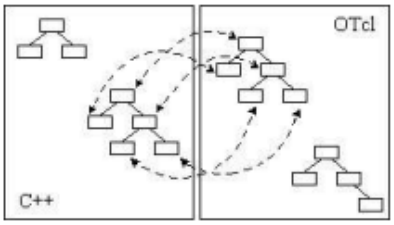


Figura 30 Enlace entre bibliotecas

2.1.5 Herramientas de diseño y análisis

Generador de topología de red

Para simular un evento se requiere generar una topología, el network simulator 2 puede generar topologías utilizando los siguientes métodos:

- Utilizando el generador de topología Inet.
- Utilizando el generador de topología GT-ITM.
- Utilizando el generador de topología Tiers.
- Mediante el uso de los otros generadores de topología.

Gt-itm

El generador de topología GT-ITM es un software adicional que implementa NS2 , sirve para la creación de gráficos aleatorios, además de trazar gráficos jerárquicos de nivel N y transit-stub .

Para su utilización comprobamos su correcta ejecución mediante el directorio inicial de NS2 , para ejecutarlo se sigue los siguientes pasos :

- Crear un archivo de especificación para GT-ITM
- Generar el archivo de topología
- Convertir el formato a NS2
- Acomodar el resultado

Brite

Otra de los generadores de topología en NS2 es BRITE, este software permite dos lenguajes de programación c++ y java, como siempre en el entorno con java se aprecia varias interfaces graficas familiarizadas con el usuario.

Adicional BRITE utiliza diversos algoritmos para conexión de borde y para la interconexión a nivel de enrutamiento, la ejecución de dicha interconexión todavía se encuentra en investigación y es un campo de estudio abierto.

Network Animator (NAM)

NAM es una herramienta fundamental que utiliza NS2 ya que nos permite visualizar la simulación en tiempo real pudiendo observar los nodos de la red, la posición , el movimiento, el flujo de tráfico en la red, esto con el fin de mejorar la comprensión del funcionamiento de las redes y los protocolos de enrutamiento.

Esta herramienta de animación esta cimentada en tcl, para obtener un archivo .nam en primer lugar se genera un archivo de rastreo en el script tcl, este archivo debe contener la información de la red como la topología, nodos, enlaces colas, entre otros parámetros.

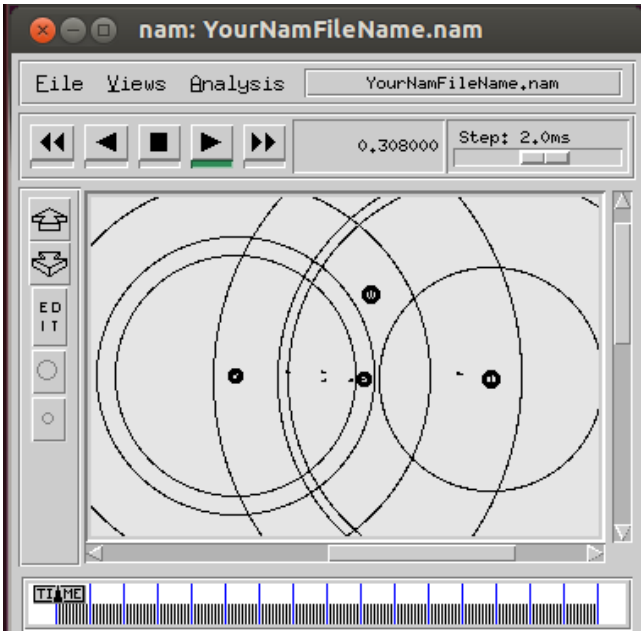


Figura 31 Herramienta NAM

Además de generar una interfaz gráfica la herramienta NAM también nos proporciona un archivo de traza similar al archivo trace pero mucho más sencillo en donde se puede interpretar de una manera rápida el funcionamiento de la red, a continuación se detalla algunos parámetros del archivo (.nam) que nos ayudará a interpretar lo que sucede en la red.

```
h -t 0.000000000 -s 3 -d -1 -p tcp -e 40 -c 2 -a 0 -i 0 -k AGT
+ -t 0.000000000 -s 3 -d -1 -p AODV -e 48 -c 2 -a 0 -i 0 -k RTR
- -t 0.000000000 -s 3 -d -1 -p AODV -e 48 -c 2 -a 0 -i 0 -k RTR
h -t 0.000000000 -s 3 -d -1 -p AODV -e 48 -c 2 -a 0 -i 0 -k RTR
+ -t 0.000115000 -s 3 -d -1 -p AODV -e 106 -c 2 -a 0 -i 0 -k MAC
- -t 0.000115000 -s 3 -d -1 -p AODV -e 106 -c 2 -a 0 -i 0 -k MAC
h -t 0.000115000 -s 3 -d -1 -p AODV -e 106 -c 2 -a 0 -i 0 -k MAC
r -t 0.000963367 -s 2 -d -1 -p AODV -e 48 -c 2 -a 0 -i 0 -k MAC
r -t 0.000963458 -s 0 -d -1 -p AODV -e 48 -c 2 -a 0 -i 0 -k MAC
```

Figura 32 Archivo .nam

: El paquete empezó a transmitirse desde el origen hacia el destino

- **r**: El paquete terminó la transmisión y empezó a ser recibido por el destino
- **d**: El paquete fue eliminado de la cola o desde el origen al destino
- **+**: El paquete entró a la cola desde el origen hacia el destino
- **- :** El paquete salió de la cola desde el origen al destino
- **-t:** Momento en el que sucedió el evento

- **-s:** Nodo origen
- **-d:** Nodo destino
- **-p:** Nombre descriptivo del tipo de paquete
- **-e:** Tamaño en bytes del paquete
- **-c:** Identificador de flujo
- **-a:** Identificador de paquete

Awk

Otra herramienta muy importante en NS2 son los scripts AWK que nos permiten procesar la información a través de los archivos de traza , en otra palabras nos indican parámetros difíciles de calcular como el delay, jitter, throughput , entrega de paquetes para una mejor comprensión del funcionamiento de la red .

Es importante que el usuario comprenda los conceptos básicos de los scripts awk a continuación se muestra los comandos que se deben ingresar en el terminal para ejecutar los scripts awk

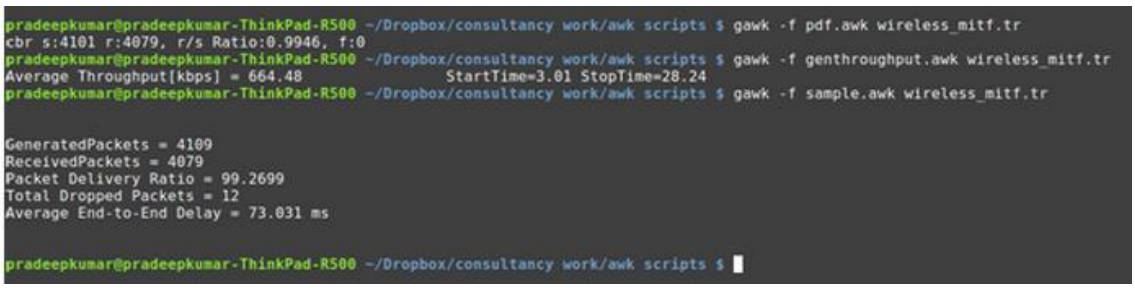


Figura 33 Herramienta Awk

El primer comando nos entregara el rendimiento de la red, el segundo comando nos indica el throughput promedio generado en la red y el tercer comando nos indica la tasa de paquetes generados, recibidos, perdidos y un promedio del retardo de la red (delay).

Xgraph

La herramienta xgraph que viene en NS2 nos sirve para interpretar los archivos de trazado y mostrarlos de forma gráfica, dentro de los scripts tcl se debe agregar algunos

comandos para realizar la interpretación del xgraph , este archivo se encarga de indicar información de la red como sobrecarga, delay entre otros además de incorporar una funcionalidad que permite comparar protocolos

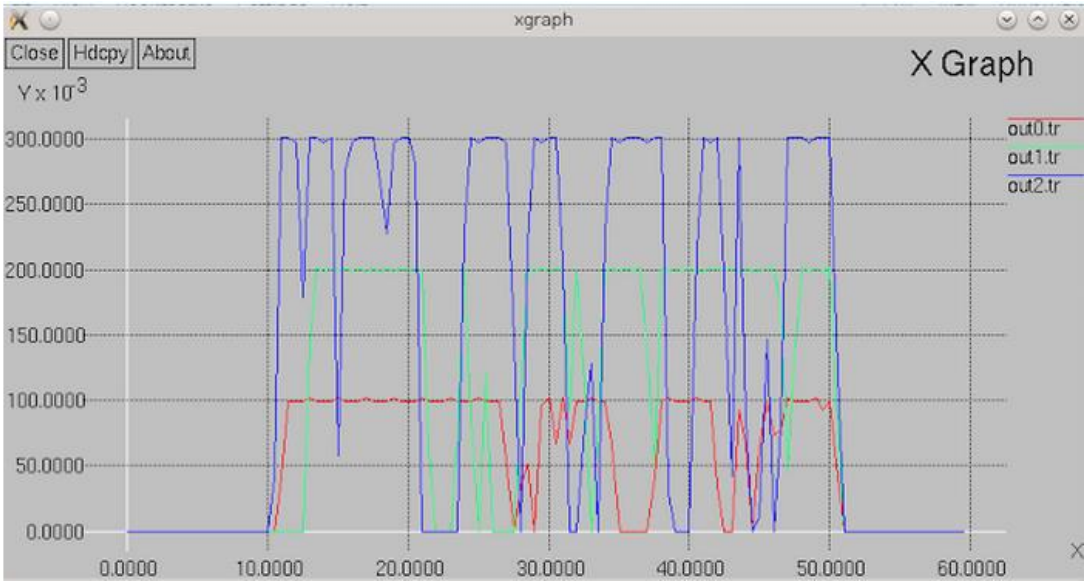


Figura 34 Herramienta Xgraph

La interfaz utilizada para determinar el tamaño de la ventana depende netamente del gestor de ventanas, cuando se ejecuta la herramienta se despliega una ventana con todos los conjuntos de datos mostrados de manera gráfica con los detalles en la esquina superior derecha, adicional presenta 3 botones en la parte superior de cierre del programa y de ayuda sobre el software.

Gnuplot

El GNUplot es una herramienta similar al Xgraph, es un programa de codirijo abierto que muestra los resultados de la simulación de manera gráfica, existen ejecutables pre-compilados disponibles para Windows, Os x, Linux.

Splot y trama son los principales comandos en gnuplot , estos se encargan de recopilar las funciones necesarias que ayudaran en el trazado del grafico del funcionamiento de la red .

Los datos discretos de un archivo se pueden mostrar especificando entre comillas el nombre del archivo , los archivos deberán tener los datos organizados en columnas de números , las columnas separadas por espacios, un solo espacio en blanco podrían

entregarnos resultados erróneos por lo que se debe tomar en cuenta esta recomendación al momento de utilizar esta herramienta .

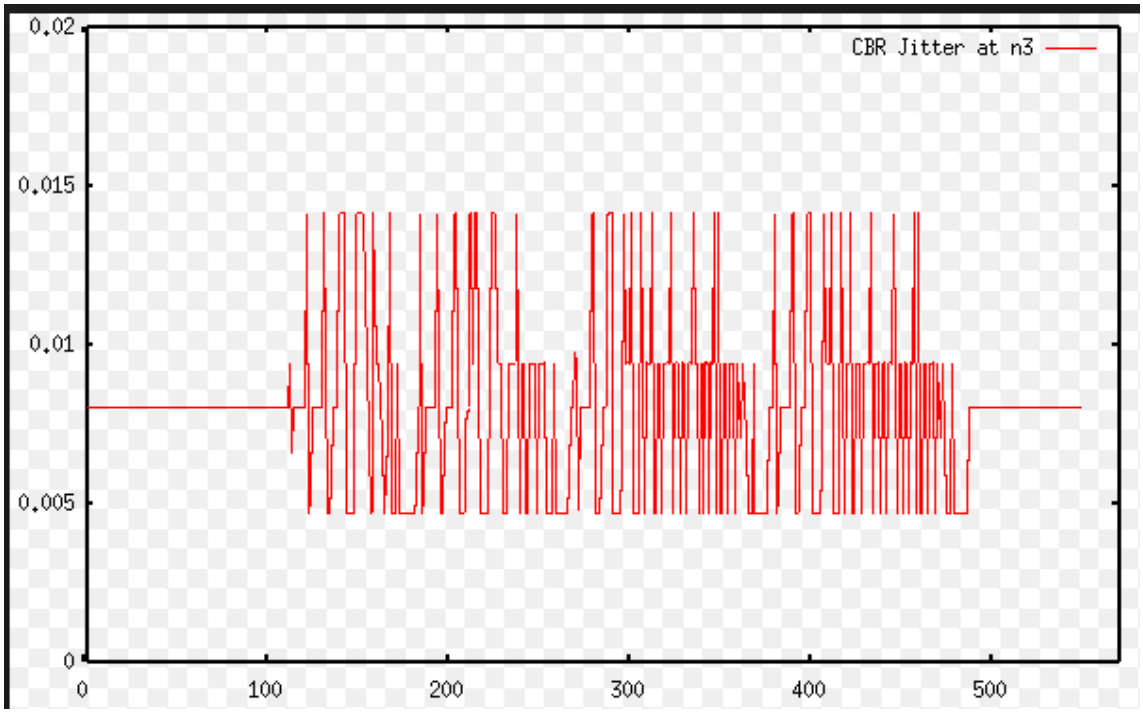


Figura 35 Herramienta Gnuplot

2.1.6 Programación de un nuevo agente

Agente Tcp

Para simular el funcionamiento y comportamiento de una red debemos inyectar cierto tipo de tráfico entre los nodos, es aquí donde entra la programación de los agentes, TCP y UDP son los agentes más comunes en NS2 que nos permiten realizar la entrega de paquetes entre nodos, un agente según el manual de NS2 es quien representa los puntos finales en donde los paquetes son generados, bien luego de definir la topología el paso siguiente es generar tráfico en la red en el script tcl, en NS2 existen varios tipos de trafico TCP pero los que se utilizan en la mayoría de escenarios son los agentes SINK que realizan la función de receptores de tráfico y además regresan mensajes ACk a los agentes que generan el tráfico para confirmar su recepción .

A continuación un resumen de los agentes TCP

- **TCP/Sink**

Es el encargado de enviar los paquetes ACK como respuesta a los paquetes que reciben de cierto generador afiliado, este tipo de agente puede es modificable.

- **TCP/Sink/DelAck**

Es en cargado de indicarnos cuando un paquete se encuentra fuera de orden, lo hace a través de un ACK negativo y consta con un intervalo de tiempo a través de ACK.

- **TCP/Sink/Sack1**

Es un agente inteligente que permite seleccionar una acción optima con los paquetes descartados, esto con el fin de que mejore la comunicación en la red , impidiendo la retransmisión de paquetes ya entregados.

Una vez definido el tipo de agente que se va a emplear se procede a crear una aplicación que recorra a través del agente TCP, la aplicación más común es FTP que cuenta con un tamaño máximo de 1024 bytes para ser transportados a través de TCP.

Para brindar prioridad en el envío de paquetes TCP existen 4 niveles, en donde 0 es el nivel de máxima prioridad y 3 el de menor prioridad, en la mayoría de escenarios no se utiliza prioridad en el envío de paquetes entre nodos.

A continuacion se muestra un ejemplo de la configuración de un agente TCP en los scripts tcl :

Figura 36 Configuración de un agente Tcp

Agente Udp

Otro protocolo de transporte usado como agente de envío de datos es UDP, al ser un protocolo que envía paquetes sin utilizar preliminarmente una conexión es muy útil para aplicaciones con tráfico de voz, utiliza la norma ITU G7.11 que cuenta con una metodología de compresión de datos para la transmisión de voz, al igual que TCP se

debe definir una aplicación para que viaje dentro del agente UDP , en este caso dicha aplicación se denomina CBR (Constant Bit Rate).

A continuación se muestra un ejemplo de la configuración de un agente UDP en los scripts tcl :

Figura 37 Configuración de un agente Udp

CAPÍTULO 3

3.1 ESCENARIOS

3.1 Consideraciones Escenario A

Para el presente trabajo se han diseñado 2 tipos de escenarios para demostrar el funcionamiento del protocolo AODV y SAODV, el escenario A está compuesto de 6 nodos ubicados arbitrariamente en el mismo rango de cobertura, en donde se implementó la presencia de un nodo malicioso o un nodo que no pertenece a la red con el fin de visualizar el comportamiento del protocolo SAODV, el nodo que no pertenece

a la red es el nodo #1 el cual se encuentra dentro de la ruta que el protocolo Aodv descubrió para enviar paquetes desde el origen (nodo #0) hacia su destino (nodo #5), esto quiere decir que los paquetes jamás llegarán a su destino ya que en el camino el protocolo Saodv detectará el nodo malicioso y descartará todos los paquetes que lleguen a él .

3.1.1 Topología , Dimensionamiento

El diseño de los escenarios de simulación es un trabajo importante ya que de ellos dependerá que los resultados obtenidos se asemejen a un entorno real, así como escoger adecuadamente los diferentes parámetros de una red Ad Hoc con el fin de establecer una simulación óptima para un fácil entendimiento e interpretación .

En el escenario A se implementa un escenario con un nodo malicioso, como se comentó en el capítulo I el protocolo SAODV brinda seguridad al protocolo AODV y está en la capacidad de detectar nodos maliciosos, en este escenario el protocolo SAODV lo ubicará mediante un número de identificación y lo aislará completamente de la red haciendo que todos los paquetes que lleguen al nodo malicioso sean descartados inmediatamente.

A continuación se muestra algunas consideraciones que se consideraron para el presente trabajo

Topología

En este tipo de red no se considera una topología definida por lo que los nodos pueden estar en constante movimiento, por lo que se colocó arbitrariamente 6 nodos en medio de un escenario bidimensional ya que así facilitaría la interpretación de los resultados y se visualiza de manera clara el intercambio de paquetes entre los nodos.

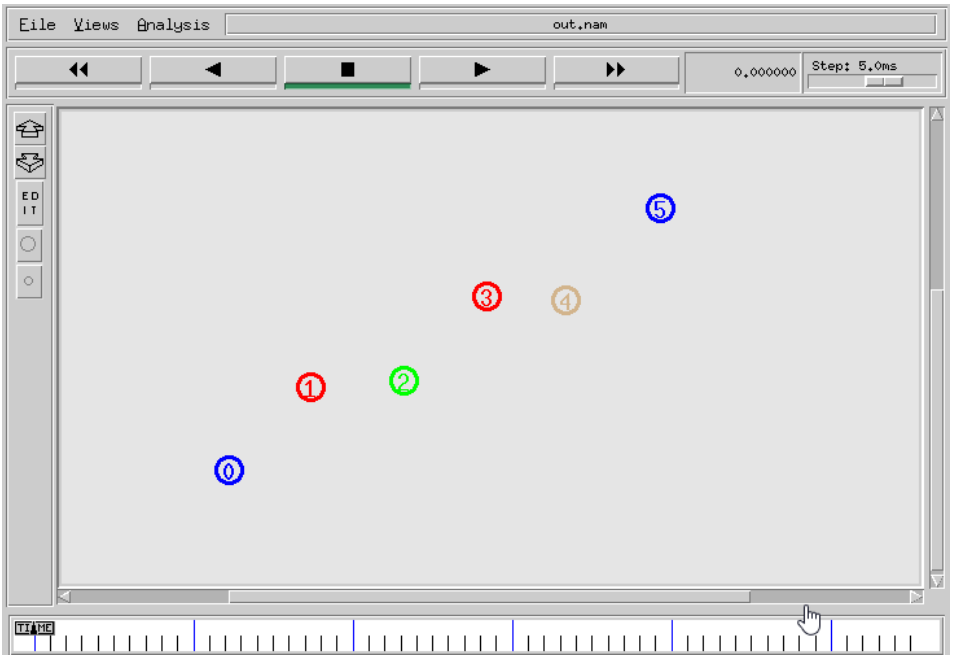


Figura 38 Topología Escenario A

Dimensionamiento

Para el dimensionamiento de la red se utilizó un número reducido de nodos , ya que el protocolo SAODV es experimental y con un número reducido fácilmente se entiende el funcionamiento del protocolo así como el intercambio de paquetes en la red , con un número elevado no se podría visualizar claramente el interactuar de los elementos de la red .

3.1.2 Parámetros de la red Ad Hoc

Para simular una red Ad-Hoc se debe considerar algunos parámetros típicos de una red inalámbrica, el simulador NS2 nos permite configurar dichos parámetros en el script tcl, la configuración es la siguiente

```
set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(ant)            Antenna/OmniAntenna
set val(ll)             LL
set val(ifq)            Queue/DropTail/PriQueue
set val(ifqlen)         50
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(nn)             6
set val(rp)             AODV
set val(x)              800
set val(y)              800
```

Figura 39 Parámetros del Escenario A

En donde

- **chan:** Tipo de canal con el que trabajaremos en este caso es Wireless
- **prop:** Modelo del radio de propagación
- **ant:** Tipo de Antena
- **ll:** Tipo de capa de enlace
- **ifq:** tipo de interface queue
- **ifqlen:** Numero de paquetes en ifq
- **netif:** Tipo de interface de red
- **mac:** Tipo de MAC
- **nn:** Numero de nodos del escenario
- **rp:** Protocolo de enrutamiento que se utiliza

Cabe recalcar que la mayoría de estos parámetros ya vienen por defecto en el caso de simular una red inalámbrica, los campos que se pueden modificar son val(x) y val(y) que son las dimensiones del escenario de simulación.

3.1.3 Modelos de conexión

Como se indicó en el capítulo anterior el NS2 trabaja regularmente con 2 tipos de agentes para inyectar tráfico a la red (TCp y UDP) y con ellos sus respectivas aplicaciones que van a viajar junto a los paquetes de tráfico (FTP y CBR), para configurar el tipo de tráfico que se desea que exista en la red se debe modificar el script tcl como se muestra a continuación :

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n(0) $tcp0
set tcp1 [new Agent/TCP]
$ns attach-agent $n(1) $tcp1
set tcp2 [new Agent/TCP]
$ns attach-agent $n(2) $tcp2
set tcp3 [new Agent/TCP]
$ns attach-agent $n(3) $tcp3
set tcp4 [new Agent/TCP]
$ns attach-agent $n(4) $tcp4
set tcp5 [new Agent/TCP]
$ns attach-agent $n(5) $tcp5
```

Figura 40 Creación de agentes Tcp

En la figura 38 anterior se visualiza como se crean agentes TCP para establecer la comunicación entre los nodos de la red, cada agente se asocia con un nodo por ejemplo el agente tcp0 se asocia con el nodo (0).

```
proc attach-CBR-traffic { node sink size interval } {
   #Get an instance of the simulator
   set ns [Simulator instance]
   #Create a CBR  agent and attach it to the node
   set cbr [new Agent/CBR]
   $ns attach-agent $node $cbr
   $cbr set packetSize_ $size
   $cbr set interval_ $interval
```

Figura 41 Instancia en el simulador

Aquí se obtiene una instancia en el simulador, se crea la aplicación CBR y se adjunta al nodo

3.1.4 Funcionamiento

Primer evento

En el escenario A existe un nodo malicioso que será el nodo (1), por lo que van a existir 2 tipos de eventos, el primer evento actúa el protocolo AODV, que como se indicó en el capítulo I, no implementa ningún método o algoritmo de seguridad entonces lo que harán los nodos es enviar los paquetes al nodo malicioso (nodo(1)) como que fuera un nodo más de la red y los paquetes llegaran al destino sin ningún tipo de protección.

En la figura 40 se visualiza como los paquetes CBR se generan desde el nodo (0) que es el origen, llegan al nodo malicioso (nodo (1)) y este continua transmitiendo al resto de la red.

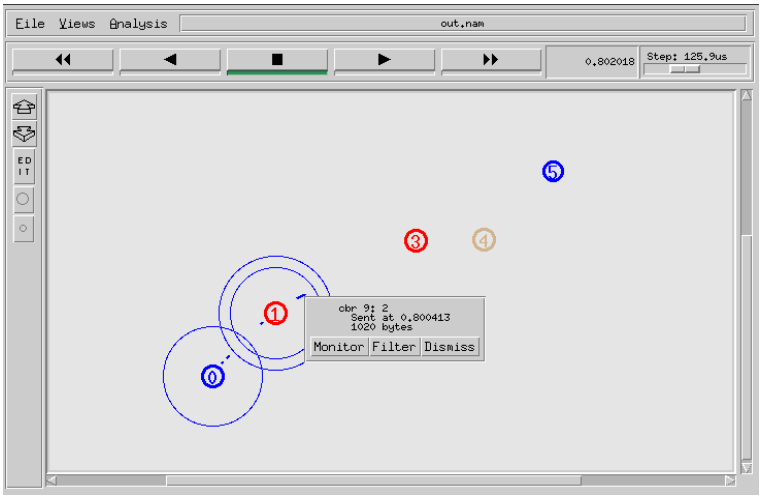


Figura 42 Funcionamiento protocolo Aodv

En la figura 41 se observa que los paquetes CBR han llegado a su destino el nodo (5) desde el origen nodo (0),

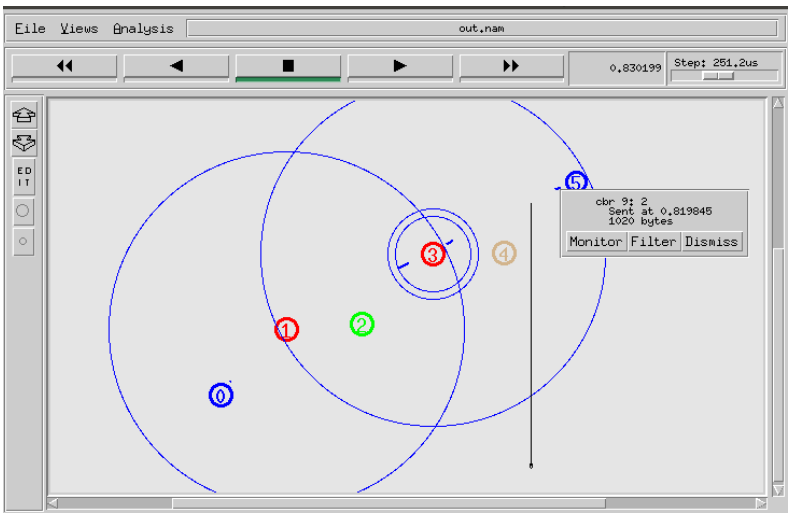


Figura 43 Funcionamiento protocolo Aodv

Adicional se observa que los paquetes no sufren ningún tipo de cambio, escogemos arbitrariamente el paquete CBR #11 para ver los detalles, ha sido enviado desde el nodo (0) hasta el nodo (1) a los 0.84 segundos de haber empezado la simulación con un tamaño de 1020 bytes.

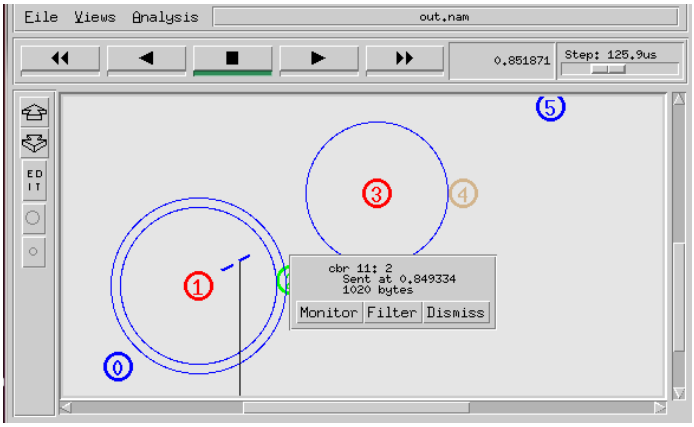


Figura 44 Funcionamiento protocolo Aodv

Segundo evento

En el segundo evento ya interviene el protocolo SAODV, como se sabe el nodo maliciosos es el nodo (1), el protocolo comparará su número de identificación con los números de los nodos que se encuentran registrados en la red y como no pertenece entonces lo aísla de la red y todos los paquetes que llegan al nodo malicioso son descartados como se muestra en la figura 43.

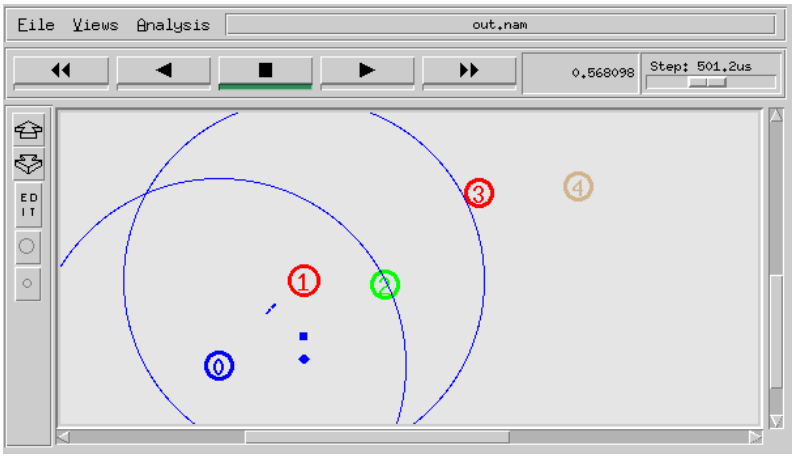


Figura 45 Funcionamiento protocolo Saodv

Como se observa en la figura 44 el paquete CBR #2 llega desde el nodo 0 al nodo 1 con un tamaño de 1020 bytes y el funcionamiento de la red es normal como que se tratará del protocolo AODV los paquetes cuentan con un identificador y tamaño de paquete

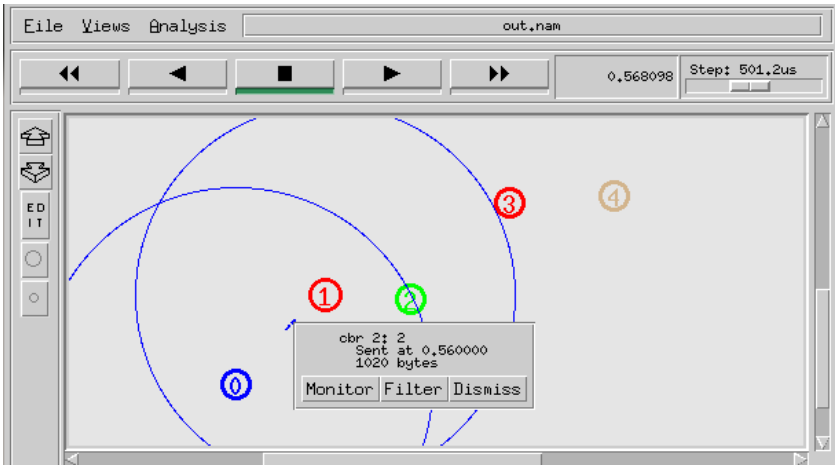


Figura 46 Funcionamiento protocolo Saodv

Una vez que el paquete llega al nodo malicioso entonces los paquetes son descartados como se muestra en la figura 45 los paquetes ya no cuentan con identificador ni con un tamaño de paquete.

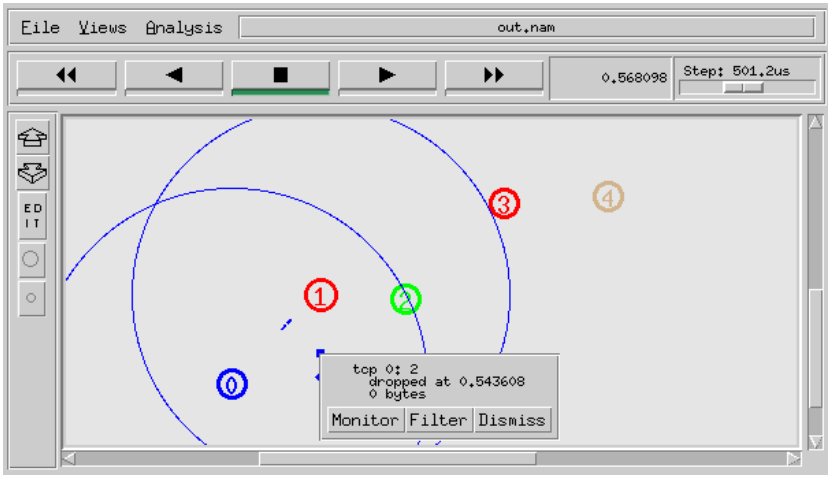


Figura 47 Funcionamiento protocolo Aodv

3.1.5 Archivos de traza

El archivo de traza nos indica cómo se comporta la red, en este archivo se puede visualizar todas la rutas que toman los paquetes para llegar hacia su destino, nodos origen, fuente, numero de secuencia , tipo de paquetes , tamaño de paquetes , direcciones MAC , puerto origen , puerto destino entre otros parámetros de una red, para generar un archivo de traza (.trace) se modifica el script tcl de configuración

añadiendo los siguientes comandos en donde se elige el nombre del archivo que se desea generar en nuestro caso es **normal.trace**

```
set namtrace [open out.nam w]
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
set topo [new Topography]
$topo load_flatgrid 800 800

set wireless_tracefile [open normal.trace w]
$ns trace-all $wireless_tracefile

create-god $val(nn)
```

Figura 48 Generación de un archivo de traza

El archivo de traza tiene un formato definido según los protocolos, es molestoso interpretarlo ya que obtiene varios campos en cada mensaje, y cada simulación genera miles de paquetes por lo que se tomará en cuenta un mensaje que va desde un origen hasta un destino con los mensajes de respuesta de ruta incluidos.

Para el caso del protocolo AODV el formato de paquetes en el archivo trace es el siguiente y a partir de este formato se regirá para realizar la interpretación del funcionamiento del protocolo.

Tabla 5

Campos de un archivo de traza

Column Number	What Happened?	Values for instance...
1	It shows the occured event	's' SEND, 'r' RECEIVED, 'D' DROPPED
2	Time at which the event occured?	10.000000000
3	Node at which the event occured?	Node id like 0
4	Layer at which the event occured?	'AGT' application layer, 'RTR' routing layer, 'LL' link layer, 'IFQ' Interface queue, 'MAC' mac layer, 'PHY' physical layer
5	show flags	—
6	shows the sequence number of packets	0
7	shows the packet type	'cbr' CBR packet, 'DSR' DSR packet, 'RTS' RTS packet generated by MAC layer, 'ARP' link layer ARP packet
8	shows size of the packet	Packet size increases when a packet moves from an upper layer to a lower layer and decreases when a packet moves from a lower layer to an upper layer
9	[....]	It shows information about packet duration, mac address of destination, the mac address of source, and the mac type of the packet body.
10	show flags	—
11	[....]	It shows information about source node ip : port number, destination node ip (-1 means broadcast) : port number, ip header ttl, and ip of next hop (0 means node 0 or broadcast).

Fuente: (Implementación del protocolo Aodv en NS2)

En donde:

- **Columna 1:** indica el tipo de evento que sucede puede ser s->envía, r->recibe, d->paquete descartado
- **Columna 2:** indica el tiempo en el que ocurrió el evento
- **Columna 3:** nodo que realizó el evento
- **Columna 4:** capa en la que ocurrió el evento
- **Columna 5:** banderas(----)
- **Columna 6:** muestra el número de secuencia del paquete
- **Columna 7:** tipo de paquete
- **Columna 8:** tamaño de paquete
- **Columna 9:**[] cadena que indica la duración del paquete, dirección mac destino, dirección mac origen ,tipo de mac
- **Columna 10:** [] cadena que indica ip origen: número de puerto, destino IP (si es -1 es enviado mediante broadcast), puerto destino, ttl y el próximo salto.

Primer evento

```
s 0.500000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
s 0.500235000 _0_ MAC  --- 0 AODV 106 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501083471 _1_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501083745 _2_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501108471 _1_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501108745 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
s 0.503454489 _1_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
s 0.503789489 _1_ MAC  --- 0 AODV 106 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.504637822 _2_ MAC  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.504637960 _0_ MAC  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.504638234 _3_ MAC  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.504662822 _2_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.504662960 _0_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.504663234 _3_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
s 0.506304688 _3_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
s 0.506919688 _3_ MAC  --- 0 AODV 106 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
s 0.507497397 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [2:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.507768022 _4_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507768160 _2_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507768434 _1_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507768434 _5_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507793022 _4_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507793160 _2_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507793434 _1_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507793434 _5_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
s 0.507793434 _5_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [5:255 0:255 30 3] [0x4 1 [5 4] 10.000000] (REPLY)
```

Figura 49 Archivo de traza generado por el Escenario A protocolo Aodv

Como se sabe el protocolo AODV empieza la transmisión de un paquete **RREQ** para realizar el **descubrimiento de ruta,** lo que va a suceder es que el nodo origen (0) envía un broadcast a todos los nodos de la red hasta que aparezca un nodo que tenga una ruta para llegar al destino o simplemente ese nodo sea el destino, y estos a su vez siguen enviando broadcast con la misma petición , cuando se encuentra al nodo destino se envía un paquete **RREP** de forma unicast por la misma ruta que siguió el paquete **RREQ**.

Como se puede observar en la primera fila el nodo 0 envía un paquete mediante broadcast (ya que en la posición 10 se encuentra -1) a toda la red a los 0.5 seg de haber empezado la simulación el paquete es de tipo **Request** para el descubrimiento de ruta el tamaño del paquete es 48 bytes

```
s 0.500000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
```

Figura 50 Envió de mensajes REQ del nodo origen

En la figura 48 se observa que el paquete REQ que es enviado por el nodo 0 es recibido por el resto de nodos en este caso el nodo 1 y 2, y estos a su vez como no

conocen la ruta ya que es una red nueva con las tablas de rutas vacías entonces realizan un broadcast al resto de miembros de la red

```
r 0.501083471 _1_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501083745 _2_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501108471 _1_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.501108745 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)

s 0.503454489 _1_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
s 0.503789489 _1_ MAC  --- 0 AODV 106 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
```

Figura 51 Reenvío de mensajes REQ por parte del resto de nodos

En la figura 49 se observa que los nodos 3 y 4 también han recibido los REQ de los nodos 1 y 2 y como no conocen el destino siguen reenviando mediante broadcast los mensajes para descubrir el nodo destino.

```
r 0.504663234 _3_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
s 0.506304688 _3_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
s 0.506919688 _3_ MAC  --- 0 AODV 106 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
s 0.507497397 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [2:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 0.507768022 _4_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507768160 _2_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507768434 _1_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
r 0.507768434 _5_ MAC  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
```

Figura 52 Reenvío de mensajes REQ por parte del resto de nodos

Hasta que llega un REQ al nodo 5, como es el nodo destino entonces envía un paquete tipo REP indicando que es el destino y lo envía hacia el nodo 0 recorriendo la misma ruta del mensaje RREQ que le llegó.

```
r 0.507793434 _5_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 28 0] [0x2 3 1 [5 0] [0 4]] (REQUEST)
s 0.507793434 _5_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [5:255 0:255 30 3] [0x4 1 [5 4] 10.000000] (REPLY)
```

Figura 53 Respuesta del nodo origen con mensaje REP

Los mensajes RREP continúan regresando por la misma ruta por la que se envió los RREQ hasta llegar al nodo o que es el origen

```
s 0.517705051 _3_ MAC  --- 0 AODV 102 [13a 1 3 800] ------- [5:255 0:255 29 1] [0x4 2 [5 4] 10.000000] (REPLY)
r 0.518521797 _1_ MAC  --- 0 AODV 44 [13a 1 3 800] ------- [5:255 0:255 29 1] [0x4 2 [5 4] 10.000000] (REPLY)

s 0.522995625 _1_ MAC  --- 0 AODV 102 [13a 0 1 800] ------- [5:255 0:255 28 0] [0x4 3 [5 4] 10.000000] (REPLY)
r 0.523812096 _0_ MAC  --- 0 AODV 44 [13a 0 1 800] ------- [5:255 0:255 28 0] [0x4 3 [5 4] 10.000000] (REPLY)
s 0.523822096 _0_ MAC  --- 0 ACK 38 [0 1 0 0]
r 0.523837096 _0_ RTR  --- 0 AODV 44 [13a 0 1 800] ------- [5:255 0:255 28 0] [0x4 3 [5 4] 10.000000] (REPLY)
```

Figura 54 Respuesta del nodo origen con mensaje REP

Una vez que el origen sabe la ruta por donde enviar los paquetes al destino entonces comienza a enviar trafico mediante paquetes CBR

```
s 0.525153039 _0_ MAC  --- 0 cbr 1078 [13a 1 0 800] ------- [0:2 5:0 30 1] [0] 0 0
s 0.530000000 _0_ AGT  --- 1 cbr 1000 [0 0 0 0] ------- [0:2 5:0 32 0] [1] 0 0
r 0.530000000 _0_ RTR  --- 1 cbr 1000 [0 0 0 0] ------- [0:2 5:0 32 0] [1] 0 0
s 0.530000000 _0_ RTR  --- 1 cbr 1020 [0 0 0 0] ------- [0:2 5:0 30 1] [1] 0 0
r 0.533777511 _1_ MAC  --- 0 cbr 1020 [13a 1 0 800] ------- [0:2 5:0 30 1] [0] 1 0
s 0.533787511 _1_ MAC  --- 0 ACK 38 [0 0 0 0]
r 0.533802511 _1_ RTR  --- 0 cbr 1020 [13a 1 0 800] ------- [0:2 5:0 30 1] [0] 1 0
f 0.533802511 _1_ RTR  --- 0 cbr 1020 [13a 1 0 800] ------- [0:2 5:0 29 3] [0] 1 0
```

Figura 55 Envió de trafico cbr

Segundo evento

En el segundo evento cuando interviene el nodo malicioso , el protocolo SAODV si realiza el proceso mismo proceso de descubrimiento de ruta explicado en el primer evento con el protocolo AODV porque los nodos todavía no se conocen , y empiezan a enviar broadcast REQ entre todos los elementos de la red , una vez que se se conoce la ruta desde el origen al destino empiezan a enviar mensajes de respuesta REP con las identificaciones de los nodos, entonces ya la red sabe que el nodo malicioso es el #1 y empiezan a descartarse todos los paquetes que llegan hacia el nodo 1 como se muestra en el archivo de traza.

```
D 0.533802511 _1_ RTR LOOP 0 cbr 1020 [13a 1 0 800] ------- [0:2 5:0 29 1] [0] 1 0
D 0.533802511 _1_ RTR TTL 0 tcp 0 [0 0 0 0] ------- [0:0 0:0 0 0] [0 0] 0 0

D 0.569601414 _1_ RTR LOOP 2 cbr 1020 [13a 1 0 800] ------- [0:2 5:0 29 1] [2] 1 0
D 0.569601414 _1_ RTR TTL 0 tcp 0 [0 0 0 0] ------- [0:0 0:0 0 0] [0 0] 0 0

D 0.599921414 _1_ RTR LOOP 3 cbr 1020 [13a 1 0 800] ------- [0:2 5:0 29 1] [3] 1 0
D 0.599921414 _1_ RTR TTL 0 tcp 0 [0 0 0 0] ------- [0:0 0:0 0 0] [0 0] 0 0
```

Figura 56 Respuesta del nodo origen con mensaje REP

3.2 Consideraciones Escenario B

El escenario B es una red conformada por 6 nodos, en este escenario se implementa la seguridad a los mensajes que es lo que realiza el protocolo SAODV, los algoritmos de

seguridad que se aplican son mensajes con cadenas hash y mensajes encriptados con el fin de proteger la integridad y confidencialidad de los datos, en este escenario se visualizará como se envían paquetes de seguridad desde el nodo origen (nodo #0) hasta el nodo destino (nodo #5) , una vez que lleguen al nodo destino se desencripta los mensajes y se calcula un nuevo valor hash que se compara con el valor hash enviado , si son iguales se garantiza la integridad y confidencialidad del paquete.

3.2.1 Topología y Dimensionamiento

Para el escenario B se ha implementado seguridad en los mensajes AODV, por lo tanto no existe una topología definida como en el escenario A, el dimensionamiento también es similar con pocos nodos para comprender mejor el funcionamiento del protocolo SAODV al momento de encriptar los mensajes.

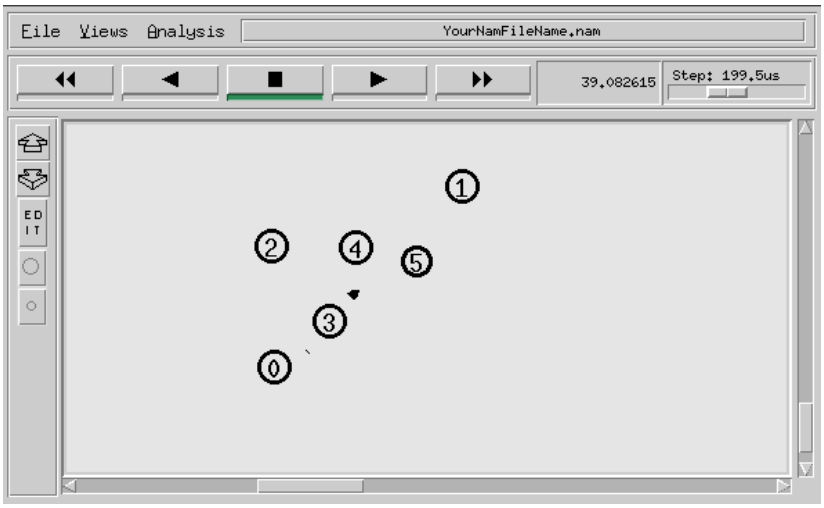


Figura 57 Topología Escenario B

3.2.2 Parámetros de la red Ad Hoc

Como se analizó en el escenario anterior aquí también se establecen casi los mismos parámetros de la red Ad-Hoc, también se comentó que dichos parámetros son propios de una red inalámbrica por lo que se repetirán en varios escenarios con varios protocolos, a continuación se indica la lista de parámetros utilizados en el escenario B.

```
set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            Queue/DropTail/PriQueue
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(ifqlen)         50
set val(nn)             6
set val(rp)             AODV
```

Figura 58 Parámetros Escenario B

En donde:

- **chan:** Tipo de canal con el que trabajaremos en este caso es Wireless
- **prop:** Modelo del radio de propagación
- **ant:** Tipo de Antena
- **ll:** Tipo de capa de enlace
- **ifq:** tipo de interface queue
- **ifqlen:** Numero de paquetes en ifq
- **netif:** Tipo de interface de red
- **mac:** Tipo de MAC
- **nn:** Numero de nodos del escenario
- **rp:** Protocolo de enrutamiento que se utiliza

3.2.3 Modelos de conexión

Los modelos de conexión entre nodos que se van a utilizar en este escenario son 2, en primer lugar los paquetes característicos del protocolo AODV mezclados con paquetes encriptados que son los que van a dar un poco de sobrecarga adicional en el funcionamiento de la red.

En la siguiente figura se observa la creación de un agente TCP que servirá para inyectar el tráfico de paquetes desde el nodo origen hacia el nodo destino, también se observa la aplicación FTP que viajará dentro del trafico TCP, esta configuración se llevará a cabo en el **primer evento** en donde solo actual el protocolo AODV sin ningún mecanismo de protección en el intercambio de mensajes

```
set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(5) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at 10.0 "$ftp start"
```

Figura 59 Modelos de conexión Escenario B

Para el **segundo evento** en donde ya se implementa seguridad en el intercambio de mensajes y en donde actúa ya el protocolo SAODV se generan 4 agentes Security_Packet en el script tcl que fueron creados mediante programación para que implementen seguridad a un paquete AODV como se detallará más adelante, cada agente está asociado a un nodo, por ejemplo el agente p0 se asocia al node_(0) y así sucesivamente con el resto de nodos y al finalizar se realiza la conexión entre los agentes que participarán en el intercambio de mensajes encriptados dentro de la red.

```
set p0 [new Agent/Security_packet]
$ns_ attach-agent $node_(0) $p0
$p0 set class_ 1

set p1 [new Agent/Security_packet]
$ns_ attach-agent $node_(1) $p1
$p1 set class_ 1

set p2 [new Agent/Security_packet]
$ns_ attach-agent $node_(4) $p2
$p2 set class_ 2

set p3 [new Agent/Security_packet]
$ns_ attach-agent $node_(5) $p3
$p3 set class_ 2

#Connect the two agents
$ns_ connect $p0 $p3
$ns_ connect $p1 $p2
```

Figura 60 Agentes de seguridad

3.2.4 Funcionamiento

Primer evento

En el escenario B como en el anterior escenario también van a existir 2 eventos, el primer evento se llevará a cabo la participación del protocolo AODV sin ningún tipo de seguridad en el intercambio de mensajes como es característica de dicho protocolo, a continuación se muestra el escenario con la simulación en curso

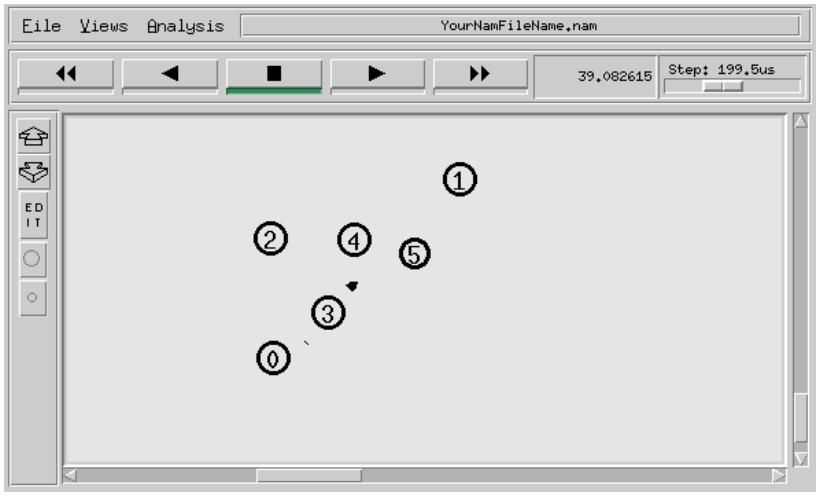


Figura 61 Funcionamiento protocolo Aodv

Se puede visualizar que los paquetes que llegan al nodo destino son TCP generados por el protocolo AODV, por ejemplo el paquete TCP#4775 es enviado a los 39.09 segundos de haber iniciado la simulación, con un tamaño de 1060 bytes y se puede visualizar que no existe ningún tipo de protección en el mensaje

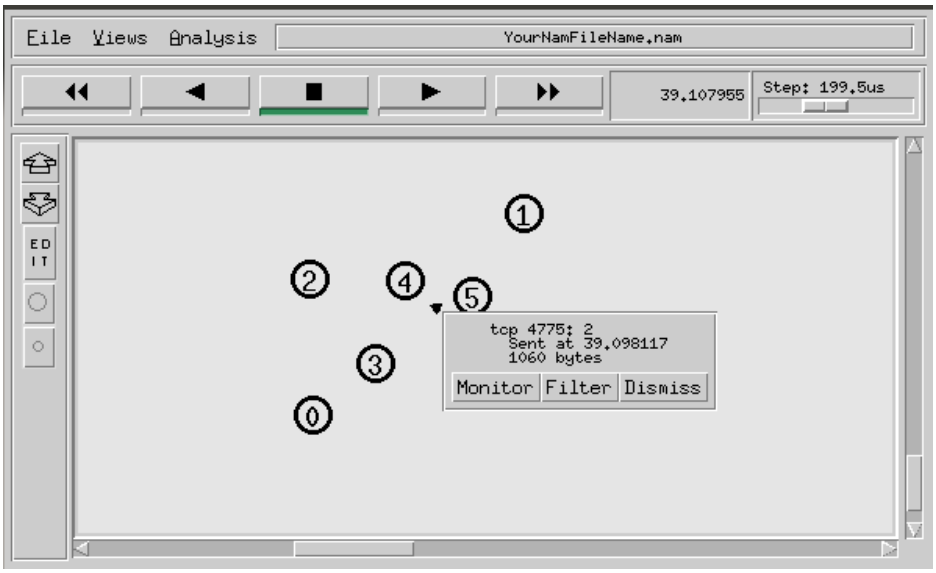


Figura 62 Funcionamiento protocolo Aodv

Como es característica del protocolo AODV también existen mensajes de respuesta (RREP) de tipo ACK que indican al nodo origen que el nodo destino ha recibido su mensaje

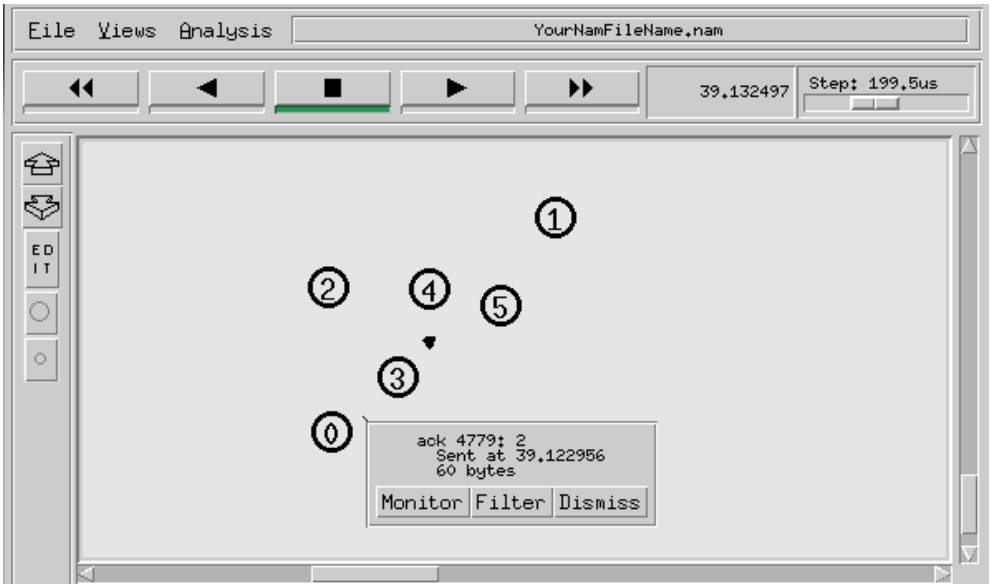


Figura 63 Funcionamiento protocolo Aodv

Segundo evento

En el segundo evento ya actúa el protocolo SAODV que implementa seguridad a los paquetes AODV mediante un algoritmo de cadena hash y adicional la encriptación de los mensajes, encripta algunos paquetes AODV para ver la diferencia entre un paquete encriptado y otro no encriptado, a continuación se muestra el funcionamiento del segundo evento

El escenario es el mismo anterior con 6 nodos desde el nodo 0 hasta el nodo 5, se va a enviar ciertos paquetes encriptado desde el nodo 0 hasta el nodo 5 hasta que finalice la simulación, a continuación se muestra el resultado de la simulación y el resultado de los mensajes encriptados.

```
user1@ubuntu:~/Desktop$ ns packet_s.tcl
num_nodes is set 6
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
Starting Simulation...
Message sent itisalongmessageIcansend with hashing 541705348
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
data intergity ensured
node 5 received packet from  0 with trip-time 9.8 ms - contend: lwlvdorqjphvvdjh
Lfdqvhqg - decrypted itisalongmessageIcansend -hash: 541705348
node 0 received packet from  5 with trip-time 11.7 ms - contend: Message_Accepte
d - decrypted _ -hash: 0
Message sent Itisashotermessage with hashing 9128228
Message sent test3 with hashing 406
Message sent test4 with hashing 486
data intergity ensured
node 0 received packet from  5 with trip-time 1.6 ms - contend: whvw7 - decrypte
d test4 -hash: 486
node 5 received packet from  0 with trip-time 3.4 ms - contend: Message_Accepted
 - decrypted _ -hash: 0
NS EXITING...
user1@ubuntu:~/Desktop$
```

Figura 64 Encriptación de mensajes

En la figura 62 se puede apreciar en primer lugar cómo se envía un código hash junto con una cadena de texto para descifrar desde el nodo 0 que es el origen hasta el nodo 5 que es el destino, se recibe el paquete a los 9.8ms como se muestra a continuación

```
node 5 received packet from  0 with trip-time 9.8 ms - contend: lwlvdorqjphvvdjh
Lfdqvhqg - decrypted itisalongmessageIcansend -hash: 541705348
```

Figura 65 Recepción del paquete de seguridad

A continuación el nodo 0 recibe un mensaje del nodo 5 a los 11.7 ms indicando que el mensaje del nodo 5 ha sido aceptado, después de desencriptar el paquete se ha comparado el nuevo valor hash con el enviado y coinciden por lo tanto el mensaje ha sido enviado con total protección como se indica a continuación

```
node 0 received packet from  5 with trip-time 11.7 ms - contend: Message_Accepte
d - decrypted _ -hash: 0
```

Figura 66 Mensaje desencriptado y aceptado

Adicional se envía un numero de hash adicional desde el destino mediante un test solo para proteger la integridad de los datos, el valor hash es enviado mediante un test 3 y un test 4 desde el nodo 0 hacia el nodo 5

```
Message sent test3 with hashing 406
Message sent test4 with hashing 486
```

Figura 67 Mensajes de test para verificar la integridad de los datos

El nodo 0 recibe un mensaje del nodo 5 a los 1.6 ms indicando que el hash calculado para proteger la integridad de los datos se lo ha realizado con el test 4 y con el hash 486 que es el mismo que envió el nodo 0 por lo tanto se despliega el mensaje de "**data intergity ensured"** indicando que la integridad de los datos está garantizada.

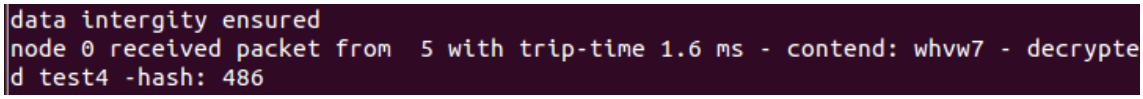


Figura 68 Integridad de datos asegurados

Finalmente el nodo 5 recibe la confirmación del nodo 0 a los 3.4 ms ,en donde le indica que el mensaje ha sido aceptado y que los datos encriptados han sido recibido con éxito y la integridad está respaldada.

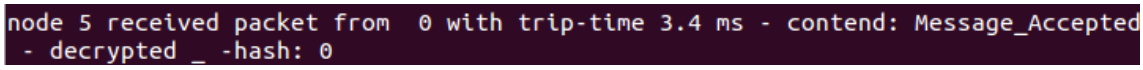


Figura 69 Mensaje de aceptación recibido por el nodo destino

Cabe recalcar que todos los paquetes encriptados que se envíen desde el nodo 0 hacia el nodo 5 seguirán con el mismo algoritmo hasta que finalice el tiempo de simulación, una manera más detallada de cómo se transportan los paquetes a través de la red se lo estudiará más adelante cuando analicemos los archivos de traza de la simulación.

Como se puede observar a comparación del primer evento que solo envía mensajes TCP para descubrir las rutas y ACK como mensajes de respuesta de ruta, en este evento además de enviar los mensajes de descubrimiento de ruta existen mensajes protegidos a través de cadenas hash y encriptación de los mismos lo que modificará la sobrecarga de la red con la implementación de las normas de seguridad.

3.2.5 Archivo de traza

Primer evento

```
s 10.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988084 _3_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988277 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988299 _4_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988304 _5_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
s 10.000988304 _5_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [5:255 0:255 30 0] [0x4 1 [5 4] 10.000000] (REPLY)
s 10.001869311 _3_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 10.003057395 _0_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 10.003057534 _5_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 10.003057579 _4_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 10.003057586 _2_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
s 10.005400011 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [2:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
r 10.006729218 _0_ RTR  --- 0 AODV 44 [13a 0 5 800] ------- [5:255 0:255 30 0] [0x4 1 [5 4] 10.000000] (REPLY)
```

Figura 70 Archivo de traza del Escenario B protocolo Aodv

En el primer evento se trata del funcionamiento normal del protocolo AODV con mensajes sin ningún tipo de protección, los mensajes se leen con el mismo formato explicado en el escenario anterior y también se explica cómo funciona el intercambio de mensajes del protocolo AODV empezando con el proceso conocido como **descubrimiento de ruta,** como se indica en la figura a continuación en primer lugar el nodo 0 envía un mensaje tipo REQ tipo broadcast .

```
s 10.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
```

Figura 71 Envió de mensajes Req del nodo origen

A continuación los nodos que reciben el mensaje al no tener información del nodo destino también envían un mensaje REQ al resto de nodos de la red , en la imagen siguiente se muestra como los nodos 2,3,4 reciben el mensaje REQ y lo envían nuevamente mediante broadcast.

```
r 10.000988084 _3_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988277 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988299 _4_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 10.000988304 _5_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)

s 10.001869311 _3_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)

s 10.005400011 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [2:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)

s 10.008979854 _4_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [4:255 -1:255 29 0] [0x2 2 1 [5 0] [0 4]] (REQUEST)
```

Figura 72 Reenvío de mensajes Req por parte del resto de nodos

Una vez que se encuentra el nodo destino (5), este envía un mensaje tipo REP indicando que es el nodo destino, el mensaje lo envía por el mismo trayecto por el que se estableció la ruta de los mensaje REQ directamente hasta llegar al nodo destino (0).

```
s 10.000988304 _5_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [5:255 0:255 30 0] [0x4 1 [5 4] 10.000000] (REPLY)
```

```
r 10.006729218 _0_ RTR  --- 0 AODV 44 [13a 0 5 800] ------- [5:255 0:255 30 0] [0x4 1 [5 4] 10.000000] (REPLY)
```

Figura 73 Respuesta del nodo destino mediante un mensaje REP

Segundo evento

En el segundo evento ya se insertan los paquetes de seguridad pero el proceso de **descubrimiento de ruta** del protocolo AODV no cambia es el mismo que en el escenario anterior, como se indica en la imagen adjunta se observa que se repite el mismo proceso empezando por el nodo 0 enviando REQ hacia el resto de nodos y estos a su vez replicando los mensaje REQ al resto de la red hasta encontrar el nodo destino que envía un REP al nodo origen

```
s 0.000000000 _0_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
s 0.000115000 _0_ MAC  --- 0 AODV 106 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000963084 _3_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000963277 _2_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000963299 _4_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000963304 _5_ MAC  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000988084 _3_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000988277 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000988299 _4_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
r 0.000988304 _5_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 30 0] [0x2 1 1 [5 0] [0 4]] (REQUEST)
```

Figura 74 Archivo de traza del escenario B protocolo Saodv

En este evento la diferencia es que ya se insertan paquetes de seguridad aleatoriamente, aumentando sobrecarga a la red como se analizará en el capítulo IV, en la siguiente imagen se visualiza que una vez finalizado el proceso de descubrimiento de ruta y que el nodo origen (0) recibe el REP del nodo destino (5) entonces se empiezan a transmitir los paquetes de seguridad con origen nodo 0 y destino nodo 5 a los 0.0067 seg de haber empezado la simulación

```
r 0.006729218 _0_ RTR  --- 0 AODV 44 [13a 0 5 800] ------- [5:255 0:255 30 0] [0x4 1 [5 4] 10.000000] (REPLY)
s 0.006729218 _0_ RTR  --- 0 Security_packet 20 [0 0 0 0] ------- [0:1 5:1 30 5]
```

Figura 75 Mensaje de seguridad enviado desde el nodo origen

Una vez que el nodo 0 empieza a enviar el paquete de seguridad al nodo 5 pasa por la ruta trazada a través de los REQ , pasa por el resto de nodos hasta llegar al nodo 5, a continuación se muestra como el nodo 0 envía el paquete de seguridad a los 0.00918 seg y el nodo 5 lo recibe a los 0.00983 seg , adicional se puede verificar que se envían paquetes de diferente tamaño (20 y 78) que corresponde al mensaje encriptado con la

función hash y el otro mensaje que protege la integridad mediante los test.

```
s 0.009183380 _0_ MAC  --- 0 Security_packet 78 [13a 5 0 800] ------- [0:1 5:1 30 5]
r 0.009807684 _5_ MAC  --- 0 Security_packet 20 [13a 5 0 800] ------- [0:1 5:1 30 5]
r 0.009832684 _5_ AGT  --- 0 Security_packet 20 [13a 5 0 800] ------- [0:1 5:1 30 5]
```

Figura 76 Mensaje de seguridad recibido por el destino

Los mensajes de seguridad también tienen un mensaje de respuesta indicando si el mensaje desencriptado es aceptado o no y eso se lo puede visualizar en la siguiente imagen donde el nodo destino (5) envía mensajes de respuesta al nodo origen (0), y finalmente este es el proceso de un mensaje de seguridad cabe recalcar que en el transcurso de la simulación se envían algunos de estos mensajes.

```
s 0.009832684 _5_ RTR  --- 1 Security_packet 20 [0 0 0 0] ------- [5:1 0:1 30 0]
s 0.011008292 _5_ MAC  --- 1 Security_packet 78 [13a 0 5 800] ------- [5:1 0:1 30 0]
r 0.011632596 _0_ MAC  --- 1 Security_packet 20 [13a 0 5 800] ------- [5:1 0:1 30 0]
r 0.011657596 _0_ AGT  --- 1 Security_packet 20 [13a 0 5 800] ------- [5:1 0:1 30 0]
```

Figura 77 Mensaje de seguridad aceptado

3.3 Configuraciones

3.3.1 Escenario A

Para lograr implementar el protocolo SAODV en el escenario A se realizó una serie de configuraciones en el código fuente del NS2 para poder detectar el nodo malicioso en la red, a continuación se resume todas las modificaciones y consideraciones que se tomaron en cuenta para realizar la simulación

- Se debe modificar el código fuente del protocolo AODV , se declara un nodo como malicioso y simplemente si lo detecta los paquetes son descartados, se modifican 2 archivos del protocolo AODV (aodv.h y aodv.cc)
- A continuación se dirige al archivo aodv.h que es el archivo de cabecera del protocolo , se encuentra en el directorio ns-allinone-2.35/ns-2.35/aodv y declaramos una variable de tipo bool que se denomine malicious como se indica a continuación

```
protected:
        int             command(int, const char *const *);
        int             initialized() { return 1 && target_; }
        bool            malicious;
```

Figura 78 Declaración de una variable tipo booleana

- Después se dirige al archivo aodv.cc que es el archivo en donde se muestra el funcionamiento del protocolo, se encuentra en el directorio ns-allinone-2.35/ns-2.35/aodv , inicializamos la variable malicious como false ya que después cambiará de estado a true cuando el nodo sea detectado como malicioso.

```
LIST_INIT(&nbhead);
LIST_INIT(&bihead);
malicious=false;
logtarget = 0;
ifqueue = 0;
```

Figura 79 Inicialización de la variable

- En el mismo archivo añadimos la siguiente instrucción, en donde comparamos el nodo malicioso con un número que vendría a ser el identificador del nodo mediante el comando de comparación **if**

```
AODV::command(int argc, const char*const* argv) {
  if(argc == 2) {
  Tcl& tcl = Tcl::instance();

if(strcmp(argv[1], "malicious") == 0) {
    malicious = true;
   return TCL_OK;
}
```

Figura 80 Comparación de la variable con un identificador

A continuación añadimos la siguiente instrucción, en donde se implementa el comportamiento del nodo malicioso mediante el código rt_resolve(Packet *p), y si el nodo es detectado la variable se ubica en true y los paquetes son descartados mediante la función DROP_RTR_ROUTE_LOOP

```
void
AODV::rt_resolve(Packet *p) {
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
aodv_rt_entry *rt;

if(malicious==true)
{
drop(p,DROP_RTR_ROUTE_LOOP);
}
```

Figura 81 Función que descarta los paquetes

- Una vez modificados los archivos antes indicados procedemos a recompilar el software NS2 para que los cambios tengan efecto en el simulador, se abre un terminal y se ejecuta los siguientes comandos en el directorio ns-allinone-2.35/ns-2.35

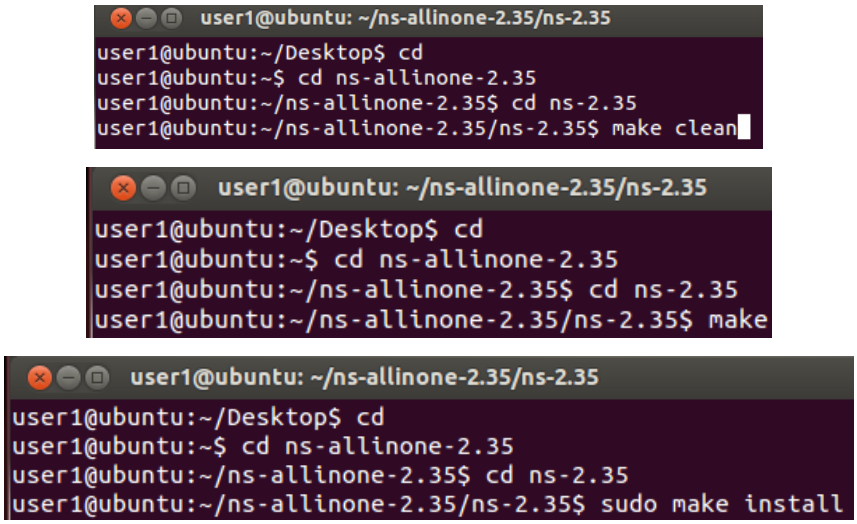


Figura 82 Comandos del NS2 para ejecutar el nuevo agente

- Finalmente procedemos a modificar nuestro script tcl en donde se visualiza que declaramos 6 nodos y el nodo 1 es el malicioso ,adicional se puede escoger que nodo se quiere que actué como malicioso con el comando **malicious** al final de la sentencia, para que la red actué con el protocolo AODV simplemente se comenta la línea del nodo malicioso para que no reconozca el nodo y actué normalmente como se indica a continuación

```
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns initial_node_pos $n($i) 30+i*100
}
$ns at 0.0 "[$n(1) set ragent_] malicious"

$ns at 0.0 "$n(0) setdest 100.0 100.0 3000.0"
$ns at 0.0 "$n(1) setdest 200.0 200.0 3000.0"
$ns at 0.0 "$n(2) setdest 300.0 200.0 3000.0"
$ns at 0.0 "$n(3) setdest 400.0 300.0 3000.0"
$ns at 0.0 "$n(4) setdest 500.0 300.0 3000.0"
$ns at 0.0 "$n(5) setdest 600.0 400.0 3000.0"
```

Figura 83 Archivo Tcl

3.3.2 Escenario B

La implementación de seguridad a un protocolo de una red inalámbrica es muy complejo ya que por ser un medio compartido se encuentra expuesto a elementos maliciosos que quieran entrar a la red para diferentes fines, es por eso que se implementó un modelo de seguridad a los mensajes del protocolo AODV mediante cadenas hash y encriptación de mensajes que sustituirán a las firmas digitales con las que trabaja el protocolo SAODV.

El algoritmo de cifrado que se utilizó se denomina cifrado de Cesar, que básicamente a una letra de una cadena de caracteres original la sustituye con otra letra que ocupe una posición más adelante en el alfabeto, la posición se puede elegir arbitrariamente en el caso de nuestro proyecto se eligió que cada letra se reemplace por otra letra con un desplazamiento de 3 posiciones más adelante en el alfabeto.

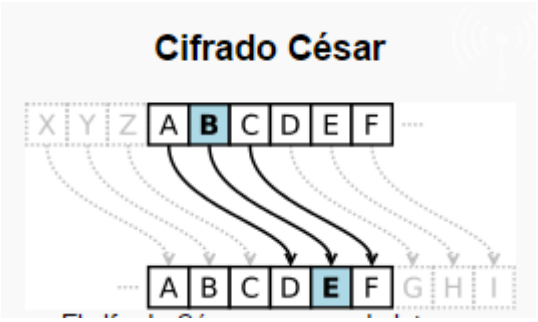


Figura 84 Cifrado por el método César

Ahora bien para implementar estos algoritmos de seguridad se utilizó la programación en lenguaje C que utiliza el simulador NS2 para crear nuevos protocolos y nuevos agentes, en este caso estaríamos creando un nuevo agente de seguridad, para

esto se ha tomado como referencia un proyecto que brinda seguridad a una red cableada y lo adaptamos para que funcione en una red inalámbrica.

- En primer lugar se crean 2 archivos (security.h) que es el archivo de cabecera del paquete de seguridad y el (security.cc) que es el archivo en donde se encuentra el funcionamiento que realiza el agente de seguridad, y se los guarda en una carpeta llamada security que va a ser creada en el directorio ns-allinone-2.35/ns-2.35 como se muestra a continuación

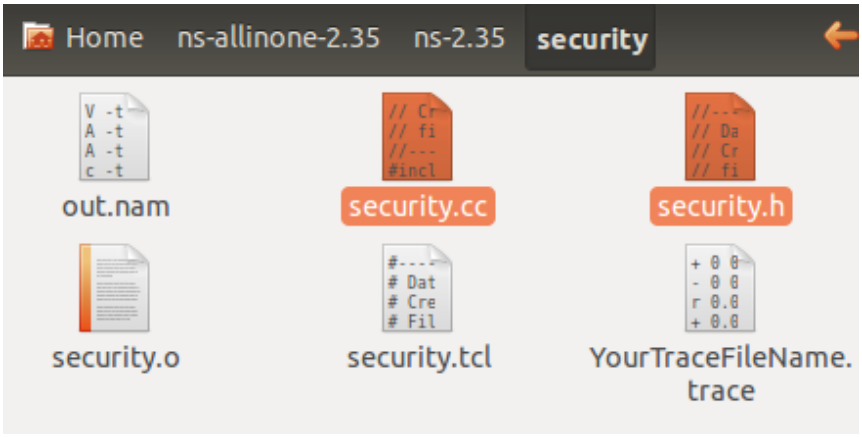


Figura 85 Directorio de la carpeta security

- A continuación se procede a generar el archivo security.h en donde se especificará todos los elementos que necesitará el agente para la cabecera, adicional se agrega todas las librerías necesarias que se necesitan para conectarse con el entorno del lenguaje C de NS2 y se declaran las variables necesarias para que el agente de seguridad realice su trabajo, variables como numero de secuencia, tiempo de envió, mensajes de respuesta entre otras.

```
#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"

struct hdr_security_packet {
        char ret;
        double send_time;
        double rcv_time;        // when security packet arrived at receiver
        int seq;                // sequence number
        char data[128];
        unsigned int hashvalue;

        // Header access methods
        static int offset_; // required by PacketHeaderManager
        inline static int& offset() { return offset_; }
        inline static hdr_security_packet* access(const Packet* p) {
                return (hdr_security_packet*) p->access(offset_);
        }
};
```

Figura 86 Archivo security.h

- En este archivo también se indica el método de acceso a la cabecera del paquete asi como la clase Security_packetAgent que se va a conectar con el archivo security.cc

```
class Security_packetAgent : public Agent {
public:
        Security_packetAgent();
        int seq;
        int oneway;     // enable seq number and one-way delay printouts
        virtual int command(int argc, const char*const* argv);
        virtual void recv(Packet*, Handler*);
        void encryption(char* out);
        void decryption(char* out);
        unsigned int hashing (char value[], unsigned int len);
};
#endif // ns_security_packet_h
```

Figura 87 Archivo security.h

- A continuación nos dirigimos al archivo security.cc en donde se implementan todas las funciones de seguridad, en la siguiente imagen se observa el algoritmo de cifrado CESAR , al momento de encriptar los datos se declara una variable key=3 que son las posiciones que van a recorrer las letra en el alfabeto hacia adelante y así viajan los datos encriptados, al momento que el mensaje llega al destino debe desencriptar los datos realizando la operación contraria ,es decir se recorre 3 posiciones del alfabeto hacia atrás y se obtiene el mensaje original como se muestra en la imagen adjunta

```
// -- CESAR encryption function ----------
void Security_packetAgent::encryption(char out[])
{
        int key =3;
        int i=0;
                for (i=0;i<strlen(out);i++)
        {
                out[i]=(out[i]+key)%128;
        }
}
// ---- CESAR decryption  ------------------
void Security_packetAgent::decryption(char out[])
{
        int key =3;
        int i=0;
                for (i=0;i<strlen(out);i++)
        {
                out[i]=(out[i]-key)%128;
        }

}
```

Figura 88 Archivo security.cc

- La función hash es un algoritmo de seguridad muy utilizado hoy en día, el algoritmo que se utilizó en este proyecto es una función hash polinomial, en donde se suman todos los códigos ascii de los caracteres multiplicados por un coeficiente x, adicional la mayoría de algoritmos de función hash como MD5, RSA utilizan palabras de 32 bits para convertirlo en un valor hash.

```
//--------------hashing fucntion-------------
unsigned int Security_packetAgent::hashing(char value[], unsigned int len)
{
   char *word = value;
   unsigned int ret = 0;
   unsigned int i;
   for(i=0; i < len; i++)
   {
      int mod = i % 32;
      ret ^=(unsigned int) (word[i]) << mod;
      ret ^=(unsigned int) (word[i]) >> (32 - mod);
   }
   return ret;
}
```

Figura 89 Archivo security.cc

- Una vez implementada la función hash se procede a comparar los valores hash , el nodo origen envía un valor hash cuando empieza a transmitir el paquete encriptado que sería la variable **hashvalue** y se compara con el nuevo valor hash que el nodo origen calcula luego de desencriptar el paquete que sería la variable **newhash** , si los dos valores son iguales entonces se imprime el mensaje “data intergity ensured” integridad de los datos asegurada,

caso contrario se imprime “data modified”, datos modificados por la tanto la integridad ha sido adulterada.

```
// show encryted data then decrytp it and show
 decryption(original_data);
 newhash=hashing(original_data,strlen(original_data));
 if(newhash==hdr->hashvalue)
 {
    printf("data intergity ensured\n");
    strcpy(authenticate_result,"Message_Accepted");
 }
 else
 {
    printf("data modified %d\n",newhash);
    strcpy(authenticate_result,"MESSAGE_ERRROR-Integrity violation");
```

Figura 90 Archivo security.cc

- Una vez creado los dos archivos se procede a modificar algunos archivos de configuración del simulador NS2 para que el nuevo agente de seguridad pueda comunicarse con el resto de funciones del NS2 , en primer lugar nos dirigimos al archivo Makefile.in ubicado en el directorio ns-allinone-2.35/ns-2.35 y se crea el objeto security.o como se muestra a continuación

```
OBJ_CC = \
        tools/random.o tools/rng.o tools/ranvar.o common/misc.o com
handler.o \
        common/scheduler.o common/object.o common/packet.o \
        common/ip.o routing/route.o common/connector.o common/ttl.o
        trace/trace.o trace/trace-ip.o \
        security/security.o \
        classifier/classifier.o classifier/classifier-addr.o \
```

Figura 91 Archivo Makefile.in

- A continuación se abre el archivo packet.h ubicado en el directorio ns-allinone-2.35/ns-2.35/common y se declara una variable para el paquete de seguridad asignando un numero al mismo y en el mismo archivo también se asigna un nombre al paquete de seguridad en este caso lo denominamos “Security_packet”

```
        // M-DART packets
static const packet_t PT_MDART = 72;

        // Security packet
static const packet_t PT_SECURITY_PACKET = 73;

        // insert new packet types here
static packet_t       PT_NTYPE = 74; // This MUST
```

```
name_[PT_CtrMcast_Decap]= "CtrMcast_Decap";
name_[PT_SRM]= "SRM";
name_[PT_SECURITY_PACKET]= "Security_packet";
name_[PT_REQUEST]= "sa_req";
name_[PT_ACCEPT]= "sa_accept";
```

Figura 92 Archivo packet.h

- Luego nos dirigimos al archivo ns-default.tcl ubicado en el directorio ns-allinone-2.35/ns-2.35/tcl/lib y agregamos el agente de seguridad que se encargará de implementar el mecanismo de seguridad en los paquetes y adicional se inicializa el tamaño de los paquetes en 0 bytes.

```
Agent/MDART set macFailed_ true
Agent/MDART set etxMetric_ true

Agent/Security_packet set packetSize_ 0
```

Figura 93 Archivo ns-default.tcl

- Y para finalizar la modificación de los archivos se abre el archivo ns-packet.tcl ubicado en el directorio ns-allinone-2.35/ns-2.35/tcl/lib y se agrega el paquete de seguridad a nivel de la capa de aplicación

```
# Application-Layer Protocols:
        Message # a protocol to carry text messages
        Ping    # Ping
    PBC     # PBC
        Security_packet
```

Figura 94 Archivo ns-packet.tcl

- Ahora se procede a recompilar el simulador NS2 para que los cambios tengan efecto, se abre una terminal y se ejecutan los siguientes comandos

```
user1@ubuntu: ~/ns-allinone-2.35/ns-2.35
user1@ubuntu:~/Desktop$ cd
user1@ubuntu:~$ cd ns-allinone-2.35
user1@ubuntu:~/ns-allinone-2.35$ cd ns-2.35
user1@ubuntu:~/ns-allinone-2.35/ns-2.35$ ./configure
```

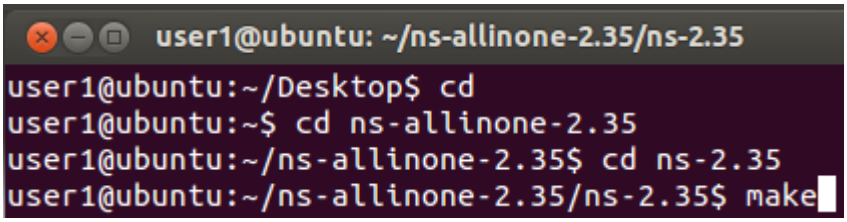


Figura 95 proceso de re-compilación

- Una vez finalizado el proceso de re-compilación procedemos a modificar el script tcl que es el que nos va a permitir crear el escenario de simulación, después de generar el trafico TCP para el intercambio de paquetes, se procede a llamar a la función del agente de seguridad en general (Agent/Security_packet) para que se conecte con el archivo security.cc y realice la encriptación de los mensajes.

```
Agent/Security_packet instproc recv {from rtt mess originmess hash} {
        $self instvar node_
        puts "node [$node_ id] received packet from \
              $from with trip-time $rtt ms - contend: $mess - decrypted
$originmess -hash: $hash"
}
```

Figura 96 Archivo tcl Escenario B Protocolo Saodv

- Después se generan agentes individuales que se asociaran a cada nodo como se muestra a continuación

```
set p0 [new Agent/Security_packet]
$ns_ attach-agent $node_(0) $p0
$p0 set class_ 1

set p1 [new Agent/Security_packet]
$ns_ attach-agent $node_(1) $p1
$p1 set class_ 1

set p2 [new Agent/Security_packet]
$ns_ attach-agent $node_(4) $p2
$p2 set class_ 2

set p3 [new Agent/Security_packet]
$ns_ attach-agent $node_(5) $p3
$p3 set class_ 2
```

Figura 97 Generación de agentes de seguridad

- Y finalmente se imprimen los mensajes de los test con las cadenas hash enviadas para proteger la integridad de los paquetes.

```
for {set i 1} {$i < 2} {incr i} {
        set result [expr $i /2]
        $ns_ at $result "$p0 send itisalongmessageIcansend"
        $ns_ at [expr $result + 0.02] "$p1 send  Itisashotermessage"
        $ns_ at [expr $result + 0.04] "$p2 send test3"
        $ns_ at [expr $result + 0.06] "$p3 send test4"
}
```

Figura 98 Impresión de mensajes

CAPÍTULO 4

4.1 Análisis de resultados y propuestas para mejorar el desempeño de la Red.

Como se indicó en capítulos anteriores los resultados totales de la simulación se obtiene a partir del archivo de traza, para obtener los resultados utilizamos dos herramientas que calcularán los diversos parámetros que tiene una red como son el throughput, delay, ratio, overhead, y así lograr finalmente realizar el estudio

comparativo de los dos protocolos involucrados en el presente proyecto AODV y SAODV.

4.1.1 Visual Trace Analizer

NS2 es una herramienta poderosa y una de las más usadas al momento se simular escenarios de redes, sin embargo no cuenta con ninguna aplicación propia del programa para interpretar los datos obtenidos de la simulación a través del archivo de traza, al Visual Trace Analyzer es un software libre de tipo ejecutable que nos permite calcular ciertos parámetros de la red y analizar el comportamiento de los flujos de la simulación de una manera sencilla permitiendo visualizar los resultados a través de gráficos ilustrativos , posee una interfaz sencilla de utilizar , es independiente, no necesita de bibliotecas externas ni instalación para utilizarla.

Para el presente proyecto se ejecutará la versión 0.2.72 en el ambiente de Windows, también está disponible para Linux.

Funcionamiento

A continuación se realiza un breve tutorial del funcionamiento de dicha herramienta:

- Al momento de hacer doble click en el ejecutable se abre la siguiente ventana, en donde se cargará en primer lugar el archivo tcl que contiene el escenario de la simulación , se da click en el cuadro que contiene 3 puntos para que busque el archivo que deseamos analizar

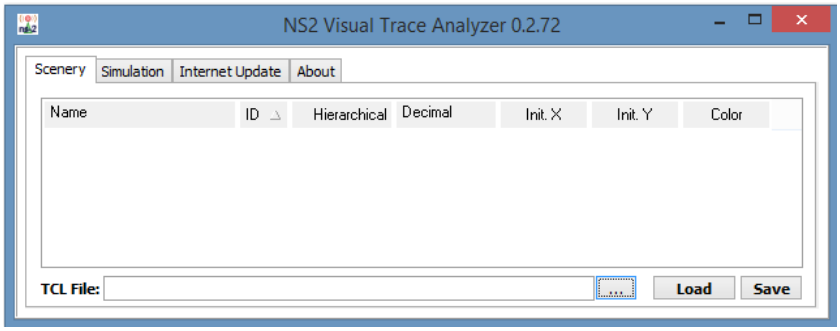


Figura 99 Entorno grafico del Visuak Trace Analyzer

A continuación se abre un cuadro con todos los archivos tcl , escogemos uno y click en abrir

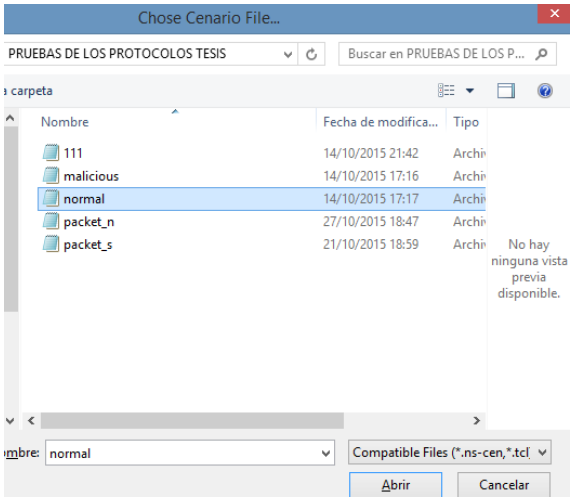


Figura 100 Proceso de carga de un archivo tcl

- Una vez que se carga el archivo tcl que se escogió se visualiza la topología con los nodos y las dimensiones del campo de simulación

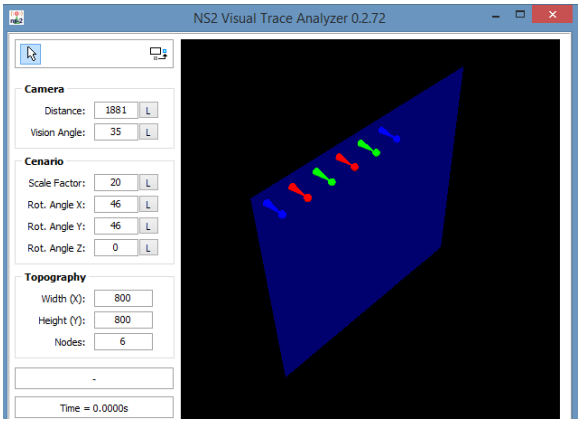


Figura 101 Nodos del escenario A

- Adicional se observa la lista de nodos con el identificador correspondiente y la posición exacta de cada nodo al momento de empezar la simulación

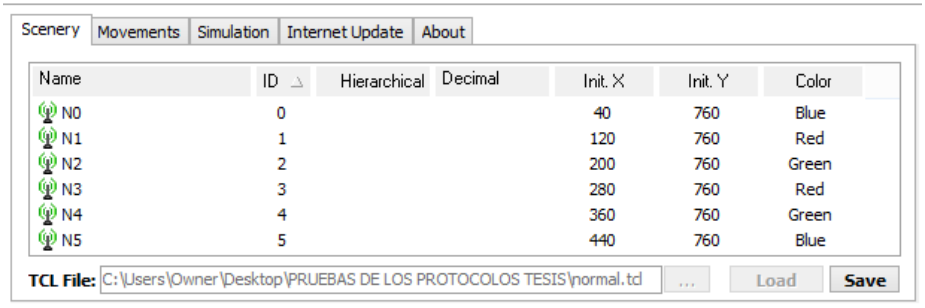


Figura 102 Coordenadas de los nodos

- Para observar el comportamiento de cada nodo y cada flujo en la simulación se procede a cargar el archivo de traza (.trace) , se dirige a la pestaña Simulation y se elige Cenario File y a continuación se carga el archivo de la misma forma que se cargó el archivo tcl

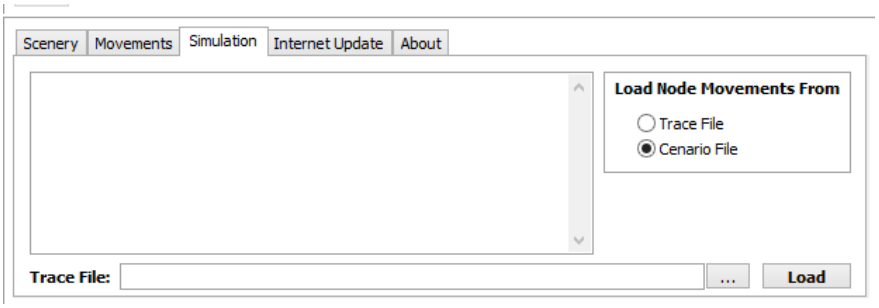


Figura 103 Proceso de carga de un archivo de traza

- Se escoge el archivo (.trace) correspondiente al archivo tcl que generamos anteriormente, en nuestro caso se elige el archivo normal.trace

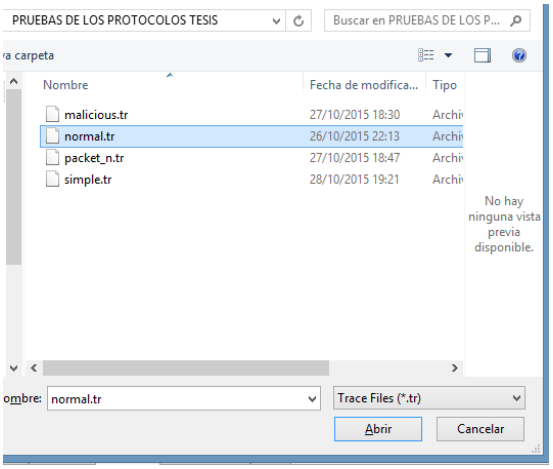


Figura 104 Proceso de carga de un archivo tcl

- Una vez que se carga el archivo de traza se puede visualizar el comportamiento de los nodos de la red de forma individual, como se observa en la imagen adjunta se puede visualizar de forma gráfica se pueden observar parámetros como delay, jitter, throughput.

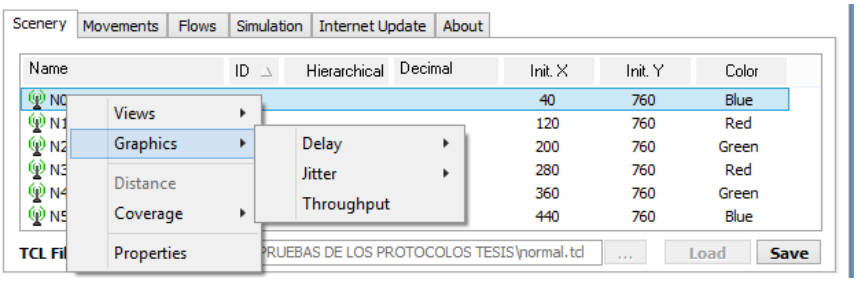


Figura 105 Parámetros de los nodos

- A continuación se visualiza el delay por paquete que existe en el nodo 1 calculado en ms durante toda la simulación , también se puede observar cual es el nodo origen y nodo destino

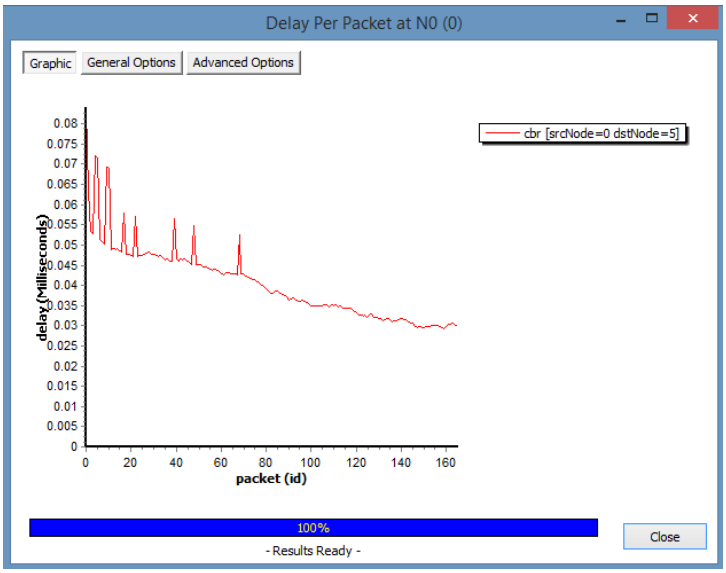


Figura 106 Delay del nodo 0

- En la siguiente imagen se observa informacion adicional del nodo 0 , tambien se visualiza los bytes generados en cada nivel, en el primer evento, ultimo evento, maximo radio de cobertura, paquetes perdidos.

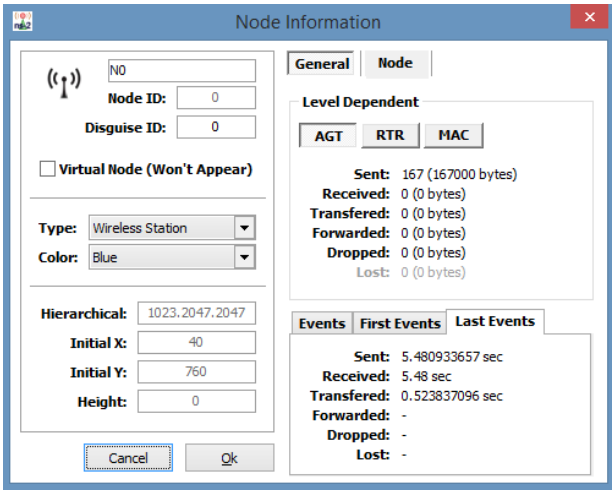


Figura 107 Información de nodos

- Otra de las funcionalidades nos permite visualizar las rutas posibles que se registraron para llegar al destino y el número de paquetes generados, descartados y transferidos

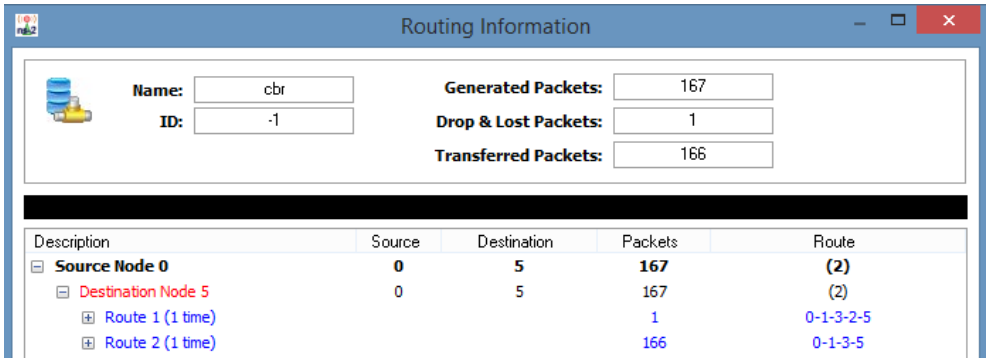


Figura 108 Entrega de paquetes y rutas

- Y finalmente la parte que interesa, el comportamiento del flujo de tráfico de toda la red durante toda la simulación, aquí se verifican parámetros como data en donde se visualiza los paquetes generados, descartados, también se verifica el delay mínimo, máximo y promedio de la red, de igual manera otros parámetros como el throughput, jitter y la conexión en donde se muestra el tiempo al que fue generado el primer paquete, el ultimo y el tiempo entre ambos

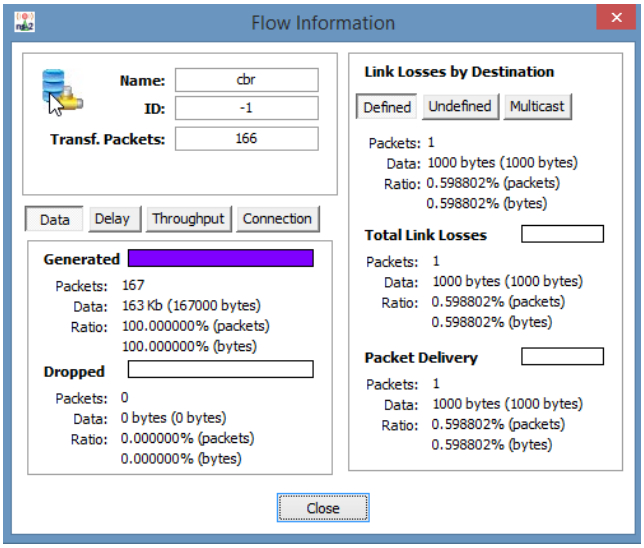


Figura 109 Parámetros de la red

4.1.2 SCRIPT AWK

Otra forma de interpretar los archivos de traza del NS2 es mediante los scripts AWK, dichos scripts son archivos escritos en lenguaje C que sirven para procesar los archivos de seguimiento manualmente, a continuación se muestra la estructura de un script AWK que consta de 3 partes:

- **Begin**

Es el comienzo del archivo, aquí se declaran las variables y constantes que se asignan con valores iniciales antes de ser procesadas en el contenido del archivo.

- **Contenido**

El contenido es donde se procesa el archivo de texto, aquí se programa el algoritmo que nos permite calcular los parámetros según sea el archivo.

- **Fin**

En esta parte se explica lo que se ejecuta después de haberse ejecutado el contenido, es decir se muestra lo que se imprimirá en el terminal con los resultados obtenidos.

A continuación se muestra la estructura del programa que calculará la sobre carga en la red, con la misma estructura cuentan el resto de programas.

- Como se indicó en el Begin se declara variables y constantes, aquí se muestra 2 variables que son el número de paquetes recibidos y el número de paquetes de enrutamiento.

```
BEGIN{
recvd = 0; received
rt_pkts = 0;
}
```

Figura 110 Contenido del begin

- Luego se muestra el contenido del script en donde se programa el algoritmo que calculará la sobrecarga en la red, se visualiza que se comparan paquetes CBR, TCP y paquetes a nivel AGT, también se observa que este tipo de script sirve para protocolos como AODV, AOMDV y OLSR.

```
{
if (( $1 == "r") && ( $7 == "cbr" || $7 =="tcp" ) && ( $4=="AGT" )) recvd++;

if (($1 == "s" || $1 == "f") && $4 == "RTR" && ($7 =="udp" || $7 == "AODV" || $7
=="message" || $7 =="AOMDV" || $7 =="OLSR")) rt_pkts++;

}
```

Figura 111 Contenido del script Awk

- Y para finalizar en la sección fin se imprime los resultados obtenidos, en este caso se observará número total de paquetes, número de paquetes enrutados, y el porcentaje de sobrecarga en la red.

```
END{
printf
("#############################################################################
\n");
printf("\n");
printf("total no of data packets\t%d\n",recvd);
printf("\ntotal no of routing packets\t%d\n",rt_pkts);
printf("                        Normalized Routing Load = %.3f\n", (rt_pkts/
recvd)*100);
printf("\n");
printf
```

Figura 112 Impresión de resultados

- Para utilizar los scripts AWK se debe instalar todos los paquetes necesarios para que el sistema reconozca los comandos.

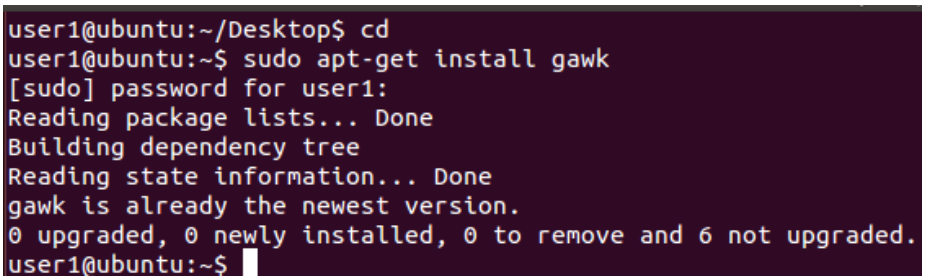


Figura 113 Paquetes para utilizar los archivos Awk

- Para ejecutar los scripts AWK se lo realiza mediante un terminal, se ubica en la carpeta donde se encuentra el archivo de traza y el archivo AWK y se ejecuta el comando que se muestra en la imagen a continuación y se muestra el resultado del porcentaje de sobrecarga en la red, el mismo proceso se sigue para el resto de archivos simplemente se cambia el nombre del script que se quiere analizar.

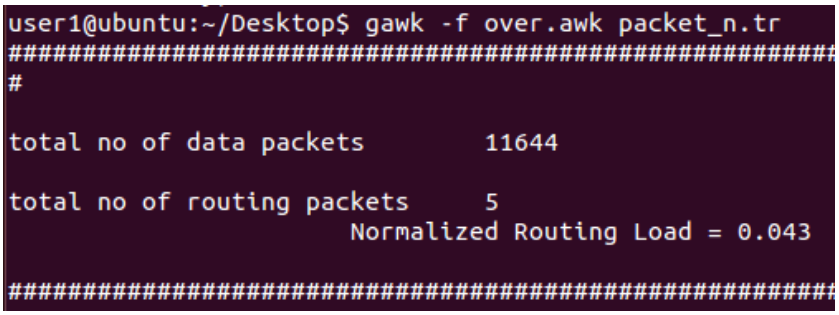


Figura 114 Entrega de resultados mediante el terminal

En este proyecto se compararán los resultados de algunos parámetros encontrados tanto con el programa Visual Trace Analyzer como con los scripts AWK para confirmar que los resultados sean precisos y confiables.

4.2 Análisis de Resultados

4.2.1 Escenario A protocolo AODV

Delay

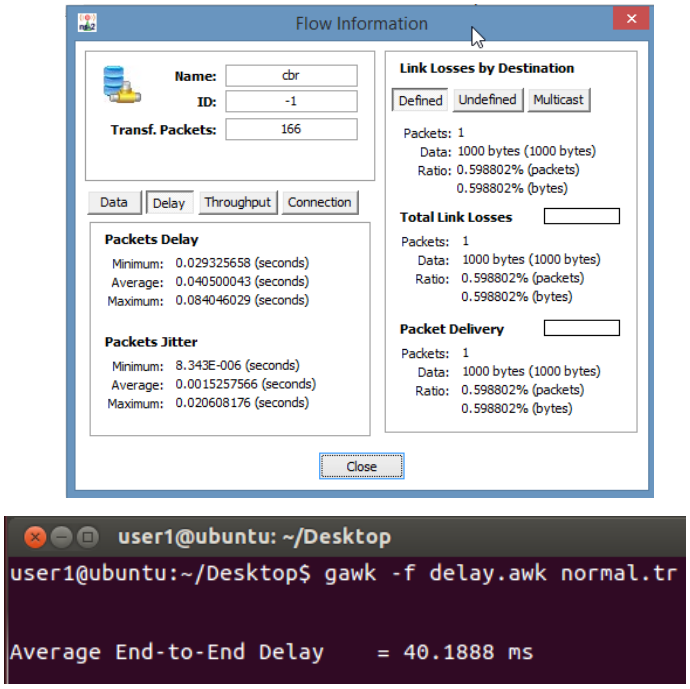


Figura 115 Delay del Escenario A protocolo Aodv

Througputh

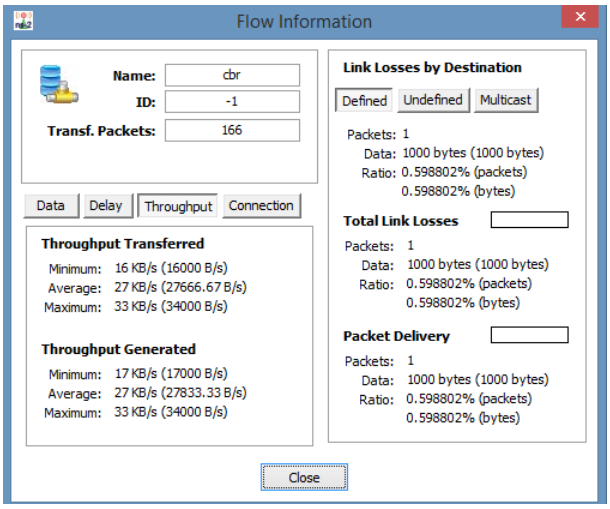


Figura 116 Througputh del Escenario A protocolo Aodv

Ratio

```
user1@ubuntu:~/Desktop$ gawk -f ratio.awk normal.tr
s:167 r:166, r/s Ratio:99.4012, f:336
```

Scenery | Movements | Flows | Simulation | Internet Update | About

Name	ID	Gen. Packets	Gen. Bytes	T. Lost Packets	T. Lost Bytes
cbr	-1	167	163 Kb	1	1000 bytes

Figura 117 Ratio del Escenario A protocolo Aodv

Over

```
user1@ubuntu:~/Desktop$ gawk -f over.awk normal.tr
############################################################
##

total no of data packets        166

total no of routing packets     8
                        Normalized Routing Load = 0.048

############################################################
##
```

Figura 118 Sobrecarga del Escenario A protocolo Aodv

Figura 105

4.2.2 Escenario A protocolo AODV

Delay

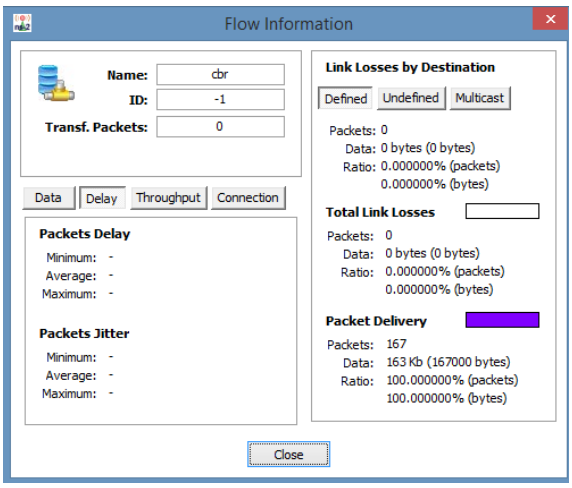


Figura 119 Delay del Escenario A protocolo Saodv

Througputh

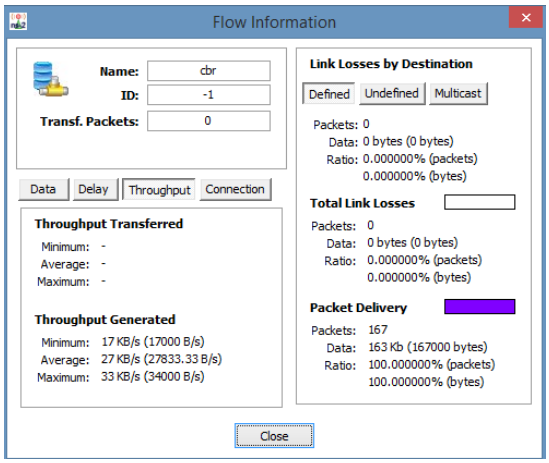


Figura 120 Througputh del Escenario A protocolo Saodv

Ratio

```
user1@ubuntu:~/Desktop$ gawk -f ratio.awk malicious.tr
s:167 r:0, r/s Ratio:0.0000, f:2
```

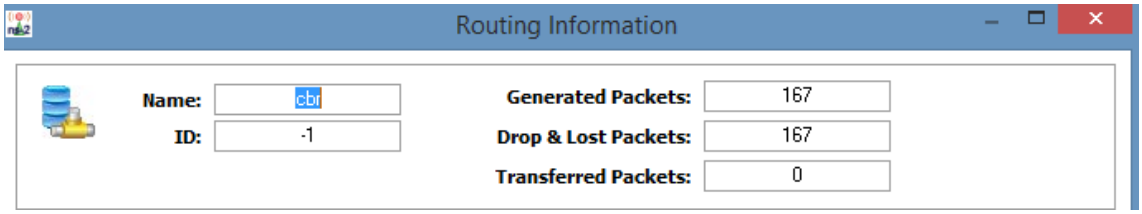


Figura 121 Ratio del Escenario A protocolo Saodv

4.2.3 Estudio Comparativo

Delay

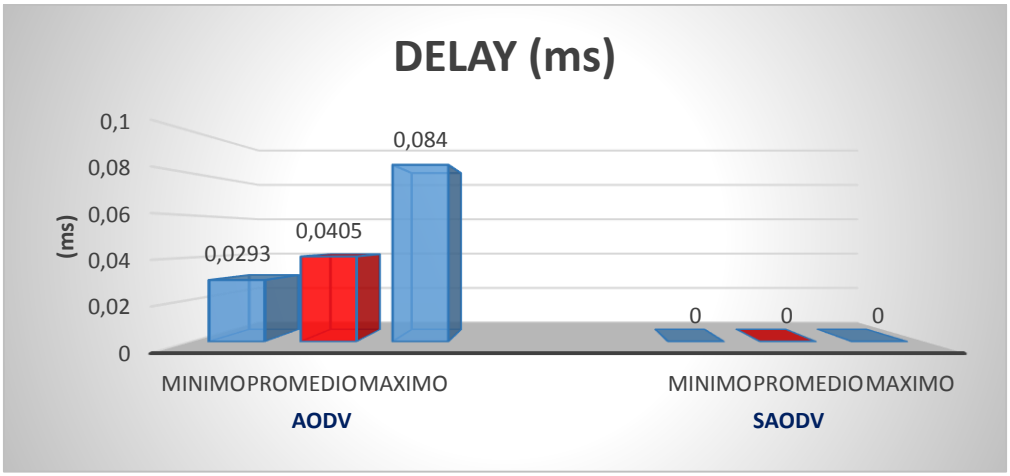


Figura 122 Comparación de Delay Escenario A

El retardo o delay es un parametro muy importante que se debe calcular tener en claro como funciona una red , el delay no es mas que el tiempo que tardan los paquetes en trasladarse desde el origen hacia el destino , dicho parametro puede variar dependiendo de muchos factores como el numero de componentes en la red, la velocidad de procesamiento , la capacidad de almacenamiento , radio de cobertura entre otros, cabe destacar que todos los parametros serán mayores en una red inalambrica a diferencia de una red cableada, ya que al ser un medio compartido existen factores que retrasan la transmision de paquetes.

Como se puede visualizar en la imagen anterior en el primer evento en condiciones normales el retardo es de 0.0405 ms, este tiempo es el promedio que se demoran todos los paquetes en llegar desde el origen hasta el destino, en cambio en el segundo evento el delay es 0 ya que los datos nunca llegarán al destino, siempre serán descartados por el nodo malicioso es por eso que no existe ningun retardo en la red.

Throughput

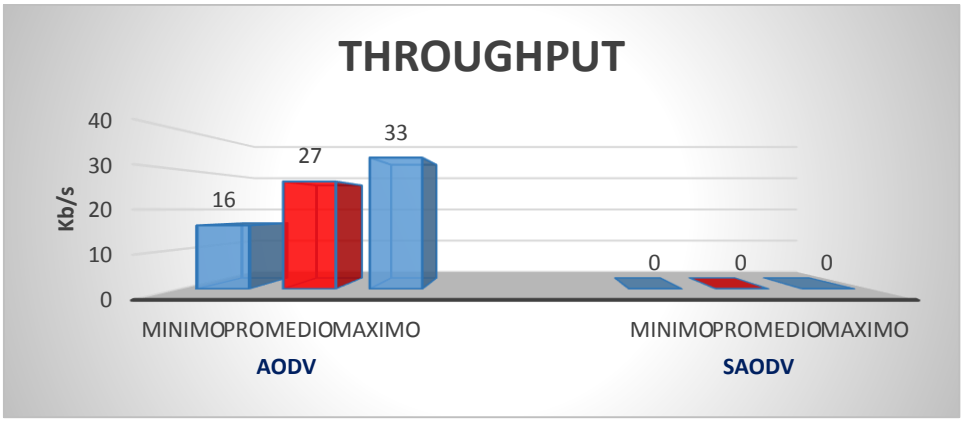


Figura 123 Comparación de Throughput Escenario A

El Throughput es la cantidad exacta de datos que fluyen en una red en un determinado tiempo, se conoce también como la tasa promedio de éxito al transmitir un mensaje, este parámetro al igual que el delay también puede variar dependiendo de ciertos factores, en el caso de una red inalámbrica al ser un medio compartido se pueden perder paquetes y eso afectaría al rendimiento de la red.

En el primer evento se registra un throughput de 27 Kb/s, que sería el promedio de la capacidad de la red para transmitir datos exactos de información, en el segundo evento no se genera ningún tipo de throughput debido a que los paquetes serán descartados al momento de detectar el nodo malicioso y los nodos no realizarán ningún tipo de transferencia

Entrega de Paquetes

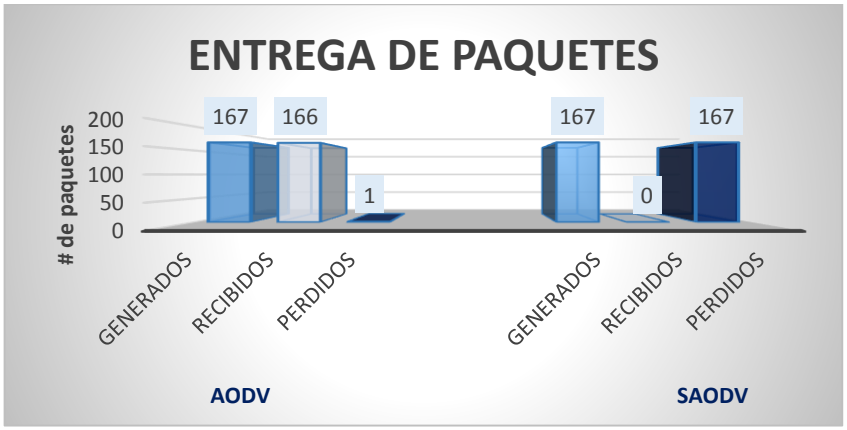


Figura 124 Comparación de Entrega de paquetes Escenario A

La entrega de paquetes nos indicará la confiabilidad y disponibilidad de una red , en el primer evento se visualiza que existen 167 paquetes generados de los cuales solo existe 1 paquete perdido , esto es normal ya que el protocolo AODV es uno de los protocolos más estables y confiables en el ámbito de entrega de paquetes en una red inalámbrica comparados con otros protocolos de encaminamiento, los algoritmos de enrutamiento de paquetes que utiliza el protocolo AODV evitan que se pierdan el menor número de paquetes al momento de transmitir, en el caso del segundo evento el protocolo SAODV detecta al nodo malicioso y hace que todos los paquetes sean descartados y que ninguno llegue al destino.

Sobrecarga

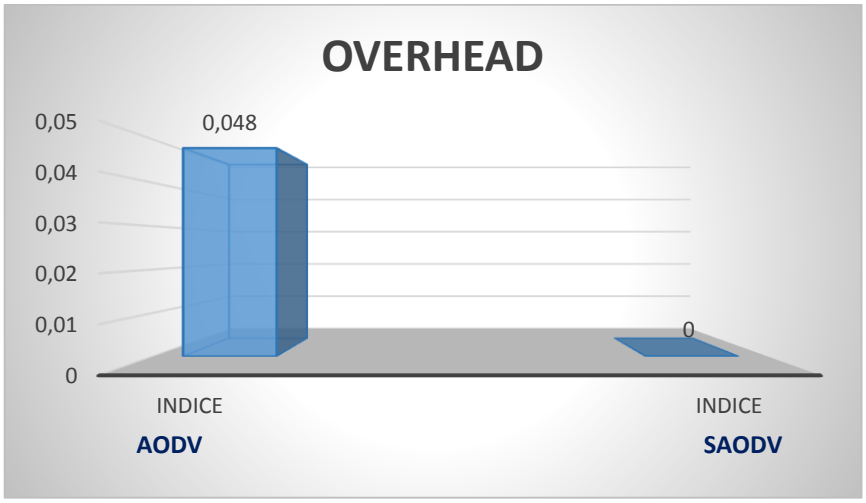


Figura 125 Comparación de Sobrecarga Escenario A

El overhead o sobrecarga son los bytes adicionales de un paquete, estos bytes no son parte de la información que se envía en un paquete pero son los bytes de control que cumplen funciones como seguridad, direccionamiento, administración de tráfico, control de secuencia , entre otros, este parámetro afecta directamente el throughput y ancho de banda de una red .

En el primer evento se calcula un overhead de 0.048 ya que el protocolo AODV si cuenta con información adicional de control en los paquetes para poder transmitir, en el segundo evento no existe ningún tipo de sobrecarga ya que el nodo malicioso al momento de descartar los paquetes no interviene los bytes de control.

4.2.4 Resultados del escenario B protocolo AODV

Delay

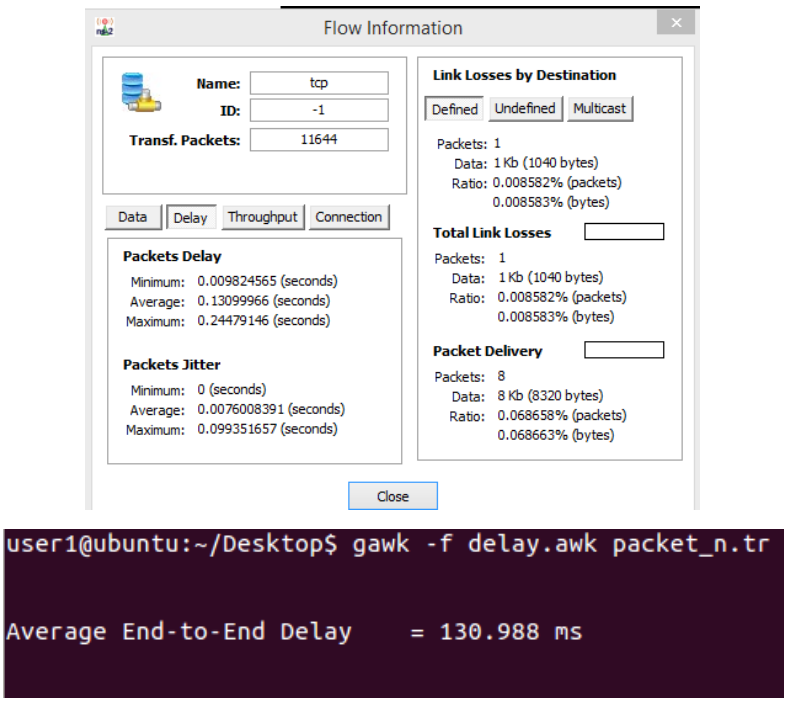


Figura 126 Delay del Escenario B protocolo Aodv

Throughput

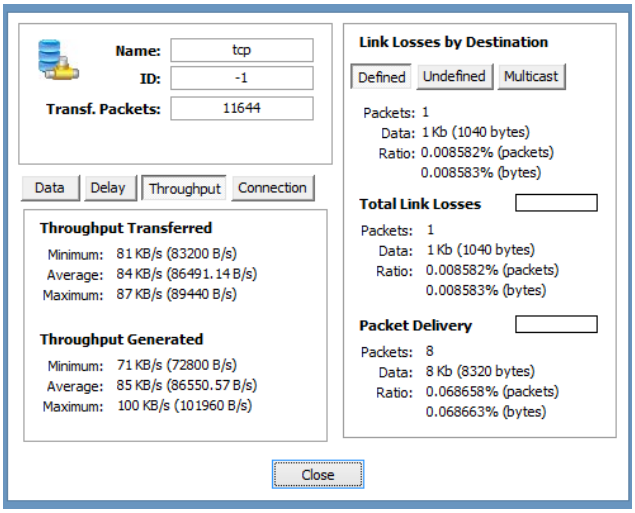


Figura 127 Throughput del Escenario B protocolo Aodv

Ratio

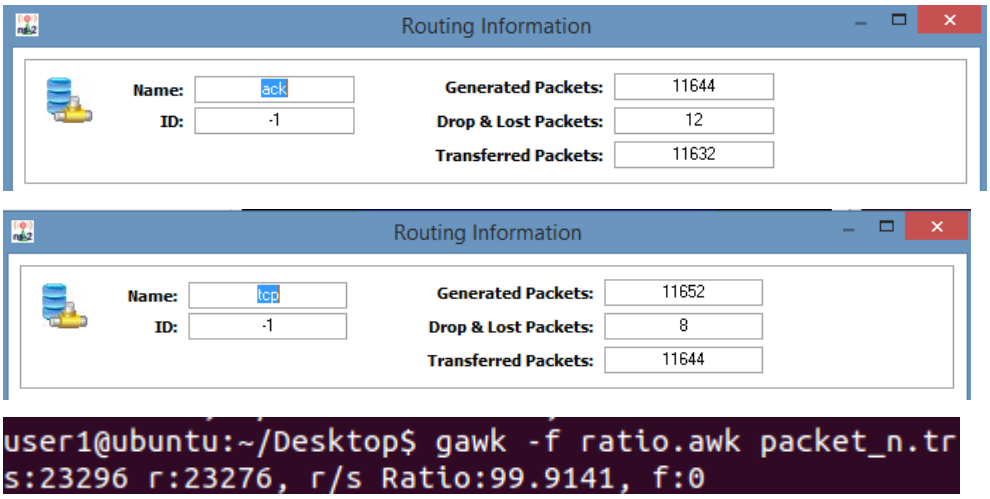


Figura 128 Ratio del Escenario B protocolo Aodv

Over

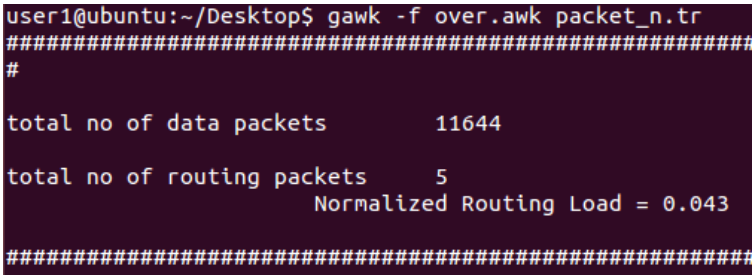
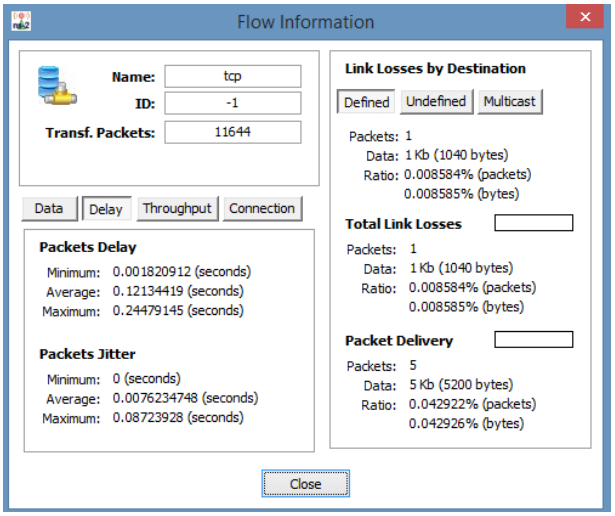


Figura 129 Sobrecarga del Escenario B protocolo Aodv

4.2.5 Resultados del escenario B protocolo SAODV

Delay




```
user1@ubuntu:~/Desktop$ gawk -f delay.awk simple.trace

Average End-to-End Delay     = 121.334 ms
```

Figura 130 Delay del Escenario B protocolo Saodv

Thropugput

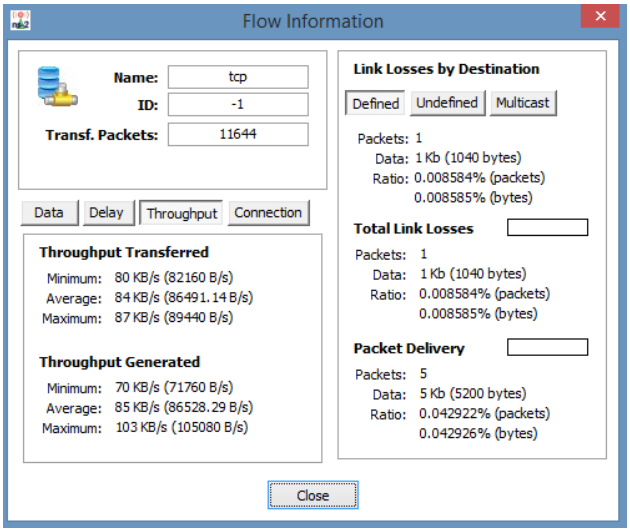


Figura 131 Thropugput del Escenario B protocolo Saodv

Over

```
user1@ubuntu:~/Desktop$ gawk -f over.awk simple.tr
###############################################################
#

total no of data packets        11644

total no of routing packets     29
                        Normalized Routing Load = 0.249

###############################################################
```

Figura 132 Sobrecarga del Escenario B protocolo Saodv

Ratio

Name	ID	Gen. Packets	Gen. Bytes	T. Lost Packets	T. Lost Bytes
tcp	-1	11649	12 Mb	5	5 Kb
ack	-1	11644	455 Kb	15	600 bytes
Security_packet	0	6	0 bytes	3	0 bytes

```
user1@ubuntu:~/Desktop$ gawk -f ratio.awk simple.tr
s:23299 r:23277, r/s Ratio:99.9056, f:0
```

Figura 133 Ratio del Escenario B protocolo Saodv

4.2.6 Estudio Comparativo

Delay

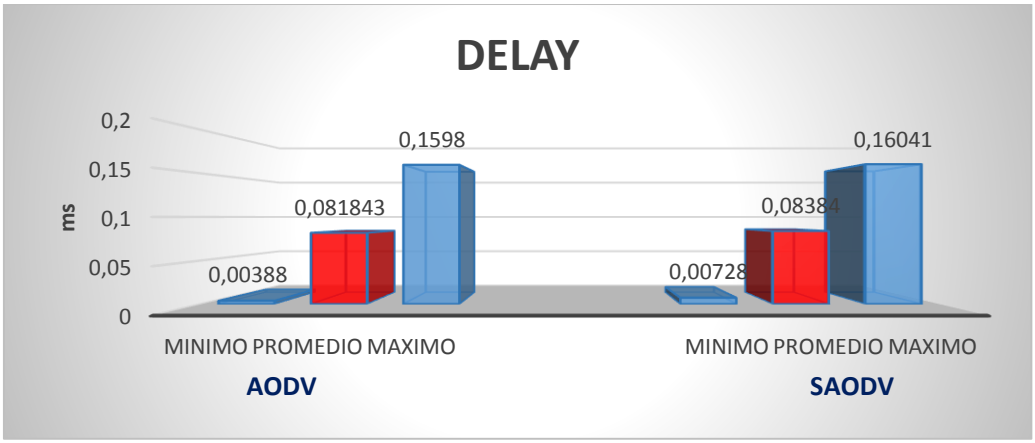


Figura 134 Comparación de Delay Escenario B

En el segundo escenario ya se visualizará información de los parámetros ya que aquí se implementa la seguridad de los paquetes al momento de ser transmitidos y llegarán

hasta el destino, el primer parámetro que se analizará el delay o retardo, en la gráfica se observa que el segundo evento cuenta ligeramente con un poco más de retardo esto debido a que los mensajes que van encriptados desde el origen tienen que ser desencriptados y verificados por el destino mediante algoritmos explicados en el capítulo anterior, esto implica que los mensajes tarden un tiempo adicional en ser transmitidos.

Throughput

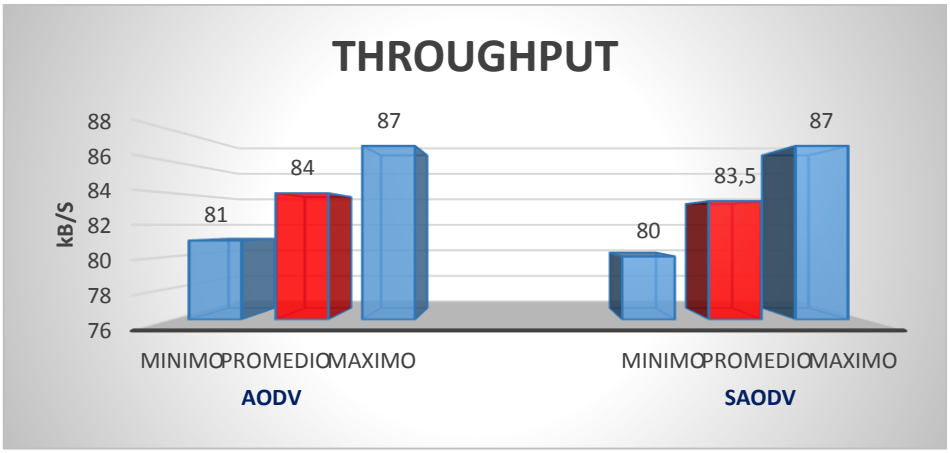


Figura 135 Comparación de Throughput Escenario B

En el caso del throughput para el primer evento y para el segundo evento son casi similares, esto significa que en el caso del protocolo AODV la cantidad de datos reales de información que se transmitan en el flujo de la red serán un poco mayor que el protocolo SAODV, sin embargo no afectaría el rendimiento de la red ya que en los resultados no se visualiza una diferencia considerable.

Entrega de Paquetes

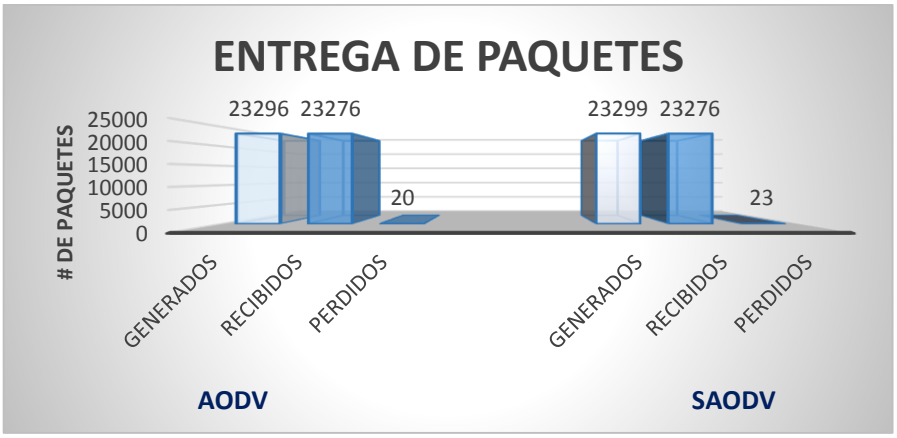


Figura 136 Comparación de Entrega de paquetes Escenario

La entrega de paquetes en ambos protocolo es casi similar, en el protocolo AODV del total de 23296 paquetes generados apenas se pierden 20 y en el caso del protocolo SAODV se pierden 23 que no es una diferencia significativa, eso es debido a que el protocolo SAODV solo aplica mecanismos de seguridad al protocolo AODV sin cambiar el algoritmo de enrutamiento que el protocolo AODV utiliza, por eso evitará que se pierdan el menor número de paquetes en la transmisión.

Sobrecarga

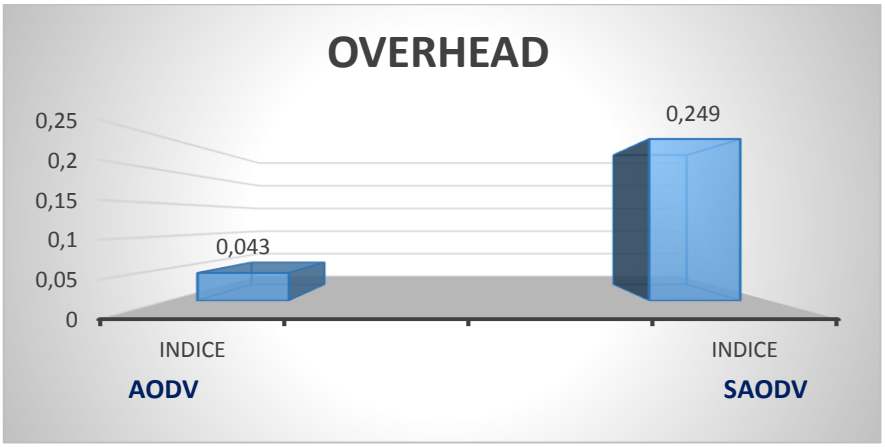


Figura 137 Comparación de Sobrecarga Escenario B

El overhead o sobrecarga como se indicó en el escenario anterior son los bytes de control adicionales aparte de los bytes de información que viaja en un paquete , bytes de control de seguridad en el caso del protocolo SAODV, la sobrecarga será mayor en el protocolo que en el protocolo AODV, ya que además de contar con los bytes de control propias del protocolo AODV también se adicionan los bytes de seguridad

propios del protocolo SAODV para verificar la integridad de un paquete, esto también afectaría el nivel de procesamiento de los equipos de una red.

4.3 Propuestas para mejorar el desempeño de la Red Ad Hoc.

Si bien utilizar una red Ad-Hoc no es muy común en aplicaciones en la actualidad, se sabe que en algunos años este tipo de redes inalámbricas lograrán un despunte significativo , si se habla de una red Ad-Hoc el protocolo AODV es el más eficiente en todo aspecto del funcionamiento una red Ad-Hoc ya que es el mejoramiento de protocolos como DSR y este a su vez mejoramiento de otros protocolos, sin embargo hay un aspecto importante que dicho protocolo no implementa la seguridad, en una red inalámbrica en donde el medio es compartido se puede sufrir diversos ataques , lastimosamente no existe un protocolo registrado que brinde una completa seguridad a una red inalámbrica y es en este aspecto que se debe trabajar para convertir a una red Ad-Hoc en un tipo de redes confiable y empezar a implementar aplicaciones en el mundo real.

El enrutamiento de paquetes en una red inalámbrica es eficiente gracias al protocolo AODV, en donde no existe demasiada sobrecarga, tampoco hay perdida de paquetes de manera significativa, el retardo en la entrega de paquetes es bajo, el único aspecto que se debe trabajar es el de la seguridad, aquí entra el protocolo SAODV que si bien es cierto realiza un gran trabajo brindando seguridad a los mensajes que transmite el protocolo AODV, no cubre todos los aspectos de seguridad que pueden ser vulnerados en una red como por ejemplo ataques por denegación de servicio que se sufre en la capa física por lo que se debe tener en cuenta este aspecto para trabajos futuros.

Al implementar algoritmos de seguridad se sufre cierta sobrecarga y aumenta el nivel de procesamiento en los equipos por lo que es indispensable contar con equipos con nivel de procesamiento óptimo para no causar retardo en el envío de paquetes, también es importante implementar algoritmos de consumo de energía en los nodos cuando estos se encuentran sin actividad (modo sleep) ya que al momento de poner en funcionamiento la red deben estar al 100% para un óptimo funcionamiento.

Cuando se desee implementar un algoritmo para detección de nodos que no pertenezcan a la red es recomendable que una red cuente con un nodo administrador

o nodo maestro que proporcione números de identificación a los nodos que deseen ingresar a la red Ad-Hoc, en el caso que estén inactivos un cierto tiempo y deseen ingresar a la red deben acudir con el nodo maestro con la identificación respectiva que se le había otorgado y así poder ingresar nuevamente.

CAPÍTULO 5

5.1 Conclusiones

- Se debe aclarar que el protocolo SAODV no soluciona completamente el problema de seguridad del protocolo AODV ya que pueden existir ataques en la capa física en donde el protocolo no puede actuar , en lo único que nos ayuda el protocolo SAODV es en implementar confidencialidad e integridad a los mensajes AODV.
- Con la implementación del algoritmo de detención de intrusos utilizado en el escenario A del protocolo SAODV es complicado que un nodo malicioso quiera involucrarse en la red ya que los nodos cuentan con un número de identificación que solo saben los verdaderos integrantes de la red, al momento de realizar el proceso de descubrimiento de ruta un nodo malicioso puede enviar mensajes REQ al resto, pero mensajes con información de tipo TCP serán descartados y no llegarán al nodo destino.
- Al momento de analizar el throughput en el escenario B en donde se implementa seguridad a los mensajes, se verifica que los dos protocolos trabajan casi de manear similar transmitiendo la misma cantidad de información en determinado tiempo, así que en cuestión de rendimiento ambos protocolos nos brindan similares características.
- En el caso de la tasa de entrega de paquetes en el escenario B, se verifica que ambos protocolos intentar evitar que se pierdan la menor cantidad de paquetes en una transmisión, los dos utilizan un mismo principio al momento de realizar el enrutamiento de los paquetes lo que garantiza que la mayoría de paquetes transmitidos llegarán a su destino, aunque el protocolo SAODV tardará un poco más en su entrega, lo transportará de una manera segura, característica que el protocolo AODV no aplica.
- En el escenario A en donde se encuentra el nodo malicioso se visualiza que en la red no se produce retardo , ni throughput, ni sobrecarga ya que al momento de detectar la presencia del nodo malicioso el protocolo SAODV descartará

todos los paquetes que lleguen al nodo malicioso impidiendo que lleguen hasta el destino.

- En términos generales el protocolo SAODV es un protocolo eficiente al momento de brindar seguridad a una red Ad-hoc, posee cambios en ciertos parámetros comparados con el protocolo AODV como un ligero aumento en el retardo y sobrecarga, sin embargo no son cambios significativos que afecten el funcionamiento general de la red, tomando en cuenta que el protocolo SAODV brinda un aspecto muy importante en toda clase de red que es la seguridad.

5.2 Recomendaciones

- Debido al diferente comportamiento de los protocolos principalmente en el ámbito de la seguridad es necesario considerar el escenario en el que se va a desempeñar una red Ad-Hoc, el protocolo AODV debido a que no implementa seguridad es idóneo para ser implementado en campos en donde se asume que todos los nodos son vecinos y son confiables por ende la seguridad no será imprescindible en este tipo de escenarios, en cambio el protocolo SAODV puede ser implementado en escenarios de campo libre en donde no se conoce mucha información de los nodos participantes de la red y no existe mucha confianza en los mismos.
- Se recomienda poner énfasis en la capa física del protocolo AODV, el protocolo SAODV no puede detectar ataques a nivel de dicha capa, por lo que se debe crear algún algoritmo para proteger la red ante cualquier ataque.
- La herramienta de simulación NS2 es un software que ofrece multitud de prestaciones al momento de simular una red, y se acerca mucho al ámbito real del funcionamiento de una red, es el simulador más utilizado en redes Ad-Hoc, para poder aprovechar al máximo esta herramienta es necesario conocer su funcionamiento, su arquitectura y como se enlazan los dos lenguajes tanto el lenguaje C y el Tcl para comprender de mejor manera la creación de escenarios e interpretar correctamente los resultados que se obtienen en una simulación .

- Para realizar el estudio comparativo se utilizó un software ejecutable que nos entrega los parámetros de la red, y adicional se utilizaron archivos AWK con el fin de verificar y comprobar que los resultados que nos entregaba el funcionamiento de la red sean confiables, se recomienda utilizar más de un método para el análisis de resultados.
- Es estrictamente necesario entender los archivos de traza que generan las simulaciones ya que en este archivo se encuentra la información completa del funcionamiento de cualquier protocolo, desde los campos de la cabecera, algoritmos de enrutamiento, algoritmos de seguridad, hasta enviar los datos de información entre nodos, a partir de este archivo se puede obtener cualquier parámetro de la red por lo que debe ser clara su interpretación.

BIBLIOGRAFIA

Peral, Alberto. *Estudio del rendimiento y la seguridaden redes ad hoc.*. Recuperado el 1 de abril de 2015 http://eprints.ucm.es/8934/1/Estudio_del_rendimiento_y_la_seguridad_en_redes_ad_hoc.pdf

Calderon, Oscar,*Seguridad en redes ad hoc* http://www.unipamplona.edu.co/unipamplona/portalIG/home_40/recursos/01_general/revista_3/13102011/08.pdf

Clep, M. (2008). *MKE Solutions.* Recuperado el 1 de abril de 2015, de http://www.mkesolutions.net/articulos/conceptos/el-protocolo-nv2-nstreme-ver-2/

Hidalgo, Francisco. (2008). *Viabilidad de la utilizacion de redes inalambricas ad hoc* . Chichester, Inglaterra: John Wiley & Sons, Ltd https://riunet.upv.es/bitstream/handle/10251/13183/TesinaMaster_FcoJavierHidalgo.pdf?sequnce=1.

Herrera.Miguel (2004). Network *Simulator.* Recuperado el 1 de abril de 2015, de http://www.inf.utfsm.cl/~jherrera/docs/mios/ns.pdf

Bonastre.Oscar .*Introduccion a la programacion de protocolos de comunicacionescon* Network *Simulator.* Recuperado el 1 de abril de 2015, de http://www.editorial-club-universitario.es/pdf/498.pdf

Gil.María Elena (2009). *Estudio de la eficiencia de encaminamiento del protocolo AODV en redes ad hoc inal´ambricas de gran escala.* Recuperado el 1 de abril de 2015, de

https://eciencia.urjc.es/bitstream/handle/10115/2546/PFC_MariaElenaGilJimenez.pdf;jsessionid=6238932CE1DB75C9F85CDE3B0BE1749F?sequence=1

Yachirema, Diana (2004). *Comparativa de los protocolos Aoddv y Olsr en redes moviles ad hoc.* Recuperado el 1 de abril de 2015, de http://es.slideshare.net/cisoft/comparativa-de-los-protocolos-aodv-y-olsr-en-redes-moviles-ad-hoc-manet

Ros, Francisco (2004). *Evaluacion de Propuestas de Interconexion a Internet para Redes Moviles Ad Hoc H´ıbridas*
. Recuperado el 1 de abril de 2015, de http://www.inf.utfsm.cl/~jherrera/docs/mios/ns.pdf