VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"IDENTIFICADOR INTELIGENTE DE INCIDENTES POR MEDIO DE UN IDS BASADO EN ANOMALÍAS UTILIZANDO LA PLATAFORMA NUMENTA"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN INGENIERÍA (COMPUTACIÓN)

P R E S E N T A:

ROSA XOCHITL SARABIA BAUTISTA

DIRECTOR DE TESIS:

DR. ENRIQUE DALTABUIT GODAS

México, D.F. **2012.**

Resumen

La seguridad en la red se vuelve cada vez más importante con el rápido desarrollo de las tecnologías. Conforme surgen las tecnologías de protección, los intrusos encuentran la manera de evitarlas. La denegación de servicio (DoS) es uno de los métodos de intrusión más populares que a menudo ocasiona grandes consecuencias y pérdidas económicas.

Por lo tanto, existe una necesidad de garantizar la seguridad mediante el uso de cortafuegos, sistema de detección de intrusos (IDS), cifrado, autenticación y otras soluciones de hardware y software. Un IDS automatiza el proceso de detección de intrusos y detecta posibles intrusiones a través de la identificación de comportamientos maliciosos que se dirigen a una red y sus recursos.

El propósito de esta investigación es proporcionar una descripción de la experiencia en el uso de esta tecnología, a través del desarrollo de una solución para la identificación de incidentes de seguridad de la información. Para ello, se implementó un sistema de detección de intrusos basado en anomalías utilizando una novedosa técnica de inteligencia artificial conocida como Memoria Temporal Jerárquica (HTM, por sus siglas en inglés), que reproduce las propiedades estructurales y algorítmicas del neocórtex, la parte del cerebro humano responsable de las funciones cognitivas.

Mediante el entrenamiento de patrones sensoriales variantes en el tiempo obtenidos del tráfico TCP/IP de la red de datos, se construyó un modelo espacio-temporal del mundo en HTM, que se evaluó con paquetes de red que contenían información de ataques de escaneo de puertos y denegación de servicio. Debido a sus características, el IDS detectó a estos ataques como anomalías.

En esta tesis se incluye una metodología desarrollada para la identificación de incidentes de seguridad a partir de las anomalías creadas. Así, se obtiene el número de paquetes asociados a la categoría, el tiempo delta, direcciones IP, números de puerto, tamaño de las ventanas y demás información relacionada con los ataques para poder determinar su causa.

Abstract

The network security becomes more and more important with the rapid development of network technology and application. As new protection technologies emerge, intruders find ways to avoid them. The Denial of Service (DoS) attack is one of the most popular intrusion methods which often make great economic losses and impact.

So, there exists a need to provide security through the use of firewall, Intrusion Detection Systems (IDSs), encryption, authentication, and other hardware and software solutions. An IDS automates the intrusion detection process and detects possible intrusions by identifying malicious behavior that targets a network and its resources.

The purpose of this research is to provide a description of the experience using this technology, through the development of a solution to identify information security incidents. To do this we implemented an anomaly-based Intrusion Detection System using a novel artificial intelligence technique known as Hierarchical Temporal Memory (HTM) that replicates the structural and algorithmic properties of the neocortex, the part of the brain responsible for cognitive functionalities.

By training on time-varying sensory patterns obtained from TCP/IP data network traffic, the HTM built a spatial and temporal model of the world which was evaluated using network packets containing information of port scan and denial of service attacks. Due to its characteristics, these attacks were detected by the IDS as anomalies.

This thesis includes a methodology developed for the identification of computer security incidents from the anomalies created. Thus, we obtain the number of packets associated with the category, delta time, IP addresses, port numbers, windows size and flags involved in the attacks to determine its cause.

Agradecimientos

A *mis padres*, María y Pedro, quienes son todo para mí. Mamá, tú me apoyaste en todo momento, me entendiste y me enseñaste a ser fuerte, decidida y ser todo lo que hoy soy, no tengo palabras para agradecer tu existencia, simplemente eres lo mejor que me ha pasado en la vida. Papá, gracias por cuidarme, quererme y permitirme terminar mis estudios y siempre, aunque dormido, acompañarme todas las mañanas.

A *mi hermana*, Lidia, que si no hubiera sido por ella no estaría aquí, que a pesar de sus regaños y llamadas de atención me enseñó a luchar siempre por cumplir mis metas y seguir adelante, de quien me siento orgullosa y quien es un ejemplo a seguir.

A *mis hermanos*, César y Juan, con quienes he podido compartir una vida llena de alegrías, de buenos y malos momentos.

A ti, *Christian*, por apoyarme siempre, comprenderme y hasta soportarme en los momentos más difíciles. Gracias por estar a mi lado incondicionalmente y por ser tan especial. No sé qué haría sin ti... Atch!

A *mis amigos* Javier, Sergio, Julio, que confiaron en mí, por darme su sincera amistad y no pedir nada a cambio, bueno… tal vez dulces.

A *mis profesores* por contribuir fuertemente en mi educación, por enseñarme que un número no refleja el conocimiento adquirido. Principalmente al Dr. Enrique Daltabuit Godas, quien con su apoyo y paciencia es pieza fundamental de este logro.

A *mi amada universidad*, que es como mi segunda casa y que lejos de ella recuerdo todos los buenos momentos que pasé. La máxima casa de estudios a quien le debo todo lo que he aprendido y los éxitos que he obtenido. A quien me comprometo representarla con orgullo y poner en alto su nombre.

La vida sigue y aún es largo el camino, me faltan muchas metas por cumplir, sueños que realizar, y todo lo que he aprendido lo aplicaré para ser mejor…

Como no te voy a querer si mi corazón es azul y mi piel dorada siempre te amaré…
¡Por mi raza hablará el espíritu!

Contenido

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

En este capítulo se incluirá una breve descripción de la relevancia y contribución de este trabajo de tesis, se plantearán los objetivos del proyecto, el alcance y la estructura del mismo.

1.1 Antecedentes

"Si piensas que la tecnología puede solucionar tus problemas de seguridad, está claro que ni entiendes los problemas ni entiendes la tecnología".

Bruce Schneier

Los ataques cibernéticos pueden no ser un fenómeno nuevo, pero los recientes ataques exitosos contra objetivos de alto impacto como Citigroup, Google, RSA, Sony y agencias gubernamentales, acentúan el fracaso de los objetivos actuales de bloquear las amenazas de seguridad en Internet.

El costo de una sola falla de seguridad puede ser muy alto para una organización. Ponemon Institute estima que va desde $1 millón a $58 millones de dólares. Sin embargo, el costo no sólo es financiero, daño a la reputación y pérdida de clientes son posibles efectos secundarios que puede representar un incidente de seguridad [1].

Recientemente, los ataques en Internet se han convertido en la amenaza principal de la seguridad de la información para las organizaciones. Los ciberdelincuentes constantemente lanzan ataques diseñados para penetrar sus defensas digitales y robar datos sensibles. De acuerdo con los estudios realizados por Gartner: “En el 2009 el 80 por ciento de las compañías sufrieron un incidente de seguridad en aplicaciones que se encontraban en Internet”.

Según el último informe de Symantec [2] “Informe sobre Amenazas a la Seguridad en Internet”, en el 2010 se registraron más de 286 millones de nuevas amenazas. Asimismo, el informe destaca incrementos importantes, tanto en frecuencia como en sofisticación de los ataques dirigidos a organizaciones, el constante crecimiento de sitios de redes sociales como plataformas de distribución de ataques y un cambio en las técnicas de intrusión.

En países como México, económicamente activos y relevantes en lo que respecta al escenario latinoamericano[1], los ataques están a la orden del día; ya que es justamente por estas características en cuanto a poderío económico o cantidad de habitantes que los atacantes ponen el foco en este tipo de países.

A pesar de que se han observado importantes avances en materia de seguridad en las organizaciones, aún existen deficiencias, especialmente en la incorporación de fundamentos hoy ya básicos de la seguridad, como contar con herramientas de seguridad adecuadas (antivirus, cortafuegos, sistemas de detección de intrusos, etc.), generar políticas de seguridad o separar el área de seguridad del área de TI. Es decir, se están tomando acciones, aunque no a la velocidad necesaria.

En ese contexto [3], las empresas deben responder a la misma velocidad para poder estar protegidos contra las amenazas más actuales y recientes del cibercrimen. Caso contrario, se trata de una carrera en continua desventaja, donde lo que se expone es la información.

1.2 Planteamiento del problema

El avance de la tecnología ha generado nuevos vectores de ataques. Un atacante ahora puede infiltrarse en una red a través de un equipo vulnerable. Por otra parte, puede utilizar la colaboración de equipos comprometidos para llevar a cabo un ataque que interrumpa la operación de los sistemas informáticos o servicios.

Los Sistemas de Detección de Intrusos (IDS) son un ejemplo de las herramientas utilizadas para proteger los sistemas informáticos en contra de este tipo de ataques. El principal objetivo de un IDS [4] es identificar el uso no autorizado, mal uso y abuso de los sistemas informáticos tanto de los usuarios internos como externos.

A partir de la información proporcionada por el IDS y el uso de herramientas como cortafuegos se pueden tomar las medidas adecuadas para interrumpir las conexiones de red, registro de eventos, dar la alarma, y recordar a los administradores del sistema tomar las medidas adecuadas.

[1] En el 2010 el número de internautas mexicanos alcanzó los 34.9 millones [46].

Recientemente, un gran número de enfoques innovadores y nuevos modelos de IDS se han propuesto. En este proyecto se propone desarrollar un IDS basado en anomalías que permita la detección de incidentes utilizando una nueva técnica de inteligencia artificial, el paradigma de Memoria Temporal Jerárquica (HTM), que es un concepto relativamente nuevo que imita el funcionamiento de la zona de neocórtex del cerebro humano.

1.3 Objetivos

El objetivo principal de este trabajo es tratar de identificar algunos incidentes de seguridad más comunes y de gran impacto a partir de las anomalías detectadas por un IDS basado en la teoría HTM desarrollada por Numenta.

Los objetivos específicos del proyecto son los siguientes:

- Poner en funcionamiento la herramienta desarrollada en la tesis de maestría titulada “Sistema de Detección de Intrusos basado en anomalías de red usando la plataforma Numenta para Cómputo Inteligente”.
- Entrenar la red HTM con tráfico normal y validar que el IDS implementado sea capaz de detectar ataques basado en anomalías, mediante la comparación de las categorías creadas durante el entrenamiento y las pruebas.
- Diseñar e implementar un mecanismo para la identificación de incidentes de seguridad utilizando el IDS basado en anomalías.

1.4 Relevancia y contribución del trabajo

La principal contribución de esta tesis se encuentra en el desarrollo e implementación de una herramienta que hace uso de inteligencia artificial para la detección de intrusos. Además, se incluye una metodología para identificar la causa de la misma, recopilando toda la información necesaria que permita determinar el tipo de incidente.

Esta tesis introduce nuevos algoritmos, llamados en conjunto Memoria Temporal Jerárquica, que muestra algunos principios básicos para construir máquinas inteligentes, tales que pueden ser utilizadas con diversos propósitos como la detección de anomalías del tráfico de red de datos.

Este proyecto es una mejora a la tesis titulada: “Sistema de Detección de Intrusos Basado en Anomalías de Red Usando la Plataforma Numenta para Cómputo Inteligente” dirigida por el mismo director de tesis, Dr. Enrique Daltabuit Godas, el cual sólo permitía saber si se había presentado una anomalía en la red pero no determinaba la causa de la misma.

1.5 Estructura de la tesis

La tesis se estructura en seis capítulos principales:

El primer capítulo titulado “*Introducción”* contiene una breve descripción del alcance del trabajo de tesis, y se da una explicación o resumen del mismo, en donde se incluye el planteamiento del problema, los objetivos y la estructura general de la tesis.

En el capítulo 2 “*Fundamentos HTM”* se describe el funcionamiento y características del sistema de memoria conocido como Memoria Temporal Jerárquica (HTM), que a través de la organización jerárquica tanto en espacio como en tiempo, captura y modela la estructura del mundo.

En el capítulo 3 titulado “*NuPIC: Implementación de una red HTM”* se explica cómo utilizar la plataforma de desarrollo NuPIC (Numenta Platform for Intelligent Computing) para crear programas basados en la teoría y tecnología HTM.

En el capítulo 4 *“Diseño e implementación”* se analiza el problema que consiste en identificar incidentes de seguridad a través de un IDS basado en anomalías, para ello se definen las especificaciones de diseño de la red HTM, los procedimientos utilizados y la metodología de evaluación propuesta, así como los escenarios de prueba.

En el capítulo 5 *“Análisis de resultados”* se revisan los resultados del experimento tanto en la etapa de entrenamiento como en las pruebas, los datos generados utilizando la metodología de evaluación propuesta para determinar si es capaz de identificar incidentes de seguridad.

En el último capítulo *“Conclusiones”*, se presentan las conclusiones de los resultados experimentales con respecto a las expectativas iniciales. Se interpretan los resultados, y se proporcionan las recomendaciones para trabajo futuro.

Capítulo 2

Fundamentos de HTM

La finalidad de este capítulo es dar a conocer los fundamentos básicos de la teoría HTM (Memoria Temporal Jerárquica), incluyendo los conceptos de jerarquía, el rol del tiempo y las funciones básicas.

2.1 Introducción

Durante décadas, los científicos en el campo de la inteligencia artificial han afirmado que las computadoras serán inteligentes cuando sean lo suficientemente potentes. Yo no lo creo, y voy a explicar por qué. Los cerebros y computadoras hacen cosas fundamentalmente diferentes. [5]

Jeff Hawkins

¿Por qué es tan difícil para las computadoras desarrollar tareas que los humanos pueden hacer tan fácil y rápido? Durante más de cincuenta años, se ha intentado desarrollar programas para reconocer imágenes, comprender el lenguaje, controlar robots, manipular objetos mediante el tacto y aprender por su cuenta [6].

En un ser humano, estas capacidades son en gran parte realizadas por el neocórtex. Memoria Temporal Jerárquica (HTM, por sus siglas en inglés Hierarchical Temporal Memory) es una tecnología que reproduce las características estructurales y algorítmicas del neocórtex, es decir, intenta capturar la forma en el que el cerebro humano aprende e infiere de su entorno [7].

A diferencia de la programación tradicional, en el que un programador crea programas específicos para resolver problemas específicos (por ejemplo, un programa para el reconocimiento del habla y otro completamente diferente para modelar el clima) y donde tiene control sobre cómo y dónde se almacena la información, HTM, por el contrario, es mejor conocido como un sistema de memoria. HTM no se programa y no ejecuta diferentes algoritmos para diferentes problemas, sino que “aprende” cómo resolver diferentes tipos de problemas.

HTM se organiza como una jerarquía de nodos en forma de árbol, donde cada nodo implementa una función de aprendizaje y memoria común. Todos los objetos en el mundo, ya sean personas, edificios, el habla, o el flujo de información a través de una red de computadoras, tienen una estructura. Esta estructura es jerárquica, tanto en espacio como en tiempo.

2.2 ¿Qué es la Memoria Temporal Jerárquica?

La Memoria Temporal Jerárquica (HTM) es un modelo de máquina de aprendizaje (aprendizaje artificial) desarrollado por Jeff Hawkins y Dileep George de Numenta, Inc. en el 2005. Modela algunas de las propiedades estructurales y algorítmicas del neocórtex, que es una parte de los cerebros de los mamíferos en el cual se realizan funciones de alto nivel como la visión, oído, movimiento, lenguaje y planeación [8].

HTM se originó a partir de un modelo llamado *Sistema de Memoria-Predicción*, descrito por primera vez por Jeff Hawkins en el libro titulado On Intelligence [7]. El Sistema de Memoria-Predicción proponía que el neocórtex utilizaba la secuencia de memoria en una jerarquía para modelar e inferir las causas en el mundo. El Sistema de Memoria-Predicción proponía varios mecanismos novedosos de aprendizaje e incluía un mapeo detallado a gran escala de la arquitectura del tálamo cortical, así como los microcircuitos de las columnas corticales [9].

HTM puede ser considerado como un tipo de red neuronal. Por definición, cualquier sistema que trate de modelar los detalles arquitecturales del neocórtex es una red neuronal. Los HTM modelan neuronas (llamadas células en terminología HTM), las cuales son organizadas en columnas, capas, regiones y en una jerarquía [8]. En la figura 1 se muestra una neurona real del lado izquierdo y del lado derecho la representación de una neurona HTM mediante fórmulas aritméticas.

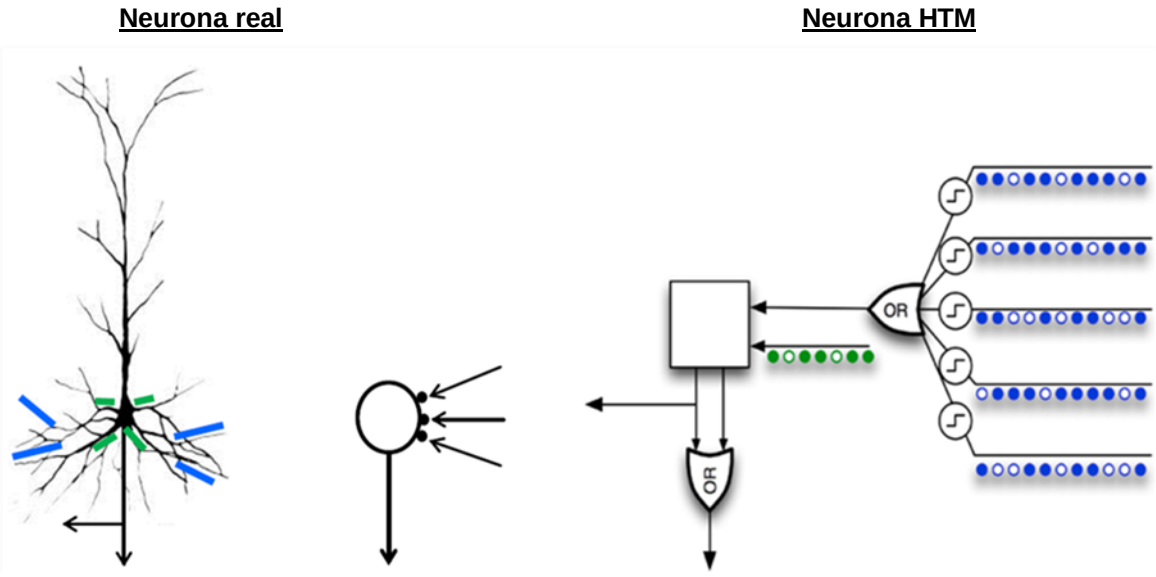


Figura 1. Comparación entre una red neurona real y neurona HTM.

Como su nombre lo indica, HTM es fundamentalmente un sistema basado en memoria que utiliza un algoritmo para resolver todos los tipos de problemas. Las redes HTM se entrenan con datos variables en el tiempo, y se basan en el almacenamiento de un gran conjunto de patrones y secuencias. La forma en que los datos se almacenan y se acceden es lógicamente diferente del modelo estándar utilizados por los programadores de hoy.

A continuación se explicarán algunos conceptos clave relacionados con la red HTM [8]: ¿por qué la organización jerárquica es importante?, ¿cómo son estructuradas las regiones HTM? y ¿por qué la información basada en el tiempo es crítica?

2.2.1 Jerarquía

Las redes HTM están estructuradas como una jerarquía de nodos, donde cada nodo está realizando el mismo algoritmo de aprendizaje. La figura 2 muestra una jerarquía simplificada de HTM [7]. Los datos sensoriales entran en el extremo inferior. La salida en el extremo superior es un vector, donde cada elemento del vector representa una posible causa de los datos sensoriales.

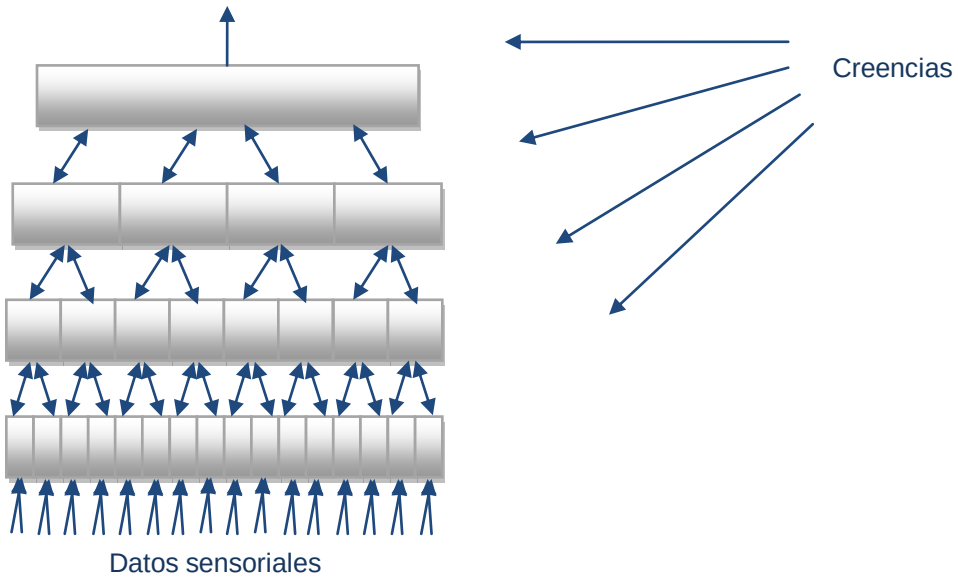


Figura 2. Jerarquía de nodos de una red HTM.

¿Por qué la jerarquía es importante?

El beneficio de la organización jerárquica es la eficiencia. Reduce significativamente el tiempo de entrenamiento y el uso de memoria, ya que los patrones aprendidos en cada nivel de la jerarquía son reutilizados cuando se combinan de manera distinta en los niveles superiores. Existen cuatro razones por las que se debe usar una jerarquía:

- ✓ *Representaciones compartidas conducen a la generalización y eficiencia del almacenamiento.*

  Muchos métodos que se han propuesto para el reconocimiento de patrones son incapaces de solucionar grandes problemas. A menudo, estos métodos fallan porque la cantidad de memoria y tiempo requerido para entrenar crece exponencialmente a medida que el espacio del problema se hace grande. HTM puede requerir mucho entrenamiento y grandes cantidades de memoria, pero no sufre los problemas de escala exponencial debido a la jerarquía.

- ✓ *La jerarquía de la red HTM coincide con la jerarquía espacial y temporal del mundo real.*

  Las redes HTM no sólo utilizan la estructura jerárquica espacial del mundo. Se aprovechan de la estructura jerárquica temporal del mundo. Los nodos en la parte inferior de una red HTM encuentran correlaciones temporales entre los patrones que ocurren relativamente juntos tanto en espacio como en tiempo: “el patrón B inmediatamente sigue al patrón A”. Debido a que cada nodo convierte una secuencia de patrones espaciales en un valor constante, el siguiente nivel en la jerarquía busca secuencias de secuencias.

  Un nodo HTM [10] [11] utiliza dos mecanismos de agrupación para formar invariantes en espacio y tiempo, tal como se muestra en la figura 3. El primero se denomina *agrupador espacial*, que es el responsable [12] de aceptar datos de entrada y hacer una clasificación espacial basada en los patrones que difieren en una distancia máxima configurable.

  El segundo mecanismo se denomina *agrupador temporal*, que agrupa a los patrones que están temporalmente cercanos. De esta manera, los patrones que son muy diferentes, pero que tienen una causa común, pueden estar en el mismo grupo [12].

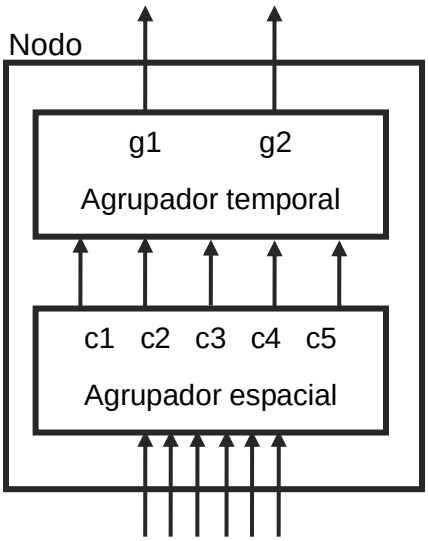


Figura 3. Estructura de un nodo – agrupador espacial y temporal.

En la siguiente figura 4 se muestra el concepto de la abstracción espacial y temporal de la jerarquía. Los niveles más altos son la abstracción de grandes áreas de espacio y larga duración en el tiempo, mientras los niveles superiores cambian sus estados lentamente en comparación con los niveles inferiores.

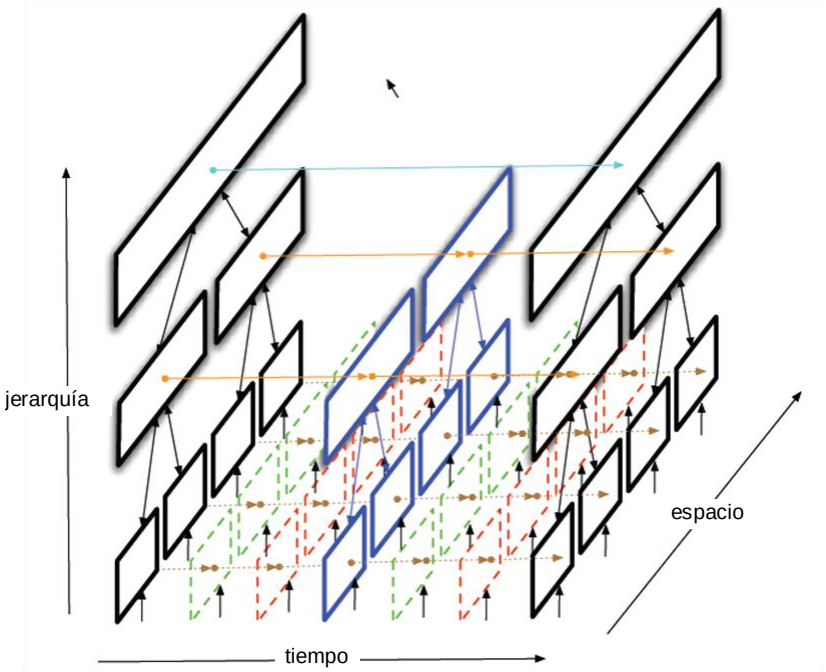


Figura 4. Concepto de la abstracción espacial y temporal de la jerarquía.

- ✓ *La propagación de creencias asegura que todos los nodos alcanzan las mejores creencias compatibles entre sí.*

  La red HTM utiliza una variación de propagación de creencias para hacer la inferencia. Los datos sensoriales generan un conjunto de creencias en el nivel inferior de la jerarquía en una red HTM, y a través del tiempo, las creencias se propagan al nivel más alto. Cada nodo en el sistema representa una creencia que es mutuamente consistente con todos los demás nodos.

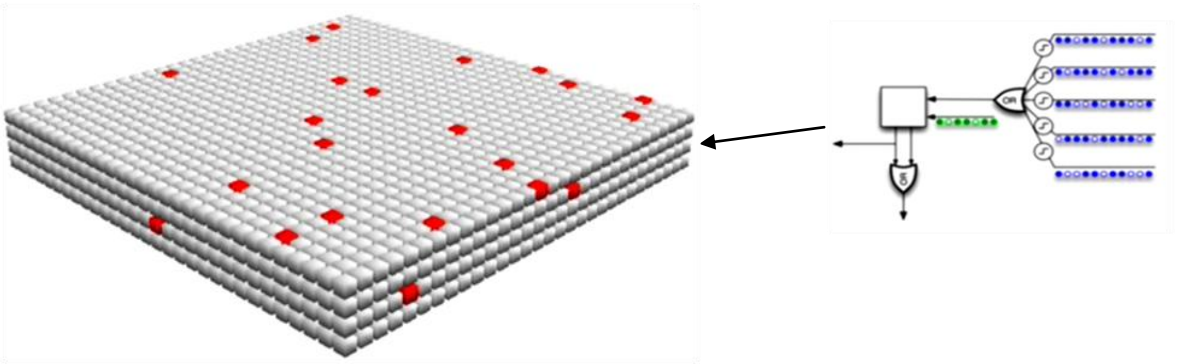
  Una de las ventajas de hacer la inferencia de esta manera es que la ambigüedad se resuelve conforme las creencias ascienden en la jerarquía. El tiempo que tarda una red HTM para inferir su entrada se incrementa linealmente con el número de niveles de la jerarquía y su capacidad de memoria, incrementa exponencialmente con el número de niveles.

Resumiendo, la estructura jerárquica reduce el tiempo de entrenamiento, reduce el uso de memoria, e introduce una nueva forma de generalización [8].

2.2.2 Regiones

Una red HTM se compone de regiones organizadas en una jerarquía. La región es la principal unidad de memoria y predicción en una red HTM. Por lo general, cada región HTM representa un nivel en la jerarquía.

A medida que asciende la jerarquía siempre hay convergencia, múltiples elementos en una región hijo converge en una región padre. Sin embargo, debido a las conexiones de retroalimentación, la información también diverge a medida que desciende de la jerarquía (una “región” y un “nivel” son casi sinónimos. Se usa la palabra “región” para describir el funcionamiento interno de una región, mientras que se usa la palabra “nivel” para referirse al rol de la región dentro de la jerarquía.) En la figura 5 se muestra una representación de las regiones que conforman una red [13].



¿Qué es una región?

- Un conjunto de neuronas dispuestas en columnas.
- Las celdas de la columna tienen la misma activación de alimentación hacia adelante.
- Las celdas de la columna tienen una respuesta diferente en el contexto.

¿Qué hace una región?

1) Crea una representación distribuida dispersa de entrada.
2) Crea una representación de entrada en el contexto de entradas anteriores.
3) Aprende las secuencias de representaciones en 2.
4) Forma una predicción basada en la entrada actual en el contexto de las entradas previas.
   Esta predicción es una representación de lento cambio de secuencia.
   Esto es la salida de una región.

Figura 5. Una red HTM consiste de regiones organizadas en una jerarquía.

2.2.3 El rol del tiempo

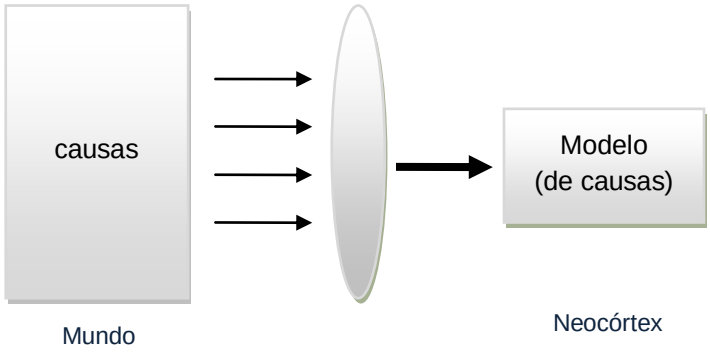
¿Por qué el tiempo es necesario para aprender? Debido a que cada nodo aprende una secuencia común de patrones, la única manera de que un nodo pueda hacer esto es si se presenta una secuencia de patrones a través del tiempo. Por lo tanto, el tiempo juega un rol crucial en el aprendizaje, inferencia, y predicción.

- ✓ **Inferencia**. Sin usar el tiempo, no podríamos inferir casi nada a través de nuestros sentidos táctiles, auditivos y visuales.
- ✓ **Aprendizaje**. Con el fin de aprender, todos los sistemas HTM deben ser expuestos a entradas que cambian en el tiempo durante el entrenamiento. El tiempo es el “supervisor”, enseña cuáles patrones espaciales permanecen juntos.
- ✓ **Predicción**. Aprender y reconocer secuencias es la base de la formación de las predicciones. Una vez que un HTM aprende qué patrones son los que siguen a otros patrones, puede predecir los que se supone serán los siguientes conociendo la entrada actual y las entradas inmediatamente anteriores.

2.3 Funciones básicas de HTM

Se ha sabido desde hace más de veinticinco años que el neocórtex (corteza cerebral) funciona con un algoritmo común; la visión, oído, tacto, lenguaje, comportamiento, y todo lo demás que el neocórtex hace, son manifestaciones de un único algoritmo aplicado a las diferentes modalidades de información sensorial.

A continuación se describirán las cuatro funciones básicas del HTM [8], mostradas en la figura 6: aprendizaje, inferencia, predicción y comportamiento. Cada región HTM ejecuta las primeras tres funciones: aprendizaje, inferencia y predicción. El comportamiento, por otro lado, es diferente.



1) Aprendizaje
2) Inferencia
3) Predicción
4) Comportamiento

Figura 6. Funciones básicas de una red HTM.

2.3.1 Aprendizaje: descubrir las causas en el mundo

Algunos objetos en el mundo pueden ser físicos como los carros, personas y edificios; otros pueden ser no físicos como las ideas, las palabras, canciones o el flujo de información en una red. El atributo importante de los objetos en el mundo desde la perspectiva de una red HTM es que ellos tienen una estructura persistente; ellos existen en el tiempo. Llamamos a los objetos en el mundo “causas”. [7]

En la figura 7 se muestra cómo un sistema HTM se relaciona con el mundo. A la izquierda de esta figura está un cuadro que representa un mundo que la red HTM está aprendiendo. El mundo se compone de objetos y sus relaciones. En el lado derecho de la figura se encuentra una red HTM, la cual se comunica con su mundo a través de uno o más sentidos, tal como se muestran a la mitad de la figura.

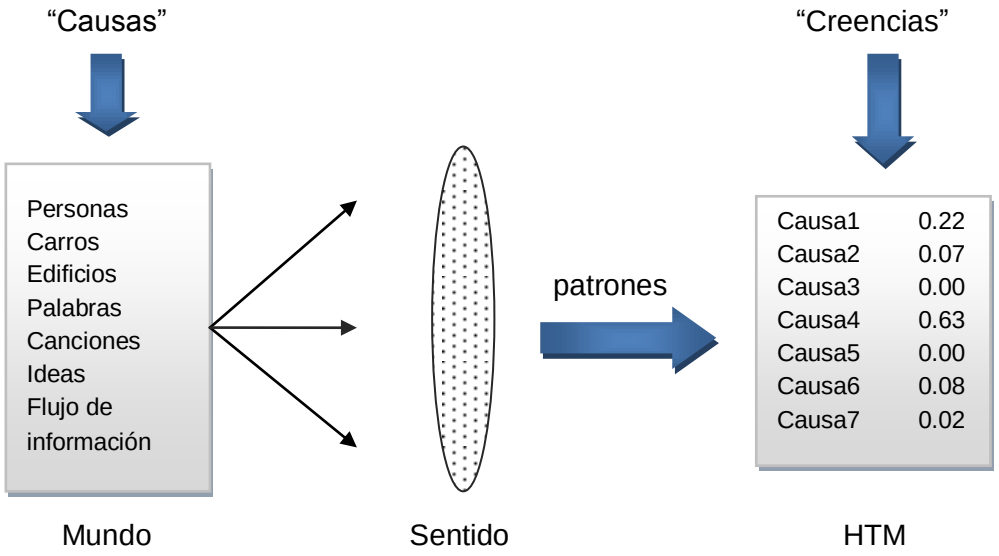


Figura 7. HTM y su relación con el mundo.

Los sentidos utilizan algunos atributos del mundo como la luz o el tacto, los cuales no necesitan ser los mismos que tienen los seres humanos. Comúnmente los sentidos no detectan directamente los objetos en el mundo, en una red HTM se presenta como un arreglo de datos, en donde cada elemento del arreglo es una medición de algunos atributos del mundo.

Desde la perspectiva de HTM, hay dos características esenciales de los datos sensoriales. En primer lugar, el dato debe medir algo que es directa o indirectamente afectado por las causas en el mundo y que es de nuestro interés. Si se quiere una red que entienda el tráfico de red de una computadora, debe detectar paquetes de red por segundo. En segundo lugar, los datos sensoriales deben cambiar y el flujo debe ser continuo a través del tiempo, mientras las causas subyacentes de los datos se mantienen relativamente estables.

La red HTM recibe el patrón espacio-tiempo proveniente de los sentidos. Al principio, la red HTM no tiene conocimiento de las causas, pero a través de un proceso de aprendizaje, “descubre” cuáles son las causas.

En la red HTM, las causas son representadas por números en un vector. En cualquier momento en el tiempo, una red HTM asignará una probabilidad a las causas individuales que están siendo detectadas. La salida de la red se manifiesta como un conjunto de probabilidades para cada una de las causas aprendidas. Esta distribución momento a momento de posibles causas es llamada una “creencia”.

Una región HTM aprende sobre su mundo encontrando patrones y luego secuencia de patrones en la información sensorial. La región no “sabe” que representa la información de entrada, trabaja en un campo puramente estadístico. Busca combinaciones de entradas que ocurran juntas normalmente, lo que llamamos patrones espaciales. Entonces busca en qué secuencia aparecen estos patrones en el tiempo, lo que llamamos patrones temporales o secuencias.

Una simple región HTM tiene una capacidad de aprendizaje limitada. Una región automáticamente ajusta lo que aprende basándose en cuanta memoria tiene y la complejidad de la entrada que está recibiendo.

2.3.1.1 Aprendizaje en un nodo

La entrada al nodo durante el proceso de aprendizaje es una larga secuencia de patrones [14]. El nodo opera los patrones de entrada de uno en uno. Para cada patrón de entrada, el nodo hace tres operaciones: memorización de los patrones, aprendizaje de probabilidades de transición y agrupación temporal.

Intuitivamente, un nodo HTM que ha aprendido puede ser considerado como aquel que ha memorizado un conjunto de patrones (coincidencia de patrones) y un conjunto de secuencias de patrones (cadenas de Markov[2]).

[2] Una cadena de Markov es una serie de eventos, en la cual la probabilidad de que ocurra un evento depende del evento inmediato anterior. Las cadenas de este tipo tienen memoria. “Recuerdan el último evento y esto condiciona las posibilidades de los eventos futuros [47].

2.3.2 Inferencia: deducir las causas de nuevas entradas

Después de que la red HTM ha aprendido cuáles son las causas del mundo y cómo representarlos, puede realizar inferencia. “Inferencia” es similar al reconocimiento de patrones. Dado un flujo de nuevos datos sensoriales, una red HTM puede “inferir” cuáles son las causas conocidas probables que se podrían presentar en el mundo en ese momento [7].

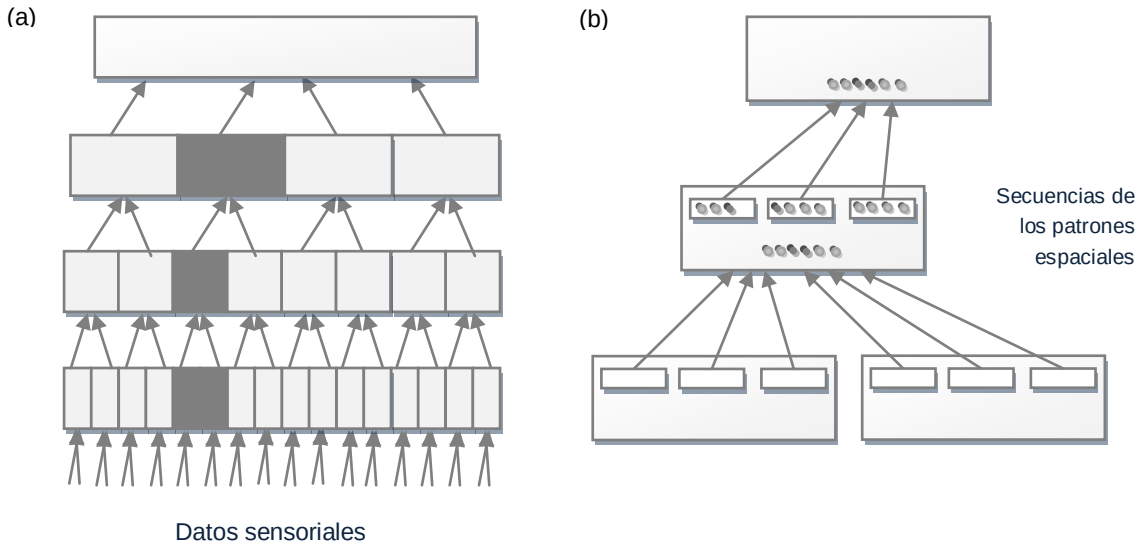
Las creencias actuales inferidas de una red HTM pueden ser leídas directamente desde el sistema a ser utilizado o desde otros lugares externos a la red (algo que no es posible en el cerebro humano). A pesar de que a veces es posible realizar inferencias con un patrón sensorial estático, la teoría detrás de la redes HTM muestra que no es posible descubrir las causas sin necesidad de cambiar continuamente las entradas.

Una región HTM encara el mismo problema que tu cerebro: las entradas puede que no sean exactamente las mismas jamás. Consecuentemente, igual que tu cerebro, una región HTM debe manejar nuevas entradas durante la inferencia y el entrenamiento. Los datos fluyen en ambos sentidos en la jerarquía y un cambio en un nodo de alto nivel puede afectar las creencias en un nodo de nivel inferior a través de una técnica conocida como propagación de la creencia [5]. Las redes HTM utilizan la propagación de la creencia Bayesiana para la inferencia (ver Anexo C).

2.3.3 Predicción

La predicción [15] es una función de todas las partes del cerebro. En cada momento de nuestra vida que estamos despiertos, nuestro cerebro está tratando de predecir qué será lo que nuestros sentidos del tacto, vista y oído experimentarán próximamente. HTM modela el neocórtex como una jerarquía en forma de árbol de las regiones de memoria, en el que cada región de memoria aprende las secuencias comunes de los patrones (ver figura 8).

Cada región de una red HTM almacena secuencias de patrones. Al hacer coincidir las secuencias almacenadas con las nuevas entradas, una región forma una predicción sobre las entradas que vendrán a continuación. Una región HTM predecirá diferentemente basado en el contexto que puede haber sido generado tiempo atrás.



(a) Un diagrama conceptual de un modelo HTM del neocórtex. (b) Cuatro regiones conectadas de (a), ilustrando la ruta de alimentación. Los círculos indican los patrones espaciales que forman los elementos de las secuencias aprendidas. Pequeños rectángulos indican las secuencias aprendidas de patrones. Cada región utiliza su secuencia de memoria para predecir cuáles son los elementos que ocurrirán probablemente y pasa esta predicción en la jerarquía.

Figura 8. Una región HTM hará diferentes predicciones basado en el contexto.

A continuación se presentan algunas propiedades clave de la predicción HTM [8]:

1) **La predicción es continua.**
   Sin ser consciente, constantemente estamos prediciendo. HTM hace lo mismo. Cuando escuchas una canción, tratas de predecir la siguiente nota. En una región HTM, la predicción e inferencia son casi lo mismo. La predicción no es un paso separado, es integral en la manera de trabajar de una región HTM.

2) **La predicción ocurre en cada región y a cada nivel de la jerarquía.**
   Si se tiene una jerarquía de regiones HTM, la predicción ocurrirá a cada nivel. Las regiones harán predicciones sobre los patrones que han aprendido. En un ejemplo del lenguaje, regiones de niveles inferiores pueden predecir los posibles siguientes fonemas, mientras que las regiones superiores pueden predecir palabras o frases.

3) **Las predicciones son sensibles al contexto.**
   Las predicciones se basan en lo que ha ocurrido en el pasado, así como lo que está ocurriendo en este mismo momento. De esta manera, una entrada producirá diferentes predicciones basado en el contexto previo. Una región HTM aprende a utilizar tanto contexto previo como el que le haga falta, y puede mantener el contexto tanto en periodos cortos y largos de tiempo. Esta habilidad es conocida como memoria de “orden variable”.

4) **La predicción conduce a la estabilidad.**
   La salida de una región es su predicción. Una de las propiedades de los HTM es que las salidas de las regiones se estabilizan – esto es, cambian más despacio y permanecen más en el tiempo – cuando más altos estén en la jerarquía. Esta propiedad es resultado de la manera en que predice una región. Una región no se limita a predecir lo que pasará justo inmediatamente. Si puede, predecirá varios pasos en el futuro.

5) **Una predicción nos dice si una nueva entrada es esperada o inesperada.**
   Cada región HTM es un detector de novedades. Debido a que cada región predice qué va a ocurrir a continuación, “sabe” cuando algo inesperado ha ocurrido. Los HTMs pueden predecir muchas posibles siguientes entradas simultáneamente, no sólo una. Puede que no sea capaz de predecir exactamente qué pasará a continuación, pero si la siguiente entrada no coincide con alguna de las predicciones, la región HTM sabrá que ha ocurrido una anomalía.

6) **La predicción ayuda a hacer el sistema más robusto al ruido.**
   Cuando un HTM predice lo que parece que ocurrirá, la predicción puede influir al sistema hacia la inferencia de lo que se predijo. Por ejemplo, si una red HTM estuviera procesando el lenguaje hablado, predeciría qué sonidos, palabras e ideas serían las que se pronunciarían a continuación. Esta predicción ayuda al sistema a añadir los datos que falten. Si un sonido ambiguo es recibido, el HTM interpretará el sonido atendiendo a lo que estaba esperando, ayudando de esta manera a inferir incluso en la presencia de ruido.

En una región HTM, la memoria secuencial, la inferencia y la predicción están estrechamente integrados. Son las funciones básicas de una región.

2.3.4 Comportamiento

Nuestro comportamiento condiciona lo que percibimos. En la medida que movemos nuestros ojos, nuestra retina recibe información sensorial cambiante en el tiempo. Mover nuestras manos y dedos causa variación en las sensaciones del texto. Casi cualquier acción produce cambios en lo que sentimos. La información sensorial y el comportamiento motor están estrechamente relacionados.

Una red HTM que ha aprendido las causas en su mundo, y cómo se comportan esas causas en el tiempo, en esencia ha creado un modelo de su mundo. En la figura 9 se muestra un sistema con una red HTM y la capacidad de generar simples comportamientos [7].

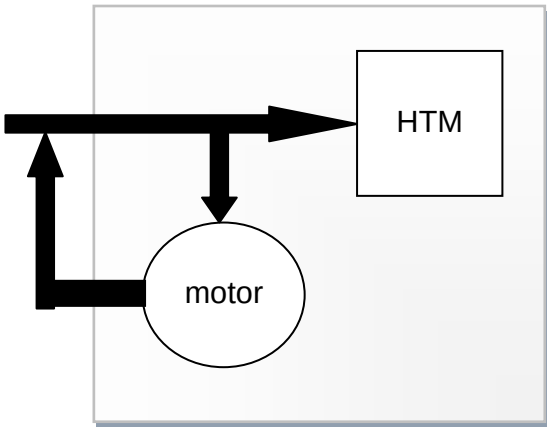
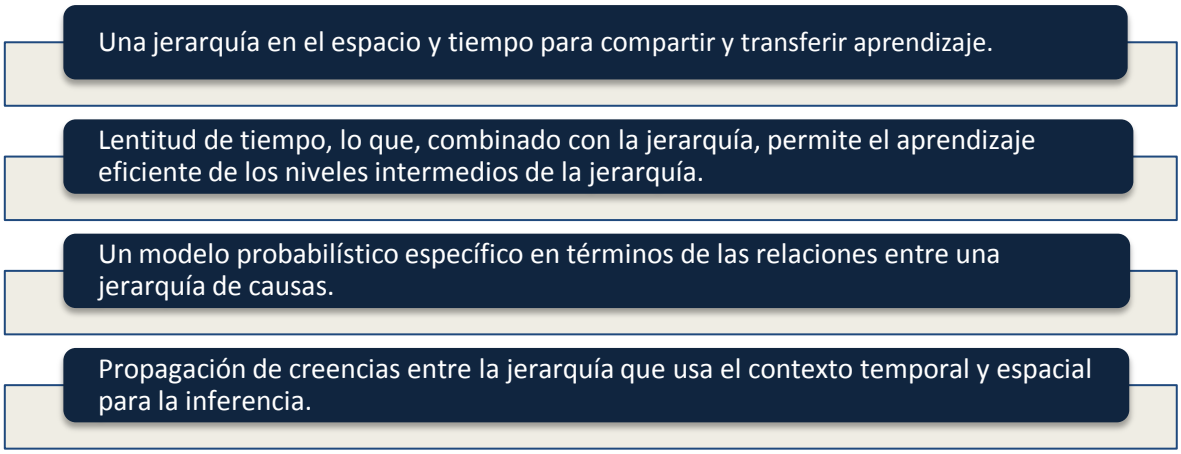


Figura 9. HTM modela el mundo.

A medida que la red HTM descubre las causas en su mundo, aprende a representarlo considerando su comportamiento, del mismo modo que aprende a representar el comportamiento de los objetos del mundo exterior. Desde la perspectiva HTM, el sistema está conectado a otro objeto más en el mundo. La red HTM forma representaciones de los comportamientos del sistema al que está conectado, y lo más importante, aprende a predecir su actividad.

2.4 Comparación con otros modelos existentes

En esta sección se comparará HTM con varias tecnologías existentes para el modelado de datos. HTM utiliza una combinación única de las siguientes ideas [16]:

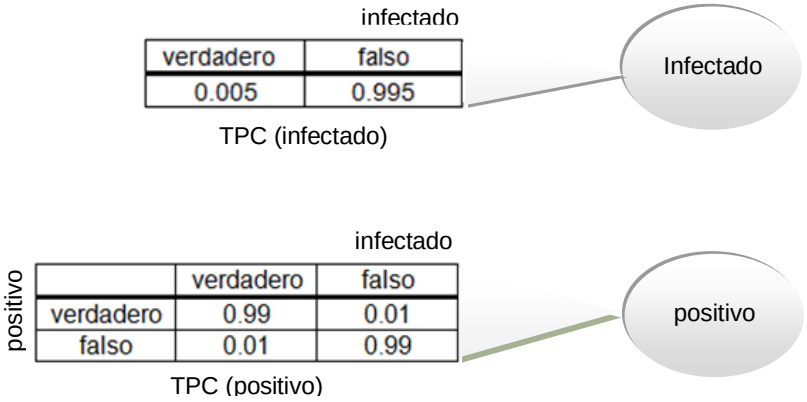


Muchas de estas ideas ya existían y han sido parte de algunos modelos que se describen a continuación. El poder de HTM proviene de una síntesis de estas ideas.

2.4.1 Modelos probabilísticos de propósito general

Existen muchas clases de modelos probabilísticos que se utilizan para analizar las relaciones entre un conjunto de variables. Ellos no especifican ni requieren algún significado especial en las variables. Del mismo modo, no especifican ni requieren una relación estadística entre las variables, a pesar de que funcionan mejor cuando se puede explotar independencias condicionales entre las variables. Estos modelos son simplemente formas eficientes de representar y realizar cálculos sobre las distribuciones de probabilidad, entre ellos se encuentran las redes bayesianas.

Una red bayesiana [17] se utiliza para modelar un dominio que contiene incertidumbre. Es un grafo dirigido cíclico (DAG), donde cada nodo representa una variable discreta aleatoria de interés. Cada nodo contiene los estados de la variable aleatoria que representa y una tabla de probabilidad condicional (TPC). Un ejemplo de este tipo de red se muestra en la figura 10.



Posible red bayesiana que modela el ejemplo de que un agricultor tiene una botella de leche que puede estar infectado o limpio. Las dos variables aleatorias son representadas como dos nodos en la red.

Figura 10. Ejemplo de red bayesiana y TPC.

Aunque estos modelos son excelentes herramientas para el análisis probabilístico, y es muy difícil que resuelvan problemas difíciles de la vida real. HTM no infringe los principios fundamentales de estos modelos, sino que van más allá, al hacer suposiciones adicionales sobre la naturaleza del mundo.

HTMs son similares a las redes bayesianas [7]; sin embargo, se diferencian en la forma que utilizan el tiempo, la jerarquía, la acción y la atención. HTMs pueden ser implementados con software en el hardware tradicional, pero lo mejor es pensar en HTM como un sistema de memoria.

2.4.2 Modelos no-generativos

Muchos algoritmos de aprendizaje no se preocupan por construir un modelo que pueda describir la entrada, sino que pasa directamente al mapeo directo de las entradas a la respuesta correcta, donde la respuesta correcta es proporcionada por alguna fuente externa. Estos modelos suelen ser supervisados, mientras que los HTM están básicamente sin supervisión. Además, esta es la capacidad de HTM para generar datos que le permiten hacer cosas como predecir en el tiempo. Ejemplo de estos modelos son las *redes neuronales*.

Una red neuronal [18] es un conjunto interconectado de elementos simples de procesamiento, unidades o nodos, cuya funcionalidad se basa en el de la neurona de los animales. La capacidad de procesamiento de la red se almacena en los pesos de las conexiones de la inter-unidad, obtenida mediante un proceso de adaptación a, o aprendizaje de, un conjunto de patrones de entrenamiento.

Las redes neuronales “clásicas” [16] son modelos de aprendizaje supervisados que comúnmente se entrenan utilizando un algoritmo conocido como “retropropagación”, un ejemplo de ello se muestra en la figura 11. Las redes neuronales clásicas generalmente no son consideradas como modelos generativos. Aunque algunas instancias de las redes neuronales utilizan el tiempo y el espacio, ellos no explotan la coherencia temporal como lo hace HTM.

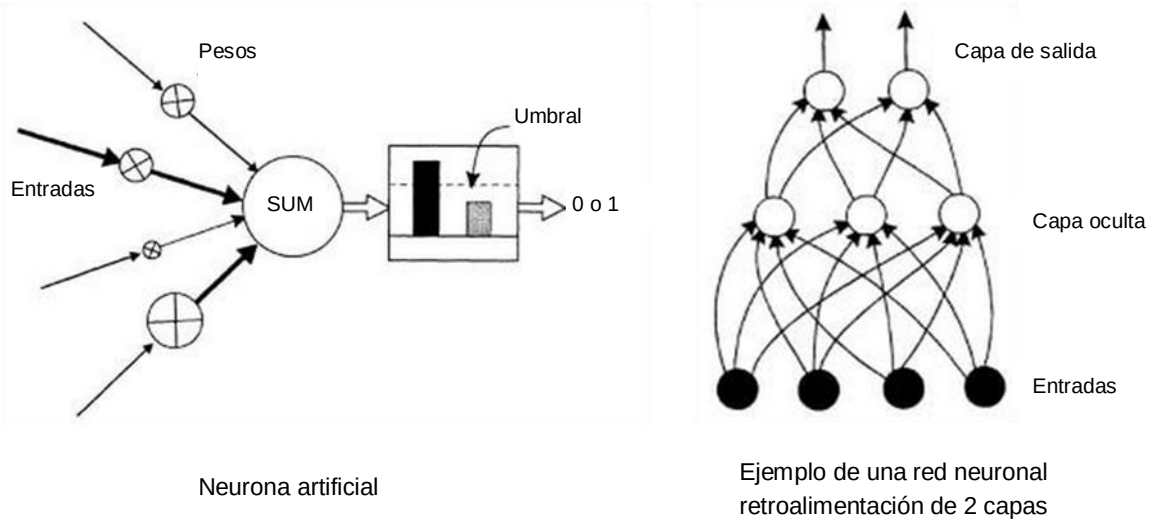


Figura 11. Algoritmo de aprendizaje: red neuronal.

Una vez que se introdujeron los conceptos básicos de HTM, en el siguiente capítulo se explicará cómo utilizar NuPIC, la plataforma de desarrollo de Numenta, para crear programas que permitan modelar las entradas del mundo y realizar inferencias.

Capítulo 3

NuPIC: Implementación de una red HTM

A continuación se explicará el proceso de desarrollo de programas basados en la teoría y tecnología HTM utilizando la plataforma NuPIC, sus aplicaciones en diversas áreas de la vida cotidiana y el trabajo previo en la detección de anomalías.

3.1 NuPIC

"La mejor forma de predecir el futuro es implementarlo"

David Heinemeier Hansson

NuPIC (Numenta Platform for Intelligent Computing) es la plataforma de desarrollo de software que permite a los desarrolladores crear programas basados en la teoría y tecnología HTM, el cual modela la estructura del neocórtex (ver figura 12). Incorpora técnicas de inteligencia artificial, tales como la inferencia, aprendizaje inductivo, y una variación de la probabilidad bayesiana [19].

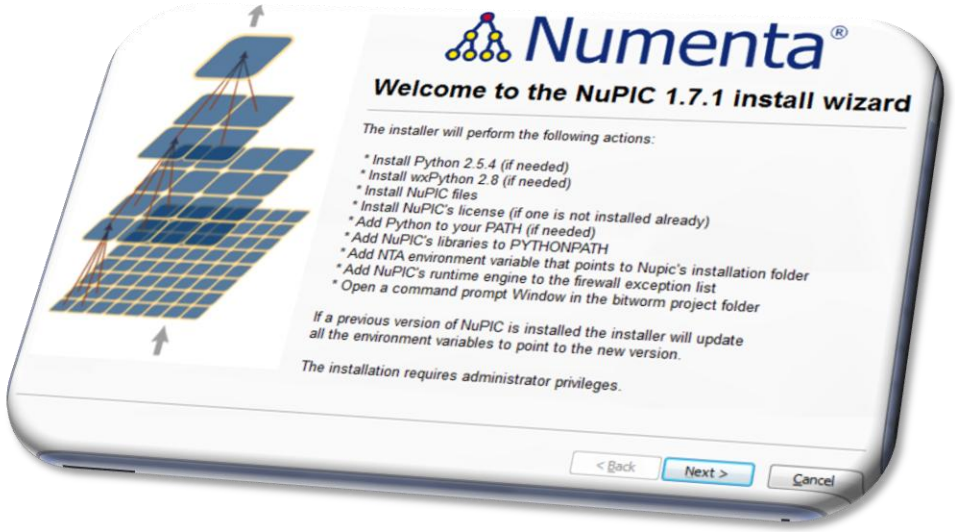


Figura 12. NuPIC: Plataforma de desarrollo Numenta.

El proceso de creación de una aplicación HTM difiere significativamente del proceso tradicional de ingeniería de software. Las redes HTM aprenden mediante la construcción de un modelo estadístico basado en una secuencia de datos de entrada.

La tarea de los desarrolladores de aplicaciones no es especificar un algoritmo sino crear una representación adecuada de los datos y encontrar la configuración óptima de los parámetros para el problema en cuestión. La calidad de la red final depende de muchos factores, tales como la configuración de la red (¿la jerarquía tiene dos o tres niveles?), los parámetros del nodo (¿su distancia máxima es 0 o 0.2?), los datos de entrenamiento (¿hay suficientes secuencias?), y así sucesivamente [20].

El proceso de desarrollo de una red HTM es iterativo en un nivel muy alto. No sólo se tiene que escribir el programa, encontrar problemas, reescribir y volver a ejecutar. En su lugar, cada tarea en el proceso puede influir en otras tareas antes o después durante el proceso. En la siguiente figura 13 se muestra el proceso de desarrollo utilizando NuPIC [21]:

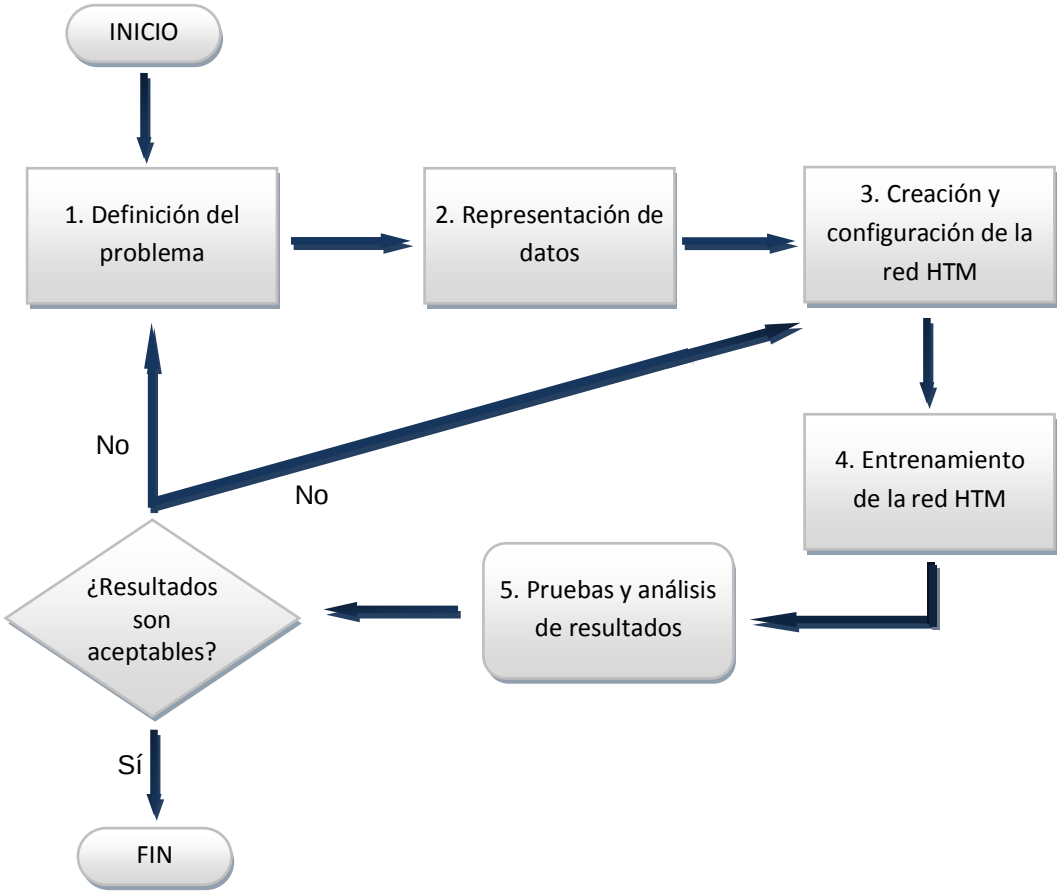


Figura 13. Proceso de desarrollo de Numenta.

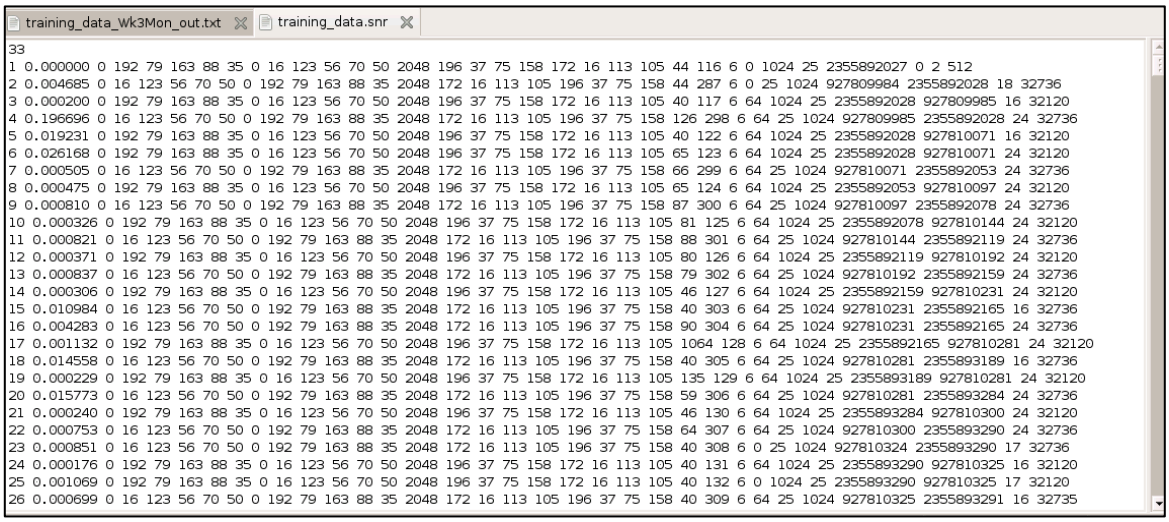
3.1.1 Definición del problema

Antes de que una red pueda ser desarrollada, es crucial entender el tipo datos que se utilizarán. Los investigadores necesitan entender a profundidad sus datos y crear un esquema de representación que enfatice sus características espaciales y temporales. En cualquier esquema, las redes HTM esperan que los datos se presenten como una serie de vectores.

3.1.2 Representación de datos

Cualquier lenguaje de programación puede ser utilizado para generar conjuntos de datos en una red HTM. En una red, cada conjunto de datos contiene dos archivos: 1) un archivo de datos de los vectores de entrada, donde se coloca cada vector en una nueva línea del archivo, 2) un archivo de categorías en la que cada fila correctamente clasifica el vector correspondiente en el archivo de datos. Con el fin de entrenar y probar una red se utilizan dos grupos distintos.

En la figura 14 se muestra el formato de un archivo de datos de entrenamiento:


training_data_Wk3Mon_out.txt | training_data.snr

```
33
1 0.000000 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 44 116 6 0 1024 25 2355892027 0 2 512
2 0.004685 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 44 287 6 0 25 1024 927809984 2355892028 18 32736
3 0.000200 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 40 117 6 64 1024 25 2355892028 927809985 16 32120
4 0.196696 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 126 298 6 64 25 1024 927809985 2355892028 24 32736
5 0.019231 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 40 122 6 64 1024 25 2355892028 927810071 16 32120
6 0.026168 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 65 123 6 64 1024 25 2355892028 927810071 24 32120
7 0.000505 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 66 299 6 64 25 1024 927810071 2355892053 24 32736
8 0.000475 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 65 124 6 64 1024 25 2355892053 927810097 24 32120
9 0.000810 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 87 300 6 64 25 1024 927810097 2355892078 24 32736
10 0.000326 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 81 125 6 64 1024 25 2355892078 927810144 24 32120
11 0.000821 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 88 301 6 64 25 1024 927810144 2355892119 24 32736
12 0.000371 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 80 126 6 64 1024 25 2355892119 927810192 24 32120
13 0.000837 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 79 302 6 64 25 1024 927810192 2355892159 24 32736
14 0.000306 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 46 127 6 64 1024 25 2355892159 927810231 24 32120
15 0.010984 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 40 303 6 64 25 1024 927810231 2355892165 16 32736
16 0.004283 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 90 304 6 64 25 1024 927810231 2355892165 24 32736
17 0.001132 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 1064 128 6 64 1024 25 2355892165 927810281 24 32120
18 0.014558 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 40 305 6 64 25 1024 927810281 2355893189 16 32736
19 0.000229 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 135 129 6 64 1024 25 2355893189 927810281 24 32120
20 0.015773 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 59 306 6 64 25 1024 927810281 2355893284 24 32736
21 0.000240 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 46 130 6 64 1024 25 2355893284 927810300 24 32120
22 0.000753 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 64 307 6 64 25 1024 927810300 2355893290 24 32736
23 0.000851 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 40 308 6 0 25 1024 927810324 2355893290 17 32736
24 0.000176 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 40 131 6 64 1024 25 2355893290 927810325 16 32120
25 0.001069 0 192 79 163 88 35 0 16 123 56 70 50 2048 196 37 75 158 172 16 113 105 40 132 6 0 1024 25 2355893290 927810325 17 32120
26 0.000699 0 16 123 56 70 50 0 192 79 163 88 35 2048 172 16 113 105 196 37 75 158 40 309 6 64 25 1024 927810325 2355893291 16 32735
```

Figura 14. Archivo de datos de entrenamiento.

3.1.3 Creación y configuración de una red HTM

Las redes HTM son creadas y configuradas mediante scripts de Python. Aunque la mayoría de la secuencia de comandos sigue un patrón estándar, cada red requiere que se personalice. Para las redes HTM básicas, no es necesario crear nodos y enlaces explícitamente, sino que se pueden utilizar las funciones de ayuda para crear la estructura, agregar sensores, niveles y enlaces, entre otros.

En la figura 15 se muestra el proceso general de desarrollo de una red HTM:

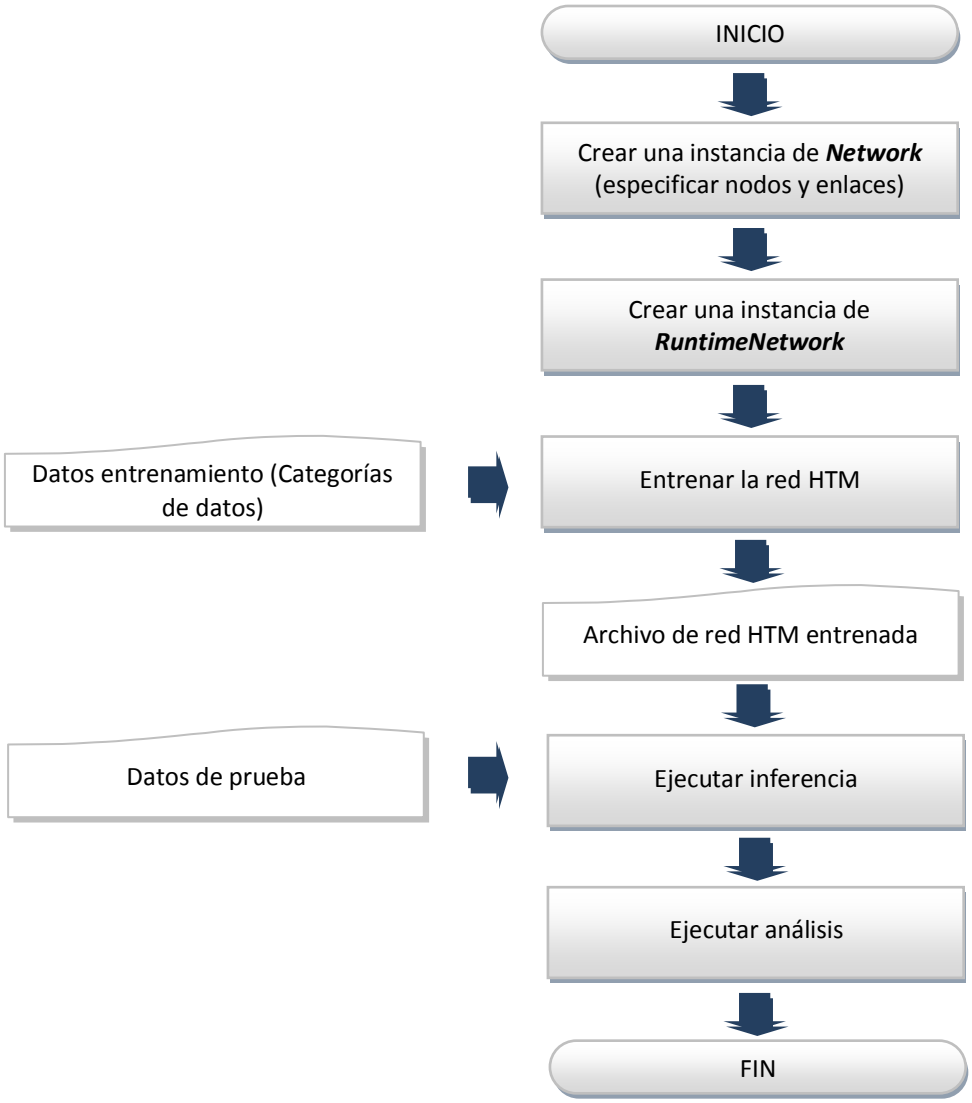


Figura 15. Descripción general del proceso de desarrollo HTM.

3.1.3.1 Creación de los componentes de una red HTM

En la tabla 1 se describen los pasos necesarios para crear una red no entrenada [20]:

Paso	Tarea
1	Crear una red HTM vacía Crear y configurar nodos, agregarlos a la red HTM
2	O Crear una plantilla de nodos, en el que se crean regiones, y se agregan a las redes HTM
4	Enlazar los elementos de la red

Tabla 1. Visión general de las tareas de creación de la red HTM.

Las redes no entrenadas no tienen alguna capacidad inherente y no pueden ser utilizadas para realizar inferencia. La figura 16 muestra un ejemplo de una red HTM. Aunque la red del ejemplo muestra una pequeña y alta simetría, una red HTM en realidad puede tener un número arbitrario de niveles y conexiones.

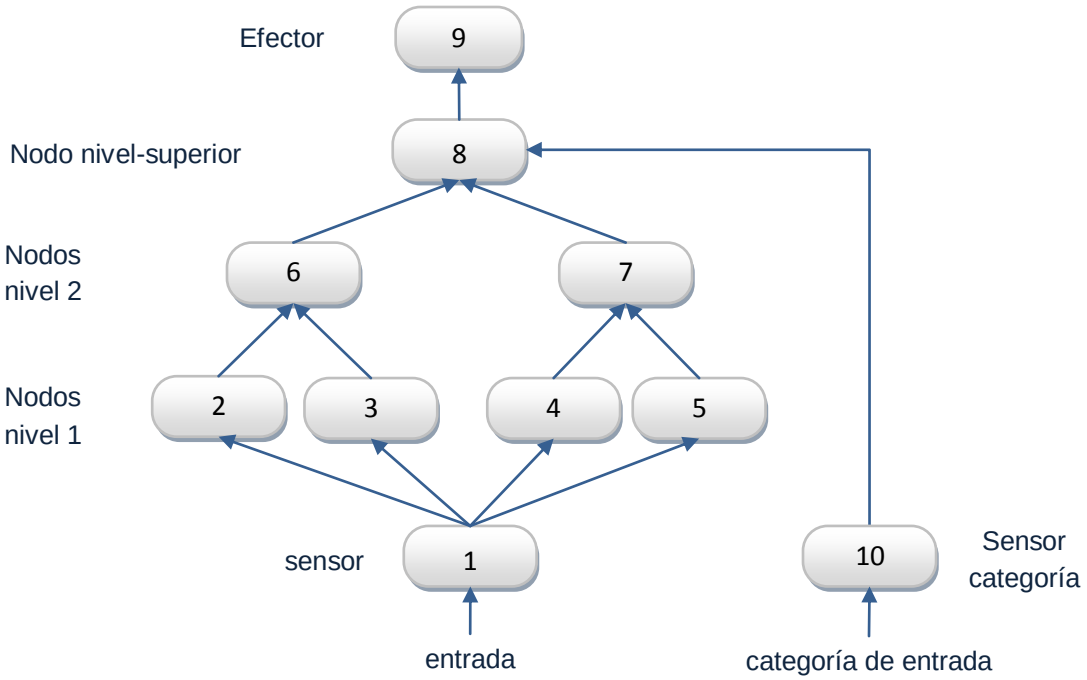


Figura 16. Estructura de una red HTM.

Nodos

Un nodo – sensor, efector, o nodo de aprendizaje – es la unidad básica de HTM Numenta. Cada nodo se implementa como un plugin de python o C++ para NuPIC. Las categorías de los nodos básicos son:

- ✓ Sensores – toman los datos de entrada y los ponen disponibles para los nodos de aprendizaje.
- ✓ Nodos de aprendizaje – toma los datos de un nodo de aprendizaje o de un sensor y modela los datos basados en la configuración de ciertos parámetros.
- ✓ Efectores – toma los datos de un nodo de aprendizaje y los envía fuera del mundo.

Regiones

Una región es un arreglo de nodos que tienen exactamente los mismos parámetros, incluyendo la misma fase. En la figura 17 se observa una región de 4 y 6 nodos.

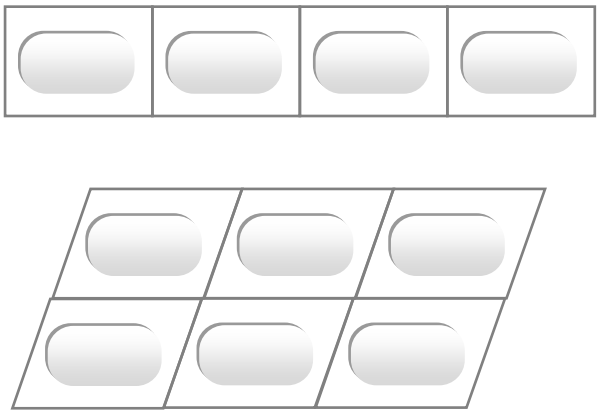


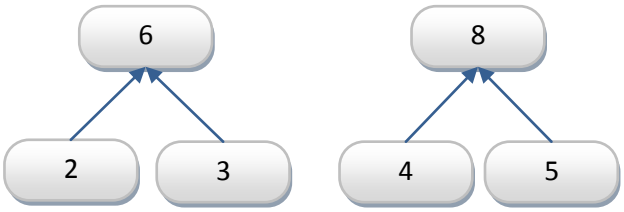
Figura 17. 1-D Región de 4 nodos y 2-D Región de 2x3 nodos.

Enlaces

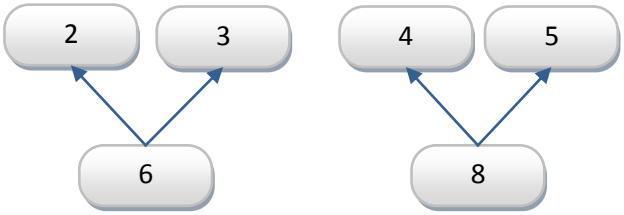
Los enlaces determinan el flujo de información en la red HTM. Se pueden vincular enlaces individuales, o arreglos de enlaces de nodos a la vez utilizando los tipos de vínculos. Existen tres tipos de enlaces:

- ✓ **SingleLink** – conexión predeterminada entre dos nodos.
- ✓ **FanIn** – para la vinculación de una región a otra región divide los nodos de manera uniforme. Existe un número de opciones:

- Múltiples nodos en niveles inferiores envían su salida a un nivel superior, donde la salida se concatena, como se muestra en la siguiente figura:



- Un número menor de nodos de nivel inferior envía su salida a un mayor número de nodos de nivel superior. En este caso, cada nodo de nivel superior recibe la salida completa desde el nodo de nivel inferior.



- Regiones 2-D. Cuando dos regiones 2-D están enlazadas, los nodos en el nivel inferior se encuentran divididos en partes iguales para enviar su salida al siguiente nivel. Por ejemplo, si se tienen 8x8 (64) nodos en el nivel inferior, y tiene 2x2 (4) nodos en el siguiente nivel, cada nodo de nivel superior recibe el aporte de 16 nodos de nivel inferior dispuestas en una cuadrícula de 4x4 (ver figura 18).

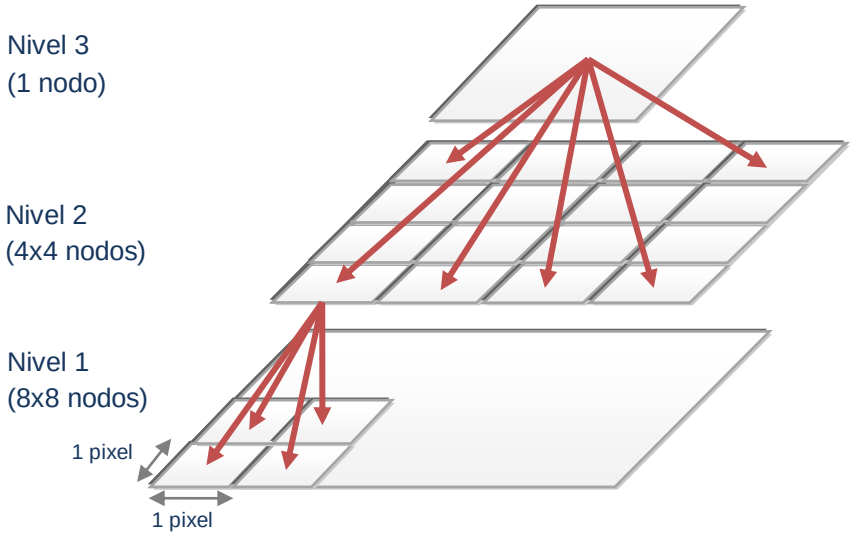
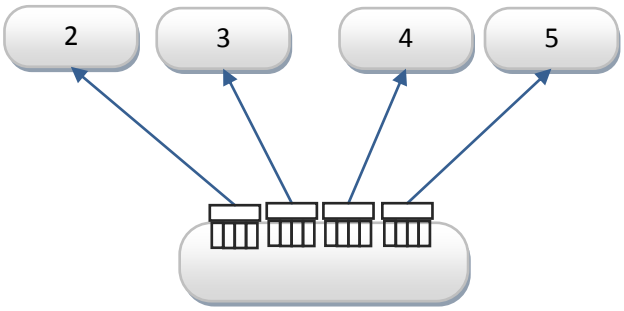


Figura 18. Regiones 2-D enlazados.

- ✓ **SensorLink** – Cuando se conecta un sensor a una región, el objetivo es, por lo general, dividir la matriz de salida del sensor de manera uniforme en todos los nodos de la región. Cuando se utiliza SensorLink, el sistema se encarga de esto automáticamente para 1-D, 2-D o matrices 3-D.



Por ejemplo, si la salida de un sensor es un arreglo de 64 elementos, y la región consiste de cuatros nodos, los primeros 16 datos van al primer nodo, los siguientes 16 datos van al segundo nodo, y así sucesivamente.

3.1.4 Entrenamiento de la red HTM

El entrenamiento de la red es uno de los pasos más fáciles en el proceso debido a una sencilla API que se utiliza para automatizar el entrenamiento. La API de automatización permite a los desarrolladores escribir rápidamente un script de entrenamiento, aunque puede tardar un tiempo en ejecutarse.

3.1.5 Pruebas y análisis de los resultados

Una vez que el script de entrenamiento ha terminado de ejecutarse, el siguiente paso es utilizar un conjunto de datos de prueba para evaluar la exactitud de la red en su conjunto, así como investigar cómo las creencias de los nodos afectan a la red. Si los resultados de las pruebas no son satisfactorios, se tienen dos opciones. Si la precisión de los datos es baja se debe a un pobre esquema de representación de datos, por lo que se deberá tratar de volver a definir los datos y desarrollar un nuevo esquema de representación. Si se necesita un ligero aumento en la exactitud, los parámetros del nodo HTM pueden ser modificados y la red puede ser otra vez entrenada.

3.2 Cómputo de alto desempeño

Además de los requisitos de asociación de datos en los algoritmos y su problema al tratar con patrones impredecibles, las rutinas de búsqueda y coincidencia hacen que la actual implementación del nodo sea un cuello de botella, por lo que se espera que los sistemas con ejecución en paralelo sean las principales plataformas en el futuro del modelado de la cognición [12].

El *cómputo de alto desempeño* consiste en la ejecución simultánea de instrucciones desde el mismo programa pero en diferentes procesadores. Implica la división del programa en múltiples procesos a fin de reducir el tiempo de ejecución.

Hoy en día, los sistemas HTM se ejecutan completamente en software. Debido a que HTM comprende muchos nodos que realizan una función similar, se puede ejecutar al mismo tiempo y fácilmente dividir entre varias CPUs. La implementación actual se puede ejecutar en CPUs de un solo núcleo, multiprocesadores y grandes clústeres de computadoras.

El uso por defecto y el más común de NRE es tener un solo procesador de nodos. Para ese caso, los cálculos se realizan en serie, y las CPU adicionales no mejoran el rendimiento. Para tomar ventaja de más de un CPU, se debe ejecutar con más de un nodo procesador (NP). NuPIC automáticamente inicia varios procesos del sistema operativo cuando se solicita más de un NP [22].

Cuando se usa el NRE con dos NPs se crean los siguientes procesos mostrados en la tabla 2:

Proceso	Descripción
Tres procesos llamados *numenta_runtime*	Un proceso Supervisor y dos procesos NP.
Proceso *launcher*	Proceso de Numenta responsable de iniciar el NRE.
Procesos *orterun* y *orted*	Procesos asociados con la gestión de los procesos subyacentes y la interface de comunicación entre los procesos.

Tabla 2. Procesos NRE.

En el caso más simple, el NRE corre en una máquina con un solo CPU. Existen dos formas de ejecutar el NRE en este caso, tal como se muestra en la figura 19:

- SP-NuPIC – El Supervisor, un NP, y el programa de control se ejecutan todos en el mismo CPU en un solo proceso. Este es el modo por defecto.
- MP-NuPIC – Si se solicitan más de un NP, el NRE automáticamente se ejecuta en modo MP-NuPIC. En ese caso, se verá el proceso Supervisor, los dos procesos NP y los otros procesos mencionados previamente.

Si bien es posible ejecutar más de un NP en un solo CPU, el rendimiento disminuye porque la CPU tiene que ser compartido entre los NPs. Sin embargo, varias instancias NP pueden ser de utilidad durante la creación de prototipos.

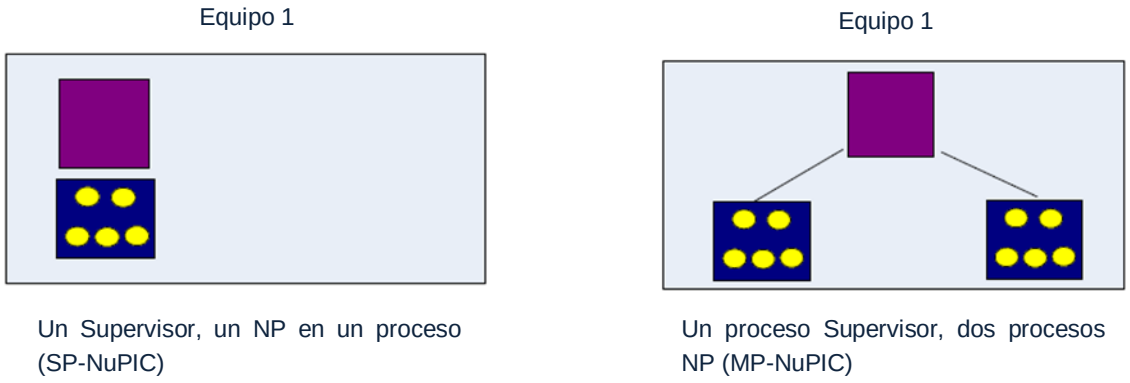


Figura 19. Diferentes configuraciones en computadoras multi-CPU.

La forma más sencilla de sacar provecho a los múltiples CPU es ejecutar varias instancias de la NRE, cada uno con una red HTM por separado. Si bien este enfoque es simple (tan simple que se conoce como “embarazosamente paralelo” en la literatura de cómputo de alto rendimiento) puede ser eficaz, y evita muchos de los potenciales problemas de procesamiento en paralelo.

En algunas situaciones, especialmente cuando se trabaja con un gran problema, utilizar un solo NRE con más de un NP puede mejorar la velocidad de ejecución.

3.3 Aplicaciones de una red HTM

3.3.1 Avances de desarrollo

¿Qué se puede hacer con la plataforma de Numenta? A través de muchas discusiones con los desarrolladores, se ha determinado que las redes HTM pueden potencialmente ser utilizadas en una amplia gama de aplicaciones. Por ejemplo, existe una clase de problemas similares al reconocimiento de contenido de una imagen que llamamos inferencia espacial. Dada una imagen o patrón novedoso, HTM dice lo que es. Los datos pueden provenir de una cámara u otro tipo de sensores [23]. En la figura 20 se observa que HTM discrimina correctamente entre vehículos en movimiento y un niño corriendo a través de la cámara.

Figura 20. Ejemplo del reconocimiento de patrones utilizando Numenta.

También se han visto muchas aplicaciones que modelan las redes, incluyendo las redes de computadoras, redes eléctricas, redes de máquinas e incluso las redes sociales. En estas aplicaciones, HTM aprende a reconocer o predecir las condiciones futuras indeseables en una red dada (en la siguiente sección se introduce una breve explicación de los trabajos que se han realizado al respecto).

La exploración petrolera constituye un ejemplo de la aplicación de HTM, ya que grandes cantidades de datos deben ser recopilados y analizados para decidir la mejor forma para perforar un pozo. Los laboratorios farmacéuticos están interesados en saber si los HTM les pueden ayudar a descubrir nuevos fármacos considerando los datos que se han acumulado en los ensayos con la droga.

HTM funcionan mejor cuando hay una estructura jerárquica de datos. Con muchas aplicaciones de HTM, la parte más complicada puede ser el momento de decidir cuáles son los datos “sensoriales” para el entrenamiento y cómo presentarlos de forma variable en el tiempo.

También se ha visto interese en el modelado de procesos de fabricación. Muchas industrias tienen una gran cantidad de datos y no es fácil descubrir los patrones en los datos o un medio para hacer predicciones sobre la base de experiencias pasadas.

Hoy en día hay aplicaciones que no se pueden resolver, dado el estado de la plataforma. Cualquier cosa que implique largas secuencias de memoria o un tiempo específico no puede ser soportada por los algoritmos de los nodos actuales. Por ejemplo, entender el lenguaje hablado, la música y la robótica requiere una sincronización precisa.

3.3.2 Trabajo previo: HTM en detección de anomalías en el tráfico de red

A pesar de que la teoría HTM aún se encuentra en desarrollo, su estudio es de gran relevancia en el área de la inteligencia artificial porque puede ser considerada como una opción para generar predicciones precisas de mundos inusuales como lo es la detección de intrusos mediante el análisis de tráfico de una red de datos.

Un ejemplo de ello es el trabajo de tesis titulado *Network Intrusion Detection, based on online traffic measurements using Hierarchical Temporal Memory Networks*, que investiga el uso de un novedoso enfoque de inteligencia artificial para la detección de intrusos basados en la identificación de anomalías del tráfico de red. Esta técnica de inteligencia artificial está basada en el paradigma de Memoria Temporal Jerárquica, que realiza funciones similares al neocortex de los cerebros humanos. Para su ejecución se evaluaron los datos de DARPA [24].

Por otro lado, existe la investigación desarrollado por Gerod M. Bonhoff [25], en el cual se explora la naturaleza del ciberespacio y se prueba la teoría HTM como un sistema de detección de anomalías y su habilidad de caracterizar el tráfico de red. En esta investigación se plantearon dos cuestiones principales: ¿puede HTM proporcionar una comprensión del mundo abstracto del ciberespacio? (ver figura 21), es decir, ¿puede una red HTM mostrar que ha aprendido patrones espacio-temporal en el tráfico de red a pesar de la falta de un maestro humano que pueda reconocer los mismos patrones? ¿Pueden esos patrones aprendidos ser utilizados posteriormente por la red HTM para hacer predicciones precisas, inteligentes y tomar decisiones? La segunda pregunta, ¿las bases de predicción son apoyadas por HTM?

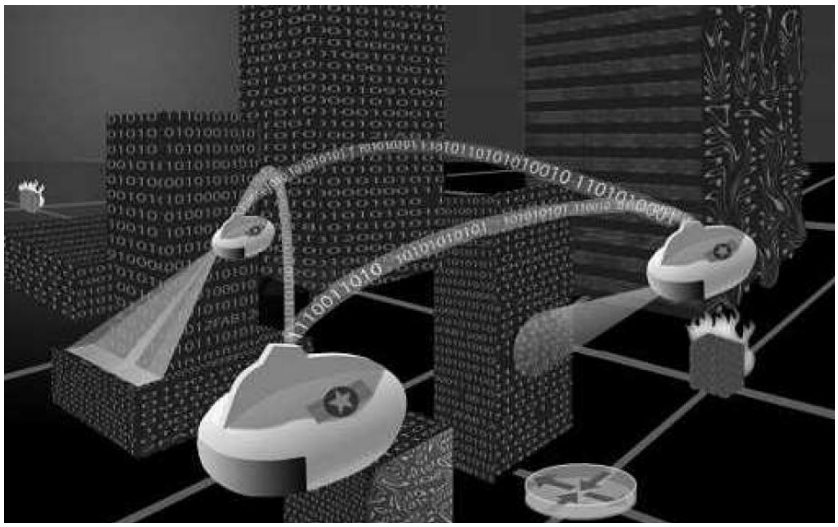


Figura 21. Interpretación artística del ciberespacio.

Asimismo, se cuenta con el trabajo de tesis *"Sistema de Detección de Intrusos Basado en Anomalías de Red Usando la Plataforma Numenta para Cómputo Inteligente"* [26] en el que se utilizaron datos de prueba proporcionados por DARPA así como datos de una red en producción para detectar ataques informáticos.

Otro esfuerzo más por entender el funcionamiento del paradigma HTM en la creación y funcionamiento de un Sistema de Detección de Intrusos fue el trabajo de tesis titulado "Sistema para Detección de Intrusos en Línea Basado en Anomalías de Red, Usando Redes de Memoria Temporal Jerárquica" [27]. En este trabajo se expusieron dos tipos de sistemas detectores de intrusos basados en anomalías, uno *pasivo* a través del uso de lotes de archivos que simulaban una red en producción, y *en línea*, utilizando un equipo instalado en la red de comunicaciones de una empresa.

3.3.2.1 Ataques detectados por los IDS

Considerado como un mecanismo de protección complementario, el IDS no se basa en mecanismos específicos sino en el monitoreo continuo de la red para detectar actividades inusuales. El monitoreo del sistema generalmente se desarrollará en tres fases: la recolección de datos, el análisis de datos recopilados y, finalmente, la generación de una alerta cuando se detecta una amenaza [28], [29].

Las diferencias entre los IDS actuales residen en el diseño de las tres fases, es decir, cómo y dónde recopila los datos, cómo busca las intrusiones de los datos recopilados, y cómo responde a las intrusiones. Los sistemas de detección de intrusos de acuerdo con sus métodos, pueden estar basados en usos indebidos (también llamado basado en conocimiento) o basado en anomalías (basado en comportamiento).

Detección de uso indebido (ver figura 22). Basa sus mecanismos de detección de ataques en los ya conocidos. Comúnmente toma los datos de auditoría para su análisis y los compara con grandes bases de datos de firmas de ataques. Los datos de auditoría pueden ser obtenidos de bitácoras de la red, de aplicaciones [30].

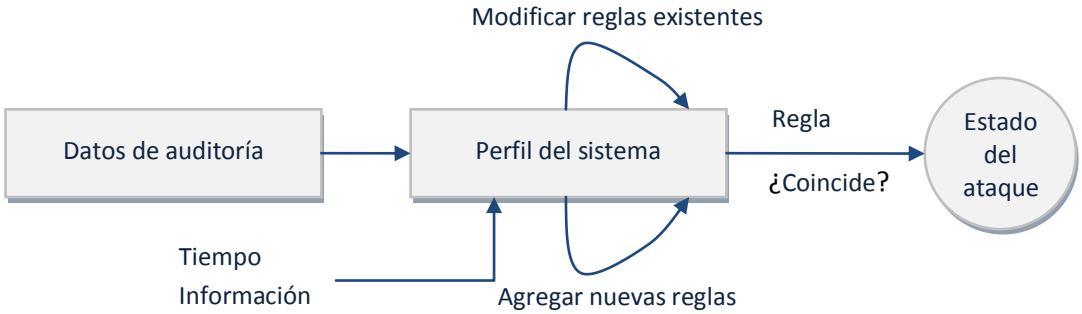


Figura 22. Un típico sistema de detección de uso indebido.

La ventaja de los sistemas de intrusos basados en uso indebido es que, en teoría tienen una menor tasa de falsas alarmas. Y el proceso de análisis de alarmas facilita a un administrador de seguridad entender y reaccionar rápidamente. La desventaja de este método de detección es que no es fácil mantener actualizada la base de conocimiento de tal sistema de detección de intrusos, además de que la mayoría de los ataques dependen del sistema operativo, versión, plataforma y aplicación.

Detección de anomalías (ver figura 23). Basa sus mecanismos en el perfil de la conducta normal del usuario. Analiza la sesión actual del usuario y lo compara con el perfil que representa el comportamiento normal del usuario estadísticamente. Otro enfoque basado en anomalías, toma en cuenta la predicción del comportamiento futuro del sistema considerando su historia. Si bien estos métodos son más exitosos en la captura de correlaciones temporales y múltiples variables, requieren más tiempo para entrenar el modelo y, en algunos casos, su aplicación puede ser inviable debido al tamaño del conjunto de datos implicados [31], [32].

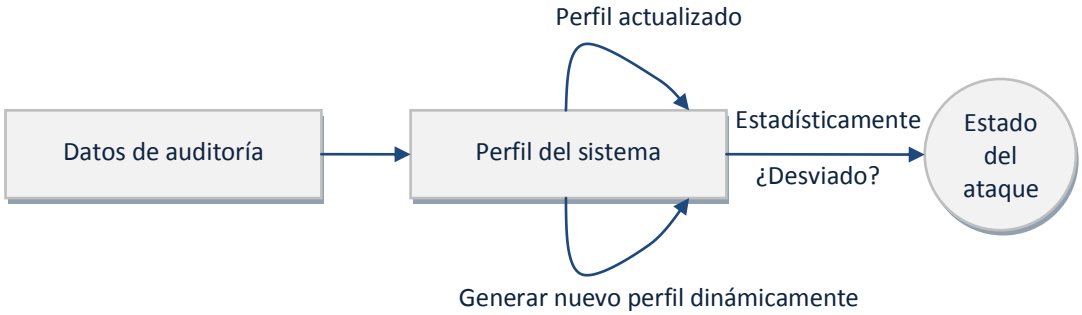


Figura 23. Un típico sistema de detección basado en anomalías.

Permite identificar nuevos tipos de intrusión o previamente no encontrados, como desviación del uso normal. Sin embargo, es alto el porcentaje de falsas alarmas, es decir, los comportamientos del sistema (aunque legítimos) pueden ser reconocidos como anomalías, y por lo tanto, ser considerados como posibles intrusiones.

Otras categorías de IDS se han definido en la literatura. Una tercera categoría denominada “Detección basado en la especificación” ha sido definida por Mishra et al. [33] como un conjunto de restricciones que describen el funcionamiento de un programa o protocolo y el seguimiento de su ejecución.

En pocas palabras, las herramientas IDS permiten la supervisión completa de las redes, independientemente de las medidas tomadas, de tal manera que la información siempre existirá para determinar la naturaleza del incidente de seguridad y su origen. Los IDS comúnmente reportan tres tipos de ataques informáticos: escaneo del sistema, denegación de servicio (DoS) y la penetración del sistema [33].

- **Escaneo**

Es el reconocimiento de la red o de un equipo en particular mediante el envío de diferentes tipos de paquetes. Utilizando la respuesta recibida del objetivo, el atacante puede aprender muchas características del sistema y vulnerabilidades. Este tipo de ataques no penetran o comprometen los sistemas.

Un ejemplo es el escaneo de puertos cuyo propósito es determinar cuáles son los puertos que están abiertos, y por lo tanto, los servicios que se están ejecutando. Detecta si está abierto, cerrado o protegido [34]. Existen varios programas que permiten escanear los puertos de la red. Uno de los más conocidos es Nmap.

Nmap [35] ("mapeador de redes") es una herramienta de código abierto para la exploración de red y auditoría de seguridad. Se diseñó para analizar rápidamente grandes redes, aunque funciona muy bien contra equipos individuales. Nmap utiliza paquetes IP "crudos" para determinar qué equipos se encuentran disponibles en una red, qué servicios (nombre y versión de la aplicación) ofrecen, qué sistemas operativos ejecutan así como docenas de otras características [35]. Para ejecutar un escaneo de puertos se utiliza el siguiente comando:

nmap –sS –A –p 1-65535 host<target>

-sS: Técnica de escaneo TCP SYN
-A: Habilita detección de Sistema Operativo, detección de versión
-p: Escanea los puertos especificados
host<target>: Nombre del equipo o dirección IP objetivo

Hping3 [36] es una herramienta encargada de generar y analizar paquetes TCP/IP, permitiendo falsificar la información enviada con el objetivo de realizar auditorías de seguridad. Es utilizada entre otras cosas para realizar un escaneo de puertos avanzados, tal como se muestra a continuación:

hping3 –scan '1-65535' –S host<target>

-scan: Modo de escaneo, describe los grupos de puertos a escanear
-S: Activa la bandera SYN de TCP
host<target>: Nombre del equipo o dirección IP objetivo

Denegación de servicio (DoS)

Es un ataque en el que el intruso hace que algunos recursos de memoria o cómputo estén demasiados ocupados o demasiado lleno para atender las peticiones legítimas, o niega el acceso a los usuarios legítimos a una máquina.

Hay muchas variedades de ataques de denegación de servicio [37]. Algunos ataques DoS (mailbomb, smurf) abusan de una función legítima. Otros (ping de la muerte) crean paquetes malformados que confunden a la pila TCP/IP o aprovechan los defectos de programación. Existen unos que consumen todo el ancho de banda o los recursos del sistema (principalmente CPU, memoria, espacio de almacenamiento. Sin olvidar los que son ocasionados por la destrucción física o alteración de los componentes de red [38].

Los ataques de Denegación de Servicio (DoS) y de Denegación de Servicio Distribuido se han tornado cada vez más frecuentes, y muchas veces son utilizados por los hacktivistas como medio de protesta, o incluso se ejecutan mediante redes botnets que utilizan a sus equipos víctimas con esa finalidad [39]. Estos ataques se llevan a cabo mediante el uso de herramientas que envían una gran cantidad de paquetes de forma automática para desbordar los recursos del servidor, logrando que el propio servicio quede inoperable o incapaz de atender todas las solicitudes.

LOIC (Low Orbit Ion Cannon) [40] es una aplicación diseñada para realizar un ataque de denegación de servicio utilizando el lenguaje de programación C#. La aplicación (ver figura 24) realiza un ataque de denegación de servicio del objetivo enviando una gran cantidad de peticiones por segundo.

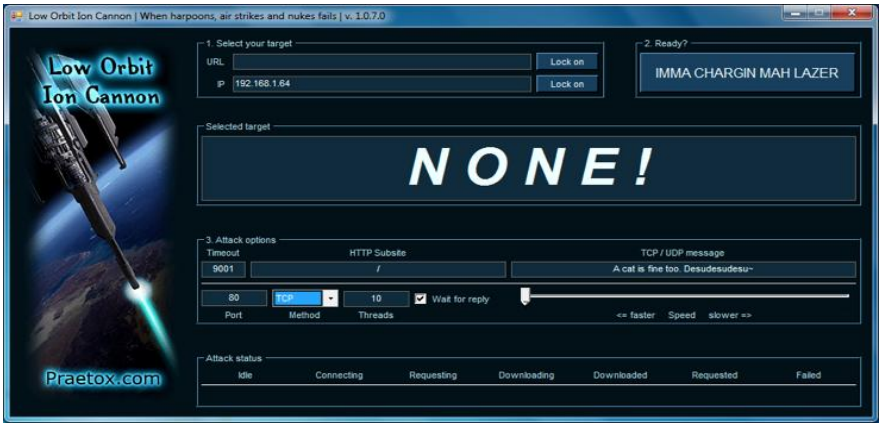


Figura 24. LOIC herramienta utilizada para simular un ataque de DoS.

- **Ataques de penetración**

Es un ataque que implica la adquisición no autorizada y/o alteración de los privilegios, recursos o datos del sistema. Se producen violaciones de integridad y control, a diferencia de los ataques DoS que violan la disponibilidad de un recurso y los ataques de escaneo que no hacen algo ilegal. Un ataque de penetración puede obtener el control de un sistema mediante la explotación de defectos de software [33].

Después de explicar el proceso de desarrollo de una red HTM utilizando NuPIC, se dará a conocer el diseño y configuración de parámetros seleccionados para crear el IDS basado en anomalías capaz de detectar incidentes de seguridad informáticos.

Capítulo 4

Diseño e implementación de un IDS basado en anomalías

En las siguientes secciones se describirá la configuración experimental diseñada para la identificación de incidentes de seguridad informáticos, así como el desarrollo de los procedimientos de entrenamiento, ejecución de pruebas y detección de anomalías utilizando la plataforma Numenta.

4.1 Análisis del problema

Hoy en día, muchas organizaciones utilizan los servicios de Internet para comunicarse y lugar para hacer negocios. Junto con el crecimiento de las actividades de las redes de computadoras, la tasa de crecimiento de los ataques informáticos ha ido avanzando, impactando en la disponibilidad, confidencialidad e integridad de la información. Por lo tanto, en una red se debe usar una o más herramientas de seguridad como cortafuegos, antivirus, IDS e IPS para proteger la información.

Recientemente, se han propuesto un gran número de enfoques innovadores y nuevos modelos de IDS. La mayoría de ellos se enfrentan a dificultades principales: baja precisión de detección, bajo rendimiento en tiempo real y, la escalabilidad limitada debido a la gran cantidad de datos que se relacionan, así como los comportamientos y técnicas de ataque complejos [41]. Sin embargo, los sistemas de detección de intrusos siguen siendo el único medio proactivo de detección y respuesta a las amenazas que provienen de dentro y fuera de una red de datos [42].

Un sistema de detección de intrusos (IDS) [43], [33] como el que se muestra en la figura 25, es una herramienta de prevención que proporciona una señal de alarma a un usuario de una computadora o al administrador de red mediante la inspección de actividades sospechosas en la red.

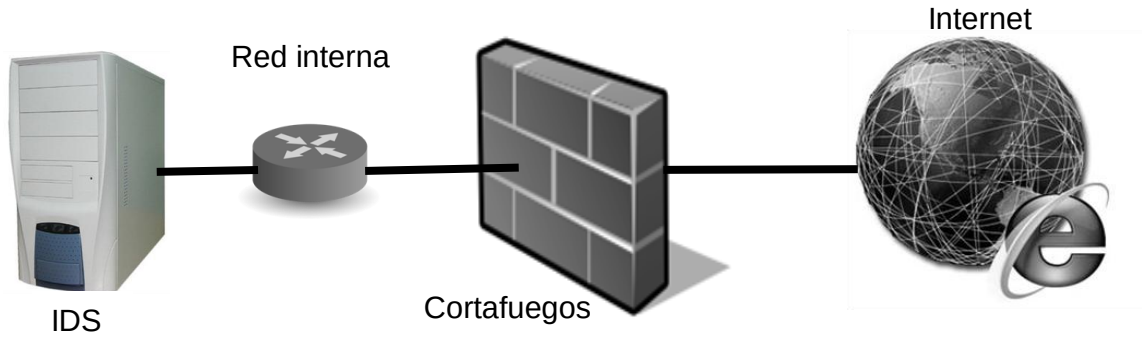


Figura 25. Sistema de detección de intrusos.

4.1.1 ¿Los ataques generan anomalías?

Se cree que atacar el tráfico de red genera anomalías por cuatro razones [44]. En primer lugar, los ataques que se aprovechan de errores en el software de la víctima, de alguna manera deberían ser inusuales, ya que cualquier error que es evocado por el tráfico normal tardaría mucho tiempo en ser detectado y corregido. En segundo lugar, un atacante podría generar anomalías por descuido, por ejemplo, poner valores inusuales pero legales en los campos de la cabecera TCP/IP cuando un ataque requiere la programación con sockets. En tercer lugar, las pruebas son necesariamente anómalas debido a que el atacante estará buscando información que los usuarios legítimos conocen, quien normalmente no intenta adivinar las contraseñas o trata de conectarse a direcciones IP o puertos inexistentes. Cuarto, un atacante deliberadamente podría insertar datos inusuales para confundir al IDS.

Derivado de lo anterior, surge la necesidad de desarrollar e implementar nuevas técnicas y metodologías para la detección de incidentes, tal como se expone en este trabajo de tesis, en el que mediante un nuevo paradigma de programación conocido como HTM se implementa un IDS basado en anomalías y se proporciona información del incidente con el objetivo de tomar las medidas de seguridad necesarias y así minimizar el impacto dentro de las organizaciones.

4.1.2 Consideraciones

Para el desarrollo de este trabajo se establecen las siguientes afirmaciones que permitirán modelar los datos para la implementación del IDS basado en anomalías:

- ✓ Se hace la suposición de que existen patrones espacio-temporales en los datos que representan el medio.

- ✓ Coherencia entre los datos de entrenamiento y prueba. Tanto los datos utilizados para crear un modelo de la red HTM y los utilizados para probar ese modelo deberán asumir que representan con exactitud el mismo mundo.

- ✓ Por último, existe la idea de que la manipulación de datos preserva la integridad de los patrones, es decir, el filtro o análisis de los datos TCPdump no deberá alterar los patrones espacio-temporal.

4.2 Configuración experimental: Identificación de incidentes

Hoy en día los ataques cibernéticos lejos de ser un hecho de ciencia ficción, son una realidad cotidiana para las empresas, y es por ello que se han desarrollado diversas herramientas que se utilizan para detectar ataques de seguridad informática, ejemplo de ello son los Sistemas de Detección de Intrusos (IDS).

En este experimento se utilizará la plataforma Numenta, para desarrollar un IDS basado en anomalías utilizando la teoría HTM (Memoria Temporal Jerárquica) (tecnología que propone varios mecanismos novedosos de aprendizaje que tratan de capturar la forma en el que el cerebro humano aprende e infiere de su entorno) que sea capaz de identificar los incidentes ocurridos dentro de una red de datos, tal como se describe a continuación.

4.2.1 Configuración de la red HTM

La construcción de la topología de la red jerárquica es el primer paso en la creación de una red HTM. Fueron utilizados cinco tipos de nodos NUPIC para la construcción de la red de detección de anomalías, los cuales se describen en la tabla 3:

Tipo nodo	Descripción
Nodo sensor – VectorFileSensor	Toma los datos del archivo de captura del tráfico de red y los convierte en la señal de entrada, para ello transforma los datos al lenguaje que la red HTM entiende.
Nodo de memoria – Zeta1Node	Implementa el algoritmo de aprendizaje Zeta1, analiza los datos recibidos de los nodos inferiores, ejecuta los algoritmos de clasificación espacial y temporal, envía el resultado a cualquier número de nodos superiores.
Nodos principales – Zeta1TopNode	Es el más alto nivel de nodos aprendizaje que se conecta con los nodos efectores.

Nodos Efectores – VectorFileEffector	Proporciona un modo significativo del uso de las habilidades de predicción y categorización de las redes, combina la información sensorial con categorías asignadas y las escribe en un archivo.
Otro – PassThroughNodes	Conectan la entrada directamente a la salida sin la manipulación o el aprendizaje.

Tabla 3. Tipos de nodos utilizados para la creación de la red HTM.

Para llevar a cabo el preprocesamiento de datos se generaron vectores de 33 elementos que representaban cada paquete del tráfico de red, tal como se muestra en la figura 26. Se consideraron inicialmente 16 campos del encabezado de los paquetes TCP/IP y Ethernet para el análisis con NuPIC. Se añadió un número de identificación para referencia del paquete como el primer elemento de cada vector, el cual junto con la categoría creada se almacenó en un archivo de salida.

En la figura 27 se muestra la configuración de los nodos NUPIC para la construcción de la red HTM que implementa el IDS basado en anomalías. Cada campo de los encabezados de los paquetes de red alimentan a un nodo de memoria Zeta1Node en el nivel 1. La salida de estos nodos a su vez alimenta a los nodos de memoria del nivel 2 que representa a la capa de protocolo de red de la que los campos se derivaron (TCP/IP, Ethernet). La salida de los nodos del nivel 2 se combina con la información del tiempo delta (desde el nivel 1) en el nodo principal Zeta1TopNode del nivel 3. La salida del nodo principal se conecta con un nodo efector VectorFileEffector que combina las categorías de salida de las redes con el número ID (primer elemento de cada vector) a través de los nodos PassThroughNodes presentes en cada nivel de la jerarquía.

Adicionalmente, cada uno de los nodos seleccionados debe ser configurado de acuerdo a las características de los datos de entrada, por lo que se establecieron ciertos parámetros que se describen en la siguiente sección.

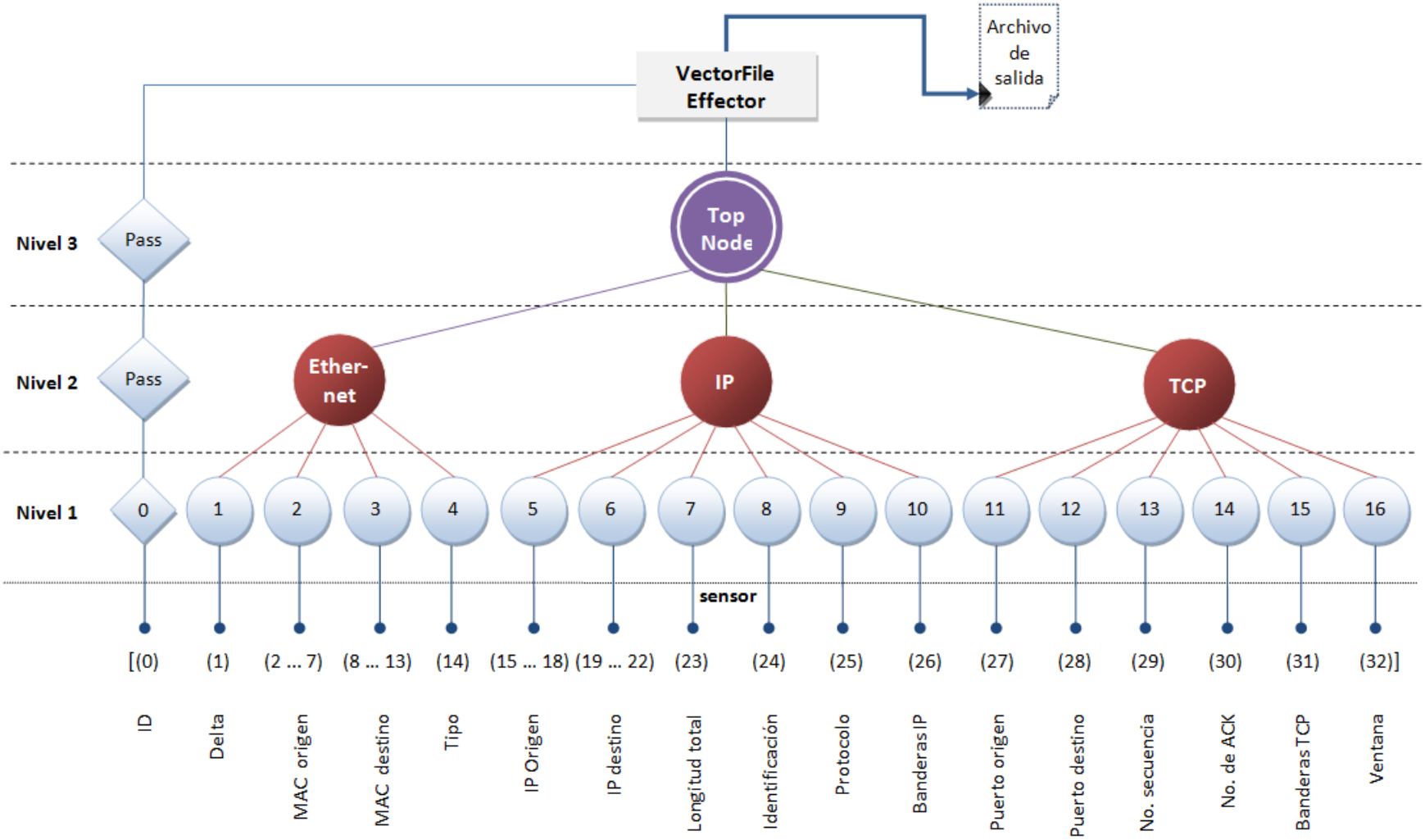


Figura 26. Configuración de la red HTM implementada para la creación de un IDS basado en anomalías.

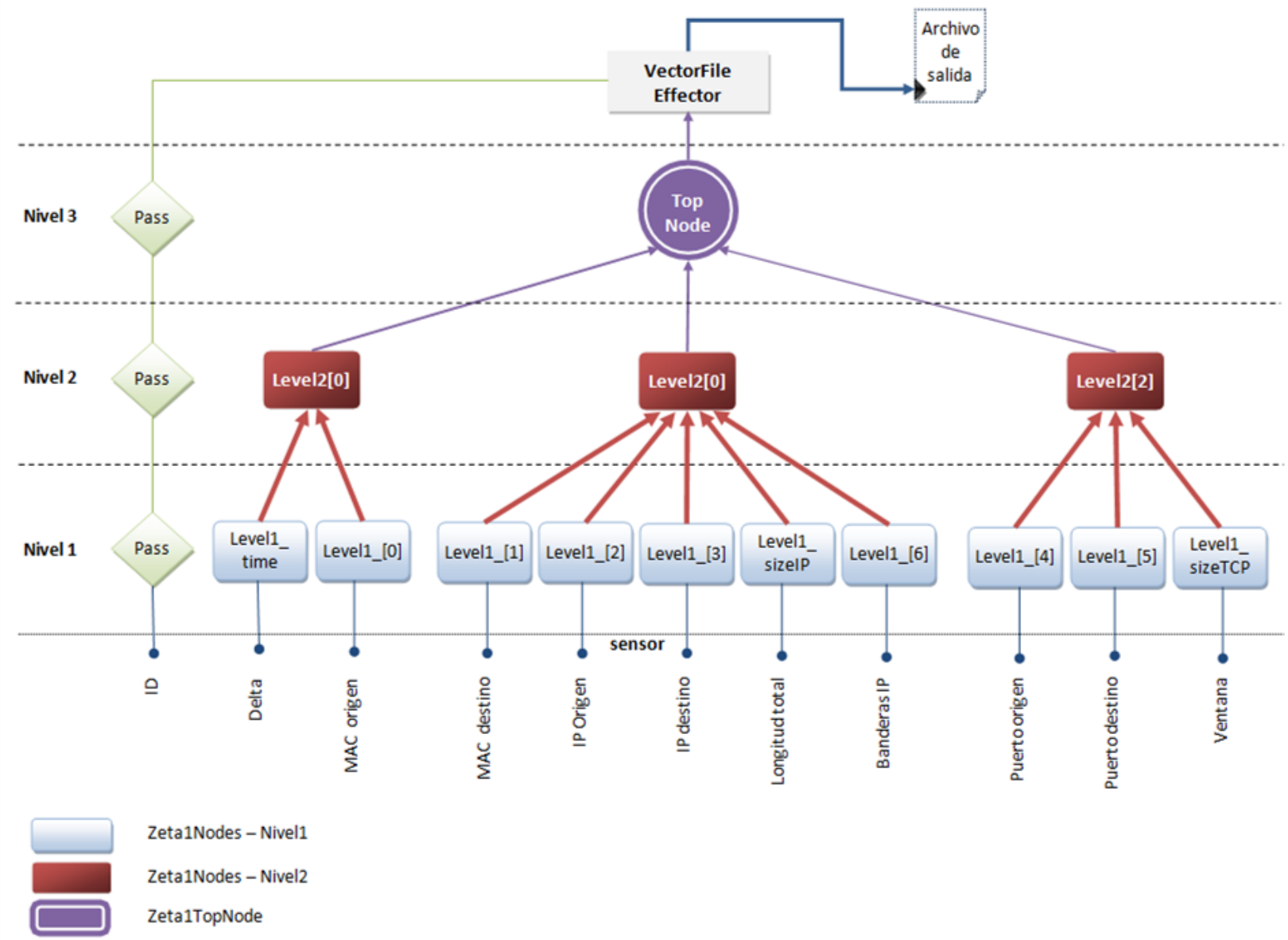


Figura 27. Configuración de los nodos NUPIC para crear la red HTM.

4.2.2 Especificaciones y parámetros

Para llevar a cabo el experimento que se describió en la sección anterior, se configuraron los parámetros de los nodos de memoria *maxDistance*, *spatialPoolerAlgoritm*, *symmetricTime*, *temporalPoolerAlgorithm* (ver tabla 4) [15]:

Parámetro	Descripción	Valor
maxDistance	Establece la distancia euclidiana[3] máxima en el que dos vectores de entrada se consideran el mismo durante el aprendizaje.	*maxDistance* = 0.0 *Justificación:* No hay dos paquetes que se consideren sean idénticos.
spatialPooler-Algorithm	Permite seleccionar el método de coincidencias en la detección de la inferencia. El algoritmo para los nodos de nivel inferior (por encima del sensor) es siempre Gaussiana.	*spatialPooler-Algorithm* = gaussian *Justificación*: Con este algoritmo "cada coincidencia de salida tiene una puntuación similar a la que se basa en la distancia euclidiana entre la entrada y la coincidencia".
symmetricTime	Toma un valor booleano que, si es cierto, el algoritmo asume que si los datos llegan en un cierto orden es igual de probable que lleguen en el orden inverso.	*symetricTime* = False *Justificación*: El hecho de que un paquete es seguido por otro no es suficiente razón para asumir que el orden inverso es correcto.
temporalPooler-Algorithm	Permite seleccionar el método para el cómputo de probabilidades de salida.	*temporalPoolerAlgorithm*: sumProp *Justificación*: Cuando se selecciona maxProp, el nodo calcula la puntuación más alta para crear los grupos temporales.

Tabla 4. Parámetros de los nodos de memoria – Zeta1Nodes.

[3] Distancia euclidiana es la distancia "ordinaria" entre dos puntos de un espacio euclídeo, que equivale a la longitud del segmento de recta que los une.

Asimismo, fue necesario definir la ubicación de los archivos utilizados como entrada (datos de entrenamiento, de prueba) y salida de la red HTM (red no entrenada, red entrenada, inferencia), y precisar los nombres de los archivos que se generaron, utilizando rutas relativas para poder ejecutar el programa en cualquier directorio, tal como se muestra a continuación (ver figura 28):

```
untrainedNetwork        = birthingPlace + "untrained_cybercraft.xml"   # Nombre de la red no entrenada
trainedNetwork          = birthingPlace + "trained_cybercraft.xml"     # Nombre de la red entrenada
trainingFile            = dataLocation  + trainingData + ".snr"        # Ubicacion de los datos de entrenamiento
trainingResults         = birthingPlace + "training_results.txt"       # Archivo que contiene la inferencia
testFile                = dataLocation  + testingData + ".snr"         # Ubicacion de los datos de prueba
testResults             = birthingPlace + "test_results.txt"           # Archivo que contiene la inferencia
```

Figura 28. Ubicación de archivos utilizados por la red HTM.

4.2.3 Datos de entrada

Para generar los datos de entrada a la red HTM en el formato vector se deben filtrar los archivos de captura de tráfico de red (archivo .tcpdump), eliminando los paquetes sin conexión TCP/IP y Ethernet. Para ello se desarrolló un script que permitiera generar los vectores de forma automática.

Como se muestra en la figura 29, se configuraron diferentes parámetros para facilitar la captura y conversión de datos. Se selecciona el directorio raíz de los archivos de tráfico de red, si no existe, se crea; se especifica el número de vectores o tramas TCP/IP que tendrá el archivo y finalmente, se utiliza el comando tcpdump para procesar el archivo de captura.

```
debian:/home/rsarabia/nta/current/share/projects/tesis09012012# time ./RunStart.pl
================================================
BIENVENIDOS A LA RED HTM
================================================
Directorio raiz de los archivos de trafico de red: /home/rsarabia/nta/nupic-1.7.1-linux32/share/projects/tesis09012012/archivos
Nombre del directorio a utilizar [trafico_010512]:
El directorio no existe, desea crearlo [S/N]? S
El directorio trafico_010512 fue creado
Existe el archivo de trafico de red [S/N]? N
Numero de archivos de trafico de red a crear [1]:
Numero de tramas TCP/IP tendra [2500]: 1000
Numero de captura de inicio [1]:
Prefijo de los archivos a generar [salida]: test
Generando 1 con 1000 tramas TCP/IP ...
tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 96 bytes
1000 packets captured
1001 packets received by filter
0 packets dropped by kernel
Archivo test1.snr creado ........................... [OK]
Proceso completado ................... [OK]
```

Figura 29. Generación de tráfico de red para entrada al sensor.

El comando tcpdump sirve para procesar el archivo de tráfico de red, si se cuenta previamente con el archivo sólo se indica el directorio donde se ubica y se utilizan las siguiente opciones: sudo para obtener los privilegios de root, –r para indicar el archivo de captura, –nn para no convertir las direcciones a nombres (direcciones de los equipos, números de puertos), –ttt para imprimir un delta en microsegundos, –xx para imprimir los datos de cada paquete seleccionando solo aquellos que pertenezcan al protocolo TCP/IP y Ethernet y el resultado se guardará en un archivo de salida.

```
`sudo tcpdump -r $pref -nn -ttt -xx 'tcp and ip' | egrep
      "IP|0x0000|0x0010|0x0020|0x0030" > $output`;
```

En caso contrario, si no se cuenta con el archivo, se utilizará el comando tcpdump considerando otras opciones: –i para especificar la interfaz donde se capturará el tráfico de red, –v para hacerlo reiterativo, –c para definir el número máximo de paquetes que contendrá el archivo de salida.

```
`sudo tcpdump -i eth0 -v -c $tramas -nn -ttt -xx 'tcp and ip' | egrep
      "IP|0x0000|0x0010|0x0020|0x0030" > $output`;
```

Adicionalmente, es necesario hacer una transformación de los datos para convertir las representaciones hexadecimales en representaciones enteras o flotantes, colocarlas ordenadamente en renglones en forma de vector para que sean aceptados por el nodo sensor de la red HTM (ver anexo A). El seudocódigo utilizado para el preprocesamiento de datos se muestra en la tabla 5:

Algoritmo 1. Generación de datos de entrada al sensor

1. INICIO
2. inicializar variables
3. establecer ID en 1
4. para cada vector en el archivo de entrada al sensor hacer
5. obtener el valor de los 32 campos vectoriales
6. convertir datos hexadecimal a decimal
7. almacenar ID y los 32 campos vectoriales en el archivo de salida
8. incrementar ID
9. fin para cada
10. FIN

Tabla 5. Seudocódigo para el preprocesamiento de datos de entrada al sensor.

El nodo sensor utilizado para desarrollar la red HTM descrita en la sección 4.2.1, es el VectorFileSensor, que lee vectores del archivo de salida del preprocesamiento de datos descritos previamente en la sección 4.2.3. A excepción de la primera línea del archivo, cada línea es un vector y cada elemento del vector debe ser de punto flotante separado por un solo espacio. La primera línea del archivo de datos instruye al VectorFileSensor sobre el número de elementos que existen en cada vector.

En la figura 30 se muestra un ejemplo del formato VectorFileSensor con 15 vectores (15 renglones), la primera línea indica el número de elementos que existe en cada vector, en este caso son 33 campos. Si tomamos como referencia el primer vector (recuadro rojo), se tienen los siguientes valores:

- ID: 1
- Delta: 0.000000
- MAC origen: 00-21-7C-3B-83-71 (0 33 124 59 131 113)
- MAC destino: 00-11-11-85-C9-BD (0 17 17 133 201 189)
- Tipo: 2048
- IP origen: 173.194.26.16
- IP destino: 192.168.1.70
- Longitud total: 1492
- Identificación: 55468
- Protocolo: 6 (TCP)
- Banderas IP: 64
- Puerto origen: 80 (web http)
- Puerto destino: 49386
- No. Secuencia: 2181041927
- No. de ACK: 477359252
- Banderas TCP: 16
- Ventana: 129

```
33
1 0.000000 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55468 6 64 80 49386 2181041927 477359252 16 129
2 0.013922 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55469 6 64 80 49386 2181043379 477359252 16 129
3 0.013562 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55470 6 64 80 49386 2181044831 477359252 16 129
4 0.011978 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55471 6 64 80 49386 2181046283 477359252 16 129
5 0.013722 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55472 6 64 80 49386 2181047735 477359252 16 129
6 0.008512 0 17 17 133 201 189 0 33 124 59 131 113 2048 192 168 1 70 173 194 26 16 40 29276 6 64 49386 80 477359252 2181047735 16 65340
7 0.005337 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55473 6 64 80 49386 2181049187 477359252 16 129
8 0.013915 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55474 6 64 80 49386 2181050639 477359252 16 129
9 0.011700 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55475 6 64 80 49386 2181052091 477359252 16 129
10 0.013741 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55476 6 64 80 49386 2181053543 477359252 16 129
11 0.013766 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55477 6 64 80 49386 2181054995 477359252 16 129
12 0.013863 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55478 6 64 80 49386 2181056447 477359252 16 129
13 0.011891 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55479 6 64 80 49386 2181057899 477359252 16 129
14 0.013720 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55480 6 64 80 49386 2181059351 477359252 16 129
15 0.013720 0 33 124 59 131 113 0 17 17 133 201 189 2048 173 194 26 16 192 168 1 70 1492 55481 6 64 80 49386 2181060803 477359252 16 129
```

Figura 30. Formato VectorFileSensor.

4.2.4 Procedimientos

Los procedimientos adoptados para el experimento fueron bastante sencillos, los cuales se describen a continuación.

4.2.4.1 Entrenamiento

Después de seleccionar los datos de un conjunto libre de ataques que simulan un modelo de tráfico normal, se genera el archivo de entrada al sensor para iniciar el entrenamiento que se llevará a cabo sin supervisión.

Primero, se especifica la ruta de ubicación del archivo de entrada al sensor y el número de vectores a entrenar, los cuales fueron variando en los diversos experimentos para hacer una comparativa de funcionalidad y tiempo de ejecución, por ejemplo, en la figura 31 se observa la ejecución del programa de una red entrenada con 200,000 vectores.

```
Te gustaria volver a entrenar a la red? [s/n] >> s

Entrenando la red HTM Network con 200000 vectores

Entrenando nivel 1...
Entrenando nivel 2...
Entrenando nivel 3...
Entrenamiento completo
Entrenamiento completo de la red HTM.

Validando la intuicion autonoma de la red...

Ejecutando inferencia...
Inferencia completa; resultados en  /home/rsarabia/nta/nupic-1.7.1-linux32/share/projects/entrenamiento04042012/d
ta_200000-test_data_1167661/training_results.txt
```

Figura 31. Entrenamiento de la red HTM.

Al concluir los tres niveles de entrenamiento, el resultado se almacena en un archivo de salida “training_results.txt”, conformado por vectores de dos elementos, en donde el primer elemento es la categoría asignada y el segundo elemento es el número de identificación del paquete, tal como se muestra en la figura 32.

```
Los numeros muestran el nodo de salida del nivel 3 seguido de la salida del sensor
36.7384  1
39.1908  2
38.191  3
38.191  4
37.7183  5
38.2486  6
37.7183  7
38.188  8
38.188  9
```

Figura 32. Resultados del entrenamiento de la red HTM (training_results.txt).

4.2.4.2 Pruebas

Una vez que la red HTM ha sido entrenada, se seleccionan los datos de un conjunto de paquetes con tráfico malicioso, en este trabajo de tesis se seleccionó el escaneo de puertos y denegación de servicio por ser de los ataques que frecuentemente se utilizan hoy en día.

El objetivo de realizar las pruebas es verificar que el IDS implementado con esta novedosa teoría HTM sea capaz de identificar los incidentes informáticos. Para ello, se especifica el archivo de datos de entrada al sensor así como el número de vectores que se probarán. El resultado de las nuevas categorías creadas es almacenado en el archivo “test_results.txt”, tal como se muestra en la figura 33.

```
 Te gustaria probar la red? [s/n] >> s

 Comienzo de la clasificacion de los paquetes de datos de prueba con 1167661 vectores...
2012-04-06 23:29:21.827028
 Ejecutando inferencia...
        Inferencia completa; resultados en  /home/rsarabia/nta/nupic-1.7.1-linux32/share/projects/entrenamiento(
ining_data_200000-test_data_1167661/test_results.txt
 Todos los datos de prueba han sido clasificados.
2012-04-09 10:59:52.640374
```

Figura 33. Ejecución de las pruebas de la red HTM.

Al concluir las pruebas, el resultado se almacena en un archivo de salida “test_results.txt”, que al igual que el archivo resultado del entrenamiento está conformado por vectores de dos elementos, en donde el primer elemento es la categoría asignada y el segundo elemento es el número de identificación del paquete.

4.2.4.3 Detección de anomalías

Si un vector de datos de prueba no se encuentra en cualquiera de los grupos obtenidos durante la fase de entrenamiento, se considerará una anomalía. Para llevar a cabo la identificación de anomalías se hace la comparación de los archivos generados durante el entrenamiento (training_results.txt) y pruebas (test_results.txt) en busca de nuevas categorías. El algoritmo utilizado para la detección de anomalías se muestra a continuación (ver tabla 6).

Algoritmo 2. Detección de anomalías

```
1.  INICIO
2.  inicializar variables
3.  leer la categoría de cada vector de entrenamiento
4.  leer la categoría de cada vector de prueba
5.  almacenar las categorías de prueba en un arreglo temporal
6.  para cada categoría en el arreglo temporal hacer
7.          si categoría = categoría de entrenamiento entonces
8.                  eliminar categoría en el arreglo temporal
9.          si no continuar revisando
10.         incrementar ID
11.         fin si
12. fin para cada
13. anomalías = arreglo temporal
14. FIN
```

Tabla 6. Seudocódigo para la detección de anomalías

La implementación del algoritmo descrito para la detección de anomalías se encuentra en el anexo B, obteniendo como resultado el archivo “anomalias.txt”, tal como se muestra en la figura 34.

```
debian:/home/rsarabia/nta/current/share/projects/entrenamiento21022012# time ./detection.pl
============================================================
                    IDENTIFICACION DE INCIDENTES
============================================================

 Identificando nuevas categorias...
 Nuevas categorias identificadas en archivo 'anomalias.txt'... [OK]
```

Figura 34. Detección de anomalías

4.2.4.4 Identificación de incidentes

Finalmente, se obtiene información de las nuevas categorías creadas durante las pruebas (anomalías) para determinar si están relacionadas con algún incidente de seguridad como ataques de denegación de servicio o escaneo de puertos, los cuales, como se mencionó previamente, se deberían poder identificar ya que no representan un comportamiento normal de la red.

Las categorías identificadas como anomalías (“anomalias.txt”) se buscan en el archivo “test_results.txt” que es el archivo que contiene el resultado de las pruebas, de ahí se obtiene el ID y se relaciona con el ID del archivo “test_data.snr” que contiene los 32 campos de cada vector, tal como se muestra en el diagrama 35.

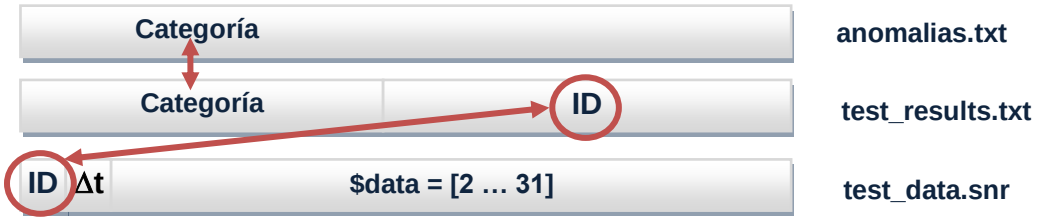


Figura 35. Identificación de incidentes.

A continuación se muestra el pseudocódigo utilizado para la identificación de incidentes (ver tabla 7):

Algoritmo 3. Identificación de incidentes

```
1.  INICIO
2.  inicializar variables
3.  leer categoría de archivo “anomalias.txt”
4.  si categoría = categoría en archivo “test_results.txt” entonces
5.          almacenar el ID de cada vector y el número de coincidencias
6.  fin si
7.  ordenar las categorías de acuerdo al número de coincidencia
8.  definir parámetros para identificar incidentes
9.  para cada categoría hacer
10.         para cada ID hacer
11.                 obtener el valor de campos del archivo “test_data.snr”
12.                 si valor = parámetros entonces
13.                         generar una alerta de seguridad
14.                 si no continuar revisando
```

```
15.                     incrementar ID
16.             fin para cada
17.             incrementar categoría
18.     fin para cada
19.     FIN
```

Tabla 7. Pseudocódigo para la identificación de incidentes.

La implementación del pseudocódigo se detalla en el anexo B, y el resultado se muestra en la figura 36. Para cada categoría anómala se obtuvo el número de paquetes que están relacionados, la suma del tiempo delta, puertos origen, longitud total, banderas, IP origen y destino.

Al analizar estos datos, el administrador de red podría identificar si se tratara de un incidente, por ejemplo, si existe un gran número de solicitudes al puerto 80, se podría inferir que se trata de un ataque de denegación de servicio por la gran cantidad de solicitudes que se realizan para que el servidor web no pueda atender a los usuario legítimos. O bien, si aparece una gran cantidad de puertos se podría inferir que se trata de un escaneo de puertos, ya que un usuario legítimo sólo realizaría consultas sobre los servicios conocidos y no trataría de averiguar cuáles puertos están abiertos.

```
CATEGORIA: 0.572205
==================
        No. paquetes: 238
        Tiempo: 0.0989300000000003
        Puerto destino [22]: 80(108) 51458(12) 51911(10) 51909(10) 51912(10)
        Longitud total: 40(120) 52(48) 1500(38) 1492(12) 904(4)
        Banderas TCP: 16(174) 24(64)
        IP origen: 192.168.1.64(116) 192.168.1.67(100) 208.117.238.160(8) 189.254.100.147(4) 74.125.81.154(4)
        IP destino: 192.168.1.64(100) 192.168.1.71(58) 192.168.1.67(52) 192.168.1.70(10) 192.168.1.69(8)

CATEGORIA: 0.570553
==================
        No. paquetes: 228
        Tiempo: 0.115769
        Puerto destino [19]: 80(142) 51915(14) 51916(8) 51912(8) 51914(6)
        Longitud total: 40(77) 52(71) 1500(56) 744(6) 904(4)
        Banderas TCP: 16(120) 24(108)
        IP origen: 192.168.1.67(88) 192.168.1.64(86) 192.168.1.71(54)
        IP destino: 192.168.1.64(142) 192.168.1.67(53) 192.168.1.71(33)

CATEGORIA: 36.0954
==================
        No. paquetes: 228
        Tiempo: 1.804874
        Puerto destino [8]: 49324(132) 80(68) 443(10) 1451(6) 54420(4)
        Longitud total: 1492(138) 40(62) 52(6) 1085(4) 349(4)
        Banderas TCP: 16(204) 24(20) 18(2) 20(2)
        IP origen: 189.254.100.211(132) 192.168.1.70(56) 192.168.1.64(12) 148.243.168.33(8) 192.168.1.71(6)
        IP destino: 192.168.1.70(132) 189.254.100.211(56) 192.168.1.64(8) 148.243.168.33(6) 65.55.185.222(6)
```

Figura 36. Ejemplo de la información obtenida por categoría.

4.2.5 Cómputo de alto desempeño

Durante el diseño y configuración de la red HTM se identificó que ciertas rutinas de búsquedas y coincidencias durante el entrenamiento y las pruebas se pueden ejecutar en varios procesadores al mismo tiempo, de esta manera, se optimiza el tiempo de procesamiento requerido. Para ello, y de acuerdo a las capacidades del equipo de cómputo utilizado (Linux Debian, Intel core 2 duo, 4GB RAM, 500 GB HDD) se configuró para aprovechar los dos CPUs.

En la figura 37 se muestran los componentes de NuPIC, con el número 1 se encuentra el entorno de tiempo de ejecución en el que se utilizaron dos procesadores para ejecutar los algoritmos de aprendizaje y entrenamiento al mismo tiempo, creando un proceso para cada uno. Por otro lado, el número 2 corresponde a las herramientas de Numenta, las cuales permiten crear la estructura de la red HTM. Por último, se representa con el número 3 los algoritmos de aprendizaje utilizados en cada nodo.

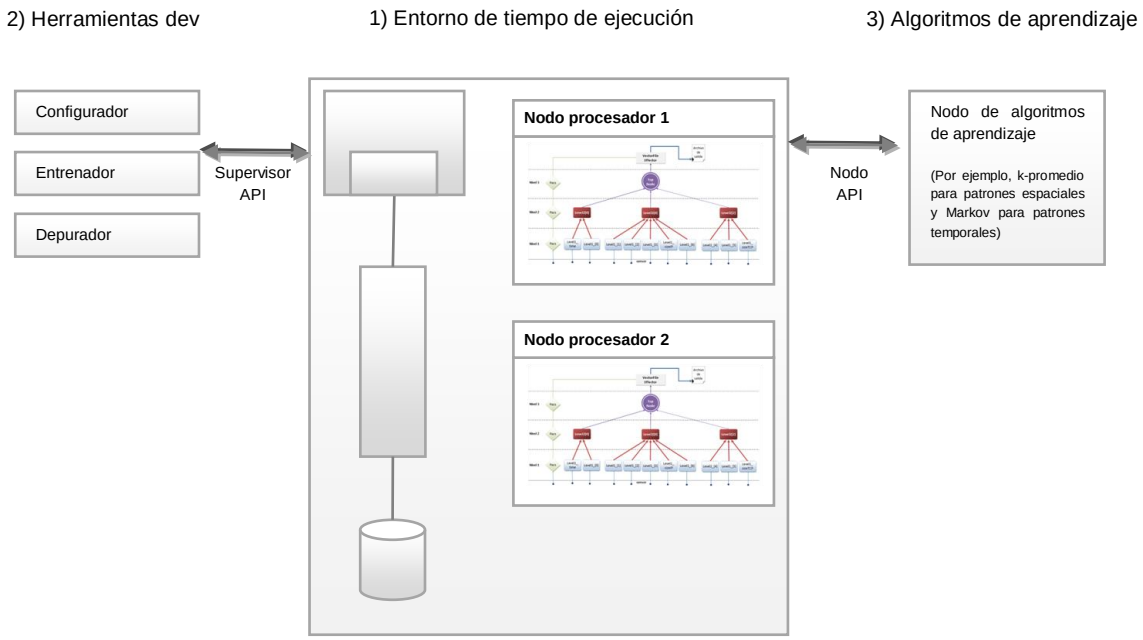


Figura 37. Entorno de programación en paralelo.

De acuerdo a las características del modelo desarrollado en este trabajo de tesis, se implementó el entrenamiento y pruebas utilizando el cómputo de alto desempeño a través de *sessions*, que es la clase que permite a los usuarios interactuar con el entorno de ejecución y que encapsula las entradas, salidas y la carga de los archivos de la red HTM.

4.2.6 Escenarios de prueba

Para verificar que el IDS implementado fuera capaz de identificar incidentes de seguridad se establecieron escenarios de prueba en los cuales se llevaron a cabo diversos ataques. La red que se utilizó para la captura de datos de entrada al sensor tanto de entrenamiento como de prueba se muestra en la siguiente figura 38.

Primero, se conectaron en red siete equipos de cómputo, uno de ellos era un servidor web con sistema operativo Windows, el cual fue el equipo objetivo durante los ataques realizados. Se llevó a cabo un ataque de denegación de servicio utilizando dos máquinas Windows (las que se encuentran en un círculo rojo) mediante la ejecución de la herramienta LOIC, muy conocida por su fácil uso ya que basta con escribir la dirección IP destino y dar un clic para comenzar con el ataque.

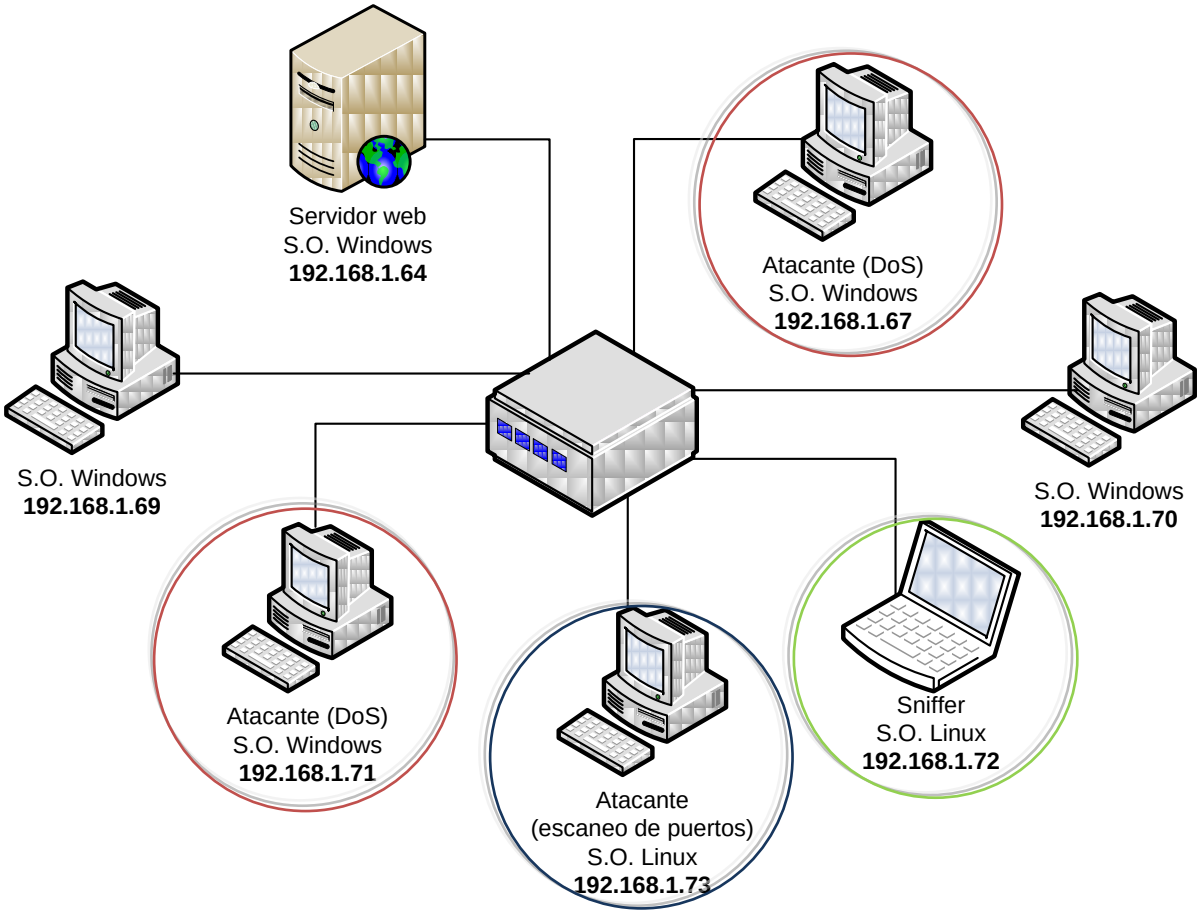


Figura 38. Arquitectura de red utilizada para identificar incidentes.

Por otro lado, se utilizó un equipo Linux desde el cual se ejecutaron las herramientas nmap y hping3 para lanzar un ataque de escaneo de puertos (equipo que se encuentra en un círculo azul), para ello se especificó la dirección IP objetivo y el rango de puertos a escanear.

Finalmente, se colocó una máquina Linux como sniffer, para capturar los archivos de tráfico de red utilizados durante el entrenamiento y las pruebas haciendo uso de la herramienta TCPdump, utilizando las opciones definidas en la sección 4.2.3 y haciendo variaciones en el número de paquetes capturados, para posteriormente analizar cómo afecta la función del tiempo en la red HTM.

Una vez definido el experimento, en el siguiente capítulo se mostrarán los resultados obtenidos durante el aprendizaje y las pruebas, así como su capacidad de identificar los incidentes de seguridad, en este caso de un ataque de denegación de servicio y escaneo de puertos, que por sus características deberían ser identificados como anomalías.

Capítulo 5
Análisis de resultados

En este capítulo se informa sobre los resultados de los experimentos realizados para la identificación de incidentes informáticos, incluyendo información relacionada con los ataques generados y las diferentes metodologías implementadas para la detección.

5.1 Implementación de la red HTM

Para llevar a cabo la implementación del IDS basado en anomalías utilizando la plataforma de Cómputo Inteligente Numenta, fue necesario configurar la red tal y como se describió en la sección 4.2.1, así como la implementación de los procedimientos descritos en la sección 4.2.4.

Una de las principales ventajas que presentó la red HTM fue su capacidad de procesar una gran cantidad de datos; sin embargo, requiere de bastante tiempo para llevarlo a cabo, por lo que surgió la necesidad de encontrar una solución que permitiera optimizar el tiempo de entrenamiento y ejecución de pruebas. Para ello, se implementó el cómputo de alto desempeño en las funciones de entrenamiento y prueba descritos en la sección 4.2.5, haciendo uso de los dos procesadores contenidos en el equipo de cómputo utilizado.

Primero, veamos cómo el proceso de entrenamiento hace uso de los recursos del equipo de cómputo utilizado, tal y cómo se utilizó en los trabajos previos mencionados en la sección 3.3.2. En la figura 39 se observan dos gráficas, la que aparece en la parte superior se refiere al histórico del uso del CPU, la línea de color naranja representa el uso del CPU 1, el cual durante intervalos cíclicos de tiempo funciona al 100%, mientras que la línea roja, que hace referencia al uso del CPU 2, funciona al 30%, en el siguiente intervalo los valores se intercambian, y ahora el CPU 2 trabaja al 100% y el CPU 1 al 30%, y así sucesivamente hasta que termine la ejecución del programa.

La segunda gráfica se refiere al uso de memoria y espacio de memoria de intercambio, mejor conocido como memoria virtual[4], en ésta se observa que se utilizó el 30% de la capacidad de memoria de 4 GB y no se hizo uso del espacio de memoria de intercambio.

[4] El espacio de memoria de intercambio es una zona del disco duro que se usa para guardar imágenes de los procesos que no han de mantenerse en memoria física.

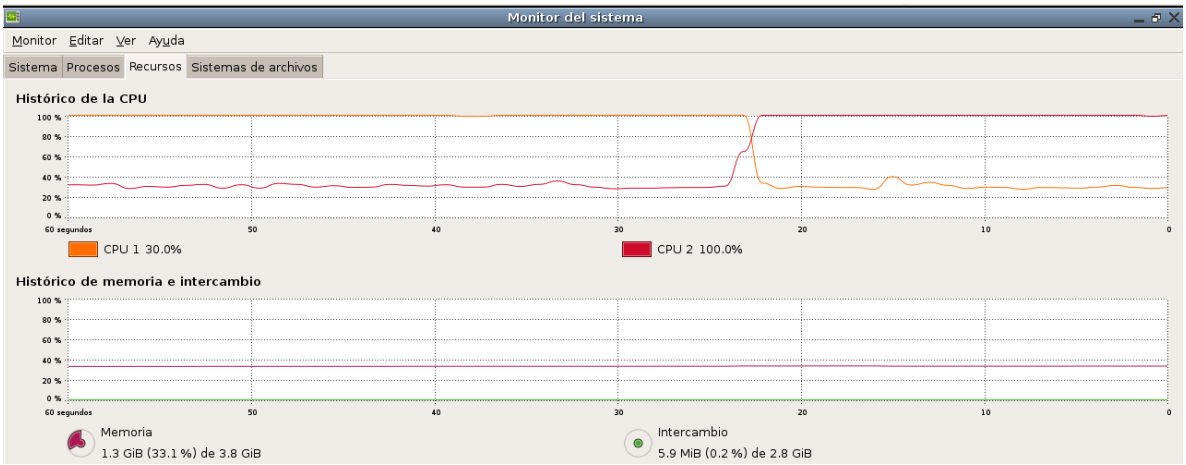


Figura 39. Uso de procesadores durante la ejecución inicial.

Ahora veamos cómo se hace uso de los recursos de cómputo una vez que se modificaron los parámetros en las funciones de entrenamiento y pruebas haciendo uso de los dos procesadores. En la figura 40 se muestra que ambos procesadores se utilizaron al 100% (ver recuadro rojo), y la memoria fue cambiando conforme se iba incrementando el tiempo, llegando a utilizarse hasta un 60% de la memoria con más de un millón de paquetes de entrada, lo cual confirma la teoría de que es capaz de procesar una gran cantidad de información, a diferencia de otros métodos, ya que HTM no sufre de los problemas de escala exponencial debido a la jerarquía.

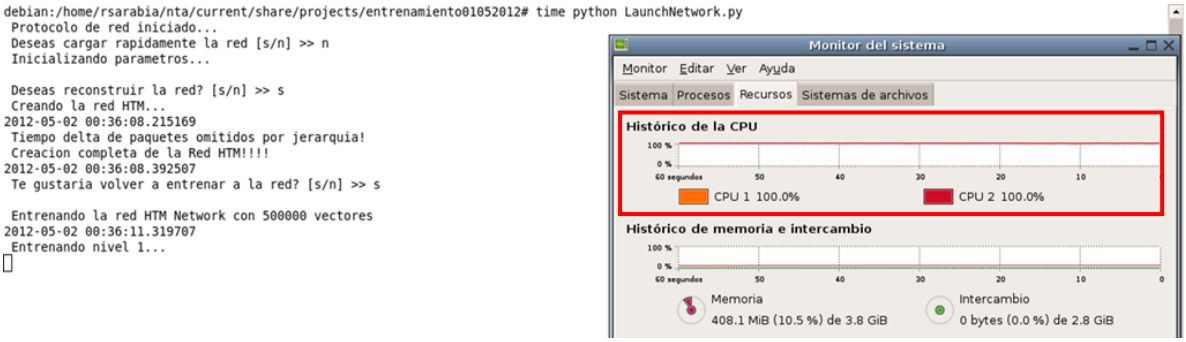


Figura 40. Uso de los procesadores utilizando cómputo de alto desempeño.

Cuando se divide la ejecución de un programa en varios CPU's se crean diferentes procesos como se mencionó en la sección 3.2, llamados numenta_runtime, launcher, orterun y orted, los cuales se observan en el recuadro rojo de la figura 41.

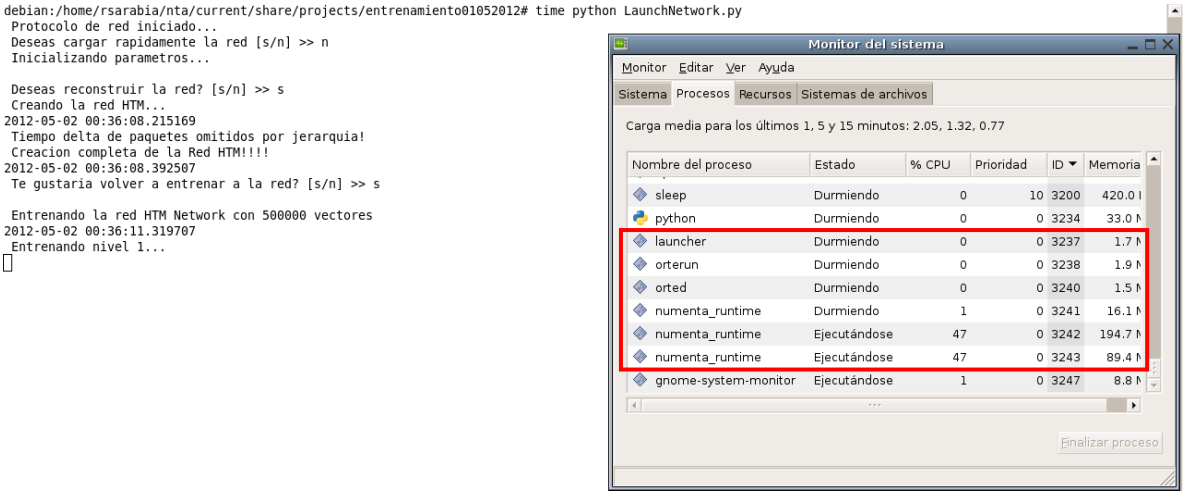


Figura 41. Procesos creados durante la ejecución en cómputo de alto desempeño.

Como se mencionó previamente, fueron aprovechadas las características del equipo utilizado, logrando así reducir los tiempos de ejecución tal como se muestra en la tabla 8, que compara los tiempos de entrenamiento y pruebas utilizando una capacidad parcial o total de los procesadores. En la figura 42 se observa que haciendo uso de 200,000 paquetes de entrenamiento y los dos CPU's al 100% se redujo el tiempo de ejecución en un 44.5%. Asimismo, cuando se realizaron las pruebas con un millón de paquetes utilizando los dos CPU's al 100% se redujo el tiempo en un 41.6%.

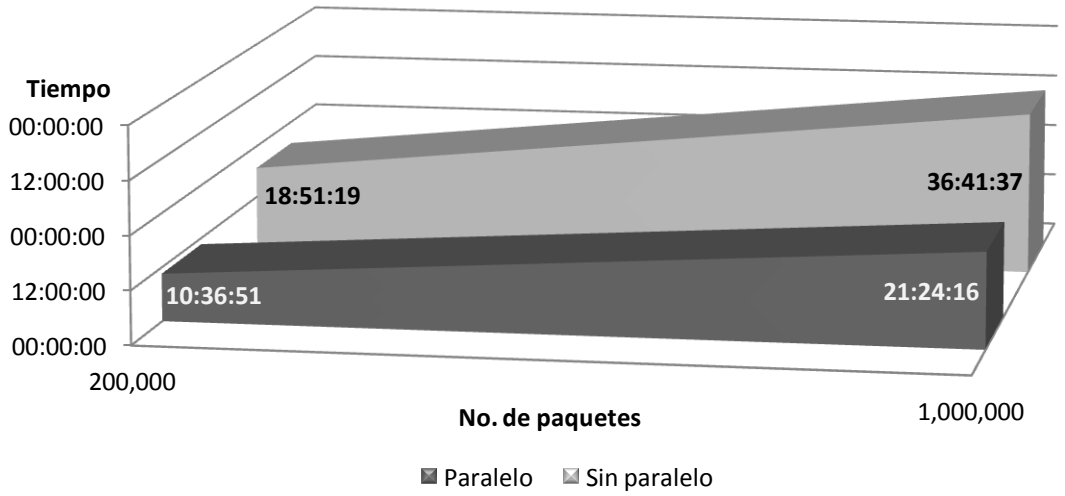


Figura 42. Gráfica comparativa de los tiempos de ejecución

Adicionalmente, se obtuvo un valor teórico, tal que al usar p procesadores el tiempo se redujera a t(n)/p. Considerando esto y tomando los valores obtenidos, se obtuvo que el tiempo de entrenamiento era de 566 minutos con una diferencia de 12.5% respecto al valor real, y el tiempo de ejecución de pruebas era de 1101 minutos con una diferencia de 16.6% respecto al valor real obtenido. Estas diferencias surgieron porque la estimación del tiempo es más compleja, no sólo se debe considerar el número de procesadores, sino también la manera en que están conectados entre sí y los módulos de memoria.

Procesamiento	No. paquetes entrenamiento	Tiempo máquina [min]	No. paquetes pruebas	Tiempo teórico [min]
t(n)	200,000	1131	1,000,000	2201
$t(n)_{real}$ 2 CPUs	200,000	637	1,000,000	1284
t(n)/p	200,000	566	1,000,000	1101

Tabla 8. Tabla comparativa de los tiempos de ejecución.

Adicionalmente, se constató que la relación que existe entre el número de paquetes que se utilizaron para entrenar a la red HTM y el tiempo invertido en el aprendizaje era directamente proporcional, es decir, a mayor número de paquetes es mayor el tiempo de aprendizaje, tal como se muestra en la figura 43.

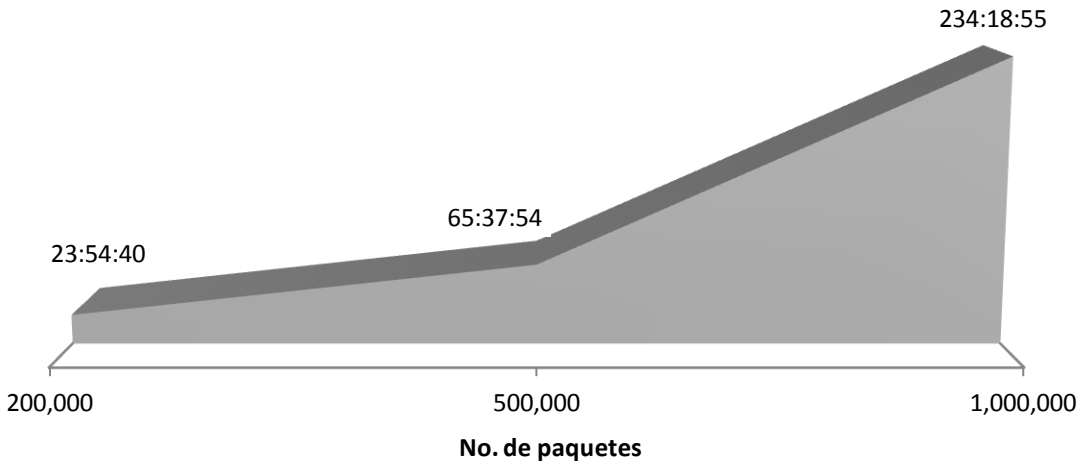


Figura 43. Relación del número de paquetes y tiempo de entrenamiento.

5.2 Escenarios de pruebas

Uno de los principales objetivos de este trabajo de tesis era poder identificar incidentes utilizando un IDS desarrollado en HTM, para lo cual fue necesario llevar a cabo diversos escenarios de pruebas, ejecutando dos tipos de ataques: escaneo de puertos y denegación de servicio. A continuación se dará a conocer los resultados obtenidos de los ataques realizados.

5.2.1 Escaneo de puertos

Un escaneo de puertos debería reportarse como una anomalía, y ser identificado como un incidente, para ello, se realizaron diversos ataques utilizando las herramientas de hping3 [36] y nmap [45], que se utilizan para la exploración de red y auditoría de seguridad, los cuales permiten determinar qué equipos se encuentran disponibles en una red, qué servicios ofrecen, qué sistemas operativos y demás características.

- **Hping3**

Mediante el comando remarcado en rojo que se muestra en la figura 44, se ejecutó un escaneo de puertos, especificando el rango de puertos a analizar, en este ejemplo se escanearon todos los puertos que se encuentran en el servidor 192.168.1.64, un servidor web, y cuyo objetivo era ver si existía una puerta abierta o alguna vulnerabilidad del cual se pudieran aprovechar los intrusos.

Comando: hping3 –scan ’1-65535’ –S 192.168.1.64
IP origen: 192.168.1.73
IP destino: 192.168.1.64

```
debian:/home/rsarabia# hping3 --scan '1-65535' -S 192.168.1.64
Scanning 192.168.1.64 (192.168.1.64), port 1-65535
65535 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
  554 rtsp       : .S..A... 128 10851  8192    46
 3389            : .S..A... 128 17261  8192    46
 5357            : .S..A... 128 26996  8192    46
10243            : .S..A... 128 29698  8192    46
 2869            : .S..A... 128 42255  8192    46
 3306 mysql      : .S..A... 128 31074  8192    46
```

Figura 44. Ataque de escaneo de puertos utilizando hping3

Durante el ataque se pudieron identificar varios puertos abiertos en el equipo objetivo -192.168.1.64-, entre los cuales se encontraban el puerto 80, que indica que se trataba de un servidor web, el 443 que es https (servidor web seguro) y el 3306 referente a una base de datos de mysql, entre otros.

Se realizó el mismo ataque utilizando otra herramienta conocida como nmap, en el que a través del comando resaltado en rojo en la figura 45 se obtuvo información de los puertos abiertos en el servidor 192.168.1.64, que a diferencia de la herramienta anterior se obtuvo información adicional como la versión de los servicios que se encontraban abiertos, por ejemplo, el puerto 80 corre la versión de apache httpd 2.2.17 php/5.3.4.

- **Nmap**

Comando: nmap –sS –A –p 1-65535 192.168.1.64
IP origen: 192.168.1.73
IP destino: 192.168.1.64

```
debian:/home/rsarabia# nmap -sS -A -p 1-65535 192.168.1.64

Starting Nmap 5.00 ( http://nmap.org ) at 2012-02-14 11:48 CST
Interesting ports on JASB (192.168.1.64):
Not shown: 65517 closed ports
PORT      STATE SERVICE       VERSION
80/tcp    open  http          Apache httpd 2.2.17 ((Win32) mod_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.3.
4 mod_perl/2.0.4 Perl/v5.10.1)
|  html-title: Access forbidden!
|_ Requested resource was http://JASB/xampp/
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn
443/tcp   open  ssl/http      Apache httpd 2.2.17 ((Win32) mod_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.3.
4 mod_perl/2.0.4 Perl/v5.10.1)
|_ sslv2: server still supports SSLv2
|_ html-title: Did not follow redirect to https://JASB/xampp/ and no page was returned.
445/tcp   open  netbios-ssn
554/tcp   open  rtsp?
2869/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
3306/tcp  open  mysql         MySQL (unauthorized)
3389/tcp  open  ms-term-serv?
5357/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ html-title: Service Unavailable
10243/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ html-title: Not Found
49152/tcp open  msrpc         Microsoft Windows RPC
49153/tcp open  msrpc         Microsoft Windows RPC
49154/tcp open  msrpc         Microsoft Windows RPC
49155/tcp open  msrpc         Microsoft Windows RPC
```

Figura 45. Ataque de escaneo de puertos utilizando nmap

5.2.2 Denegación de servicio

Un ataque de denegación de servicio también debería ser detectado como una anomalía, para comprobarlo se realizaron ataques utilizando la herramienta LOIC [40], muy conocida y utilizada por el grupo hactivista Anonymous durante sus protestas ya que con un sólo clic cualquier persona puede formar parte del ataque.

En la figura 46 se observa la aplicación LOIC, el cual con sólo especificar la dirección IP o URL del equipo objetivo y seleccionar el botón ubicado en la parte superior derecha, da inicio al ataque de denegación de servicio.

- **LOIC**

IP origen: 192.168.1.67, 192.168.1.67
IP destino: 192.168.1.64
Puerto destino: 80

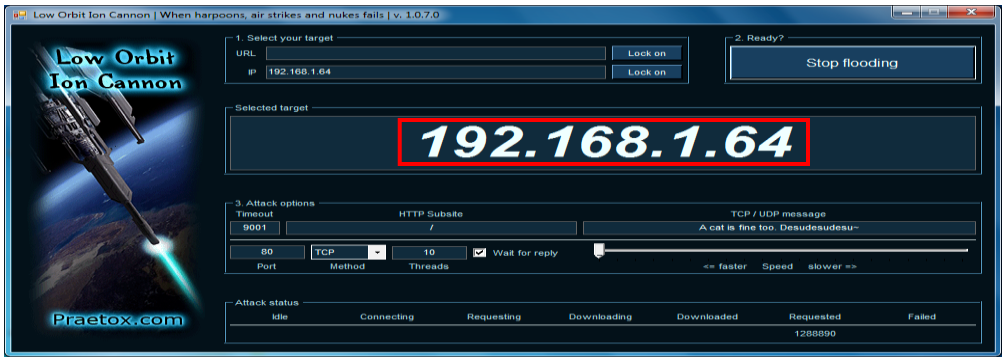


Figura 46. Ataque de denegación de servicio utilizando LOIC.

De los ataques realizados (ver tabla 9) se obtuvo la cantidad de información en MB que fue generada durante los ataques, apreciando un incremento significativo de la cantidad de tráfico de red principalmente en el ataque DoS con 329 MB en tan sólo 46 segundos como se observa en la figura 47.

Ataque	Comando	Tamaño [MB]	Tiempo inicial	Tiempo final
Escaneo de puertos	hping3 –scan ’15-65535’ –S 192.168.1.64	101	00:01:23	00:02:26
	nmap –sS –A –p 1-65535 192.168.1.64	55	00:01:42	00:02:08
DoS	LOIC	329	00:01:35	00:02:21

Tabla 9. Información referente a los ataques realizados.

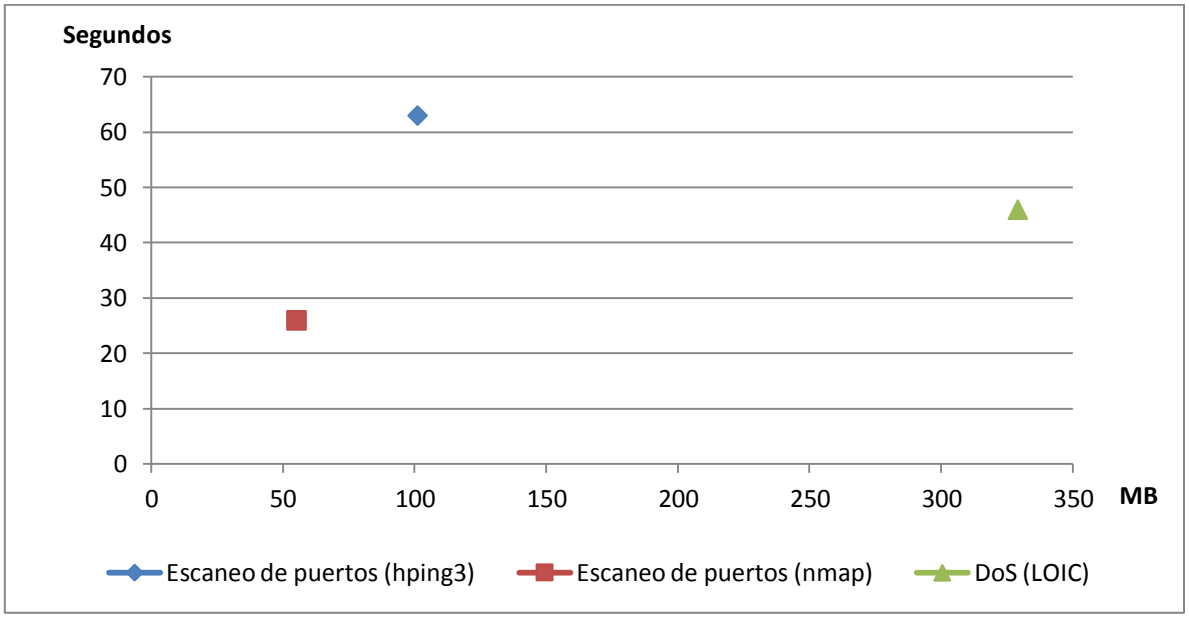


Figura 47. Comparación entre tiempo y cantidad de información en cada ataque.

En la figura 48 se observa que independientemente del tipo de paquetes utilizados en el entrenamiento, el tiempo que requiere la red para realizar las pruebas depende del número de paquetes que se utilizaron para entrenar a la red, a mayor número de paquetes, mayor el tiempo requerido para las pruebas.

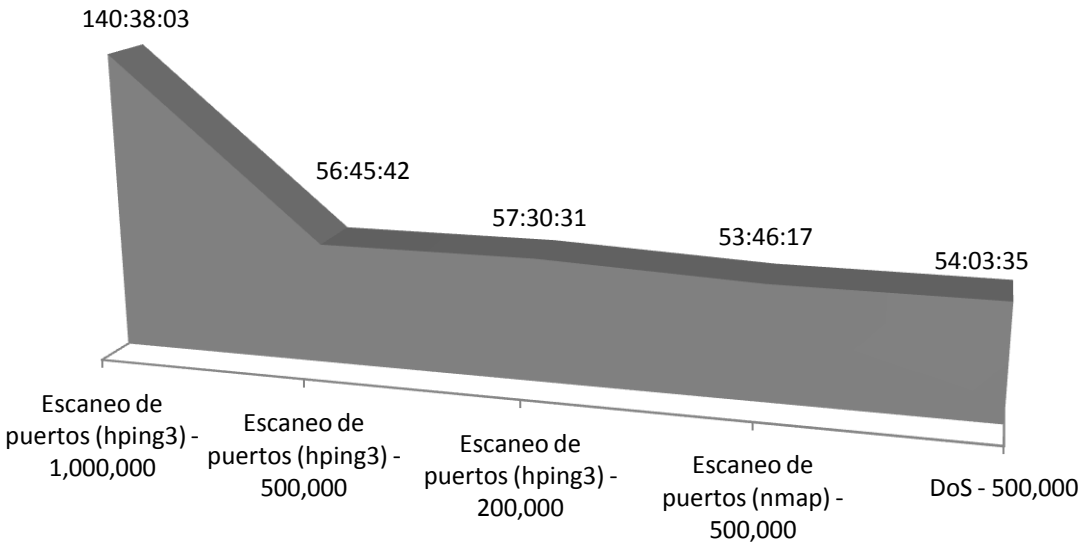


Figura 48. Relación entre el tipo de ataque y tiempo de pruebas

5.3 Detección de anomalías

Una vez que fue entrenada la red HTM y se ejecutaron los ataques descritos en la sección anterior, se determinaron las anomalías generadas utilizando dos herramientas de programación con diferentes comandos.

La primera herramienta utilizada fue bash, un programa informático de UNIX cuya función consiste en interpretar órdenes. Se utilizó el comando diff, utilería para la comparación de archivos que genera las diferencias entre dos archivos, con el cual se comparó el archivo generado de las categorías de pruebas contra el archivo generado de las categorías de entrenamiento, el resultado de esta diferencia se guarda en un archivo que contiene las anomalías, tal como se muestra en la figura 49.

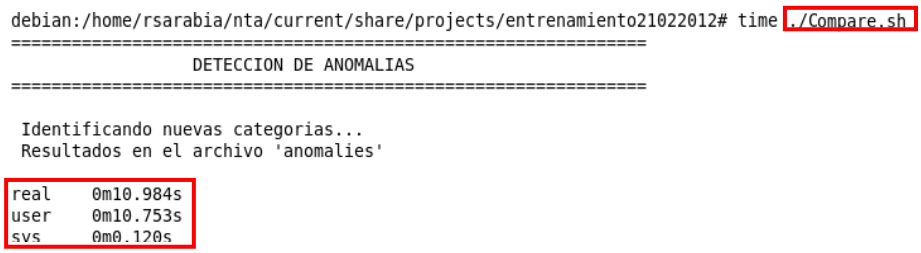


Figura 49. Detección de anomalías utilizando bash.

La segunda herramienta utilizada fue perl, que dadas sus características brinda un poderoso manejo de las cadenas de caracteres y expresiones regulares. Se utilizó la función hash de hashes, que consiste de una o más llaves individuales y su valor asociado, cuyas operaciones se desarrollan en tiempo constante dado que la técnica utiliza búsquedas lineales. Por lo que, se obtuvieron los hashes de entrenamiento y de pruebas y, de estas últimas se eliminaron las que fueron creadas previamente durante el entrenamiento. Los valores que permanecieron en el hash son consideradas como anomalías. La ejecución del script[5] desarrollado se observa en la figura 50.

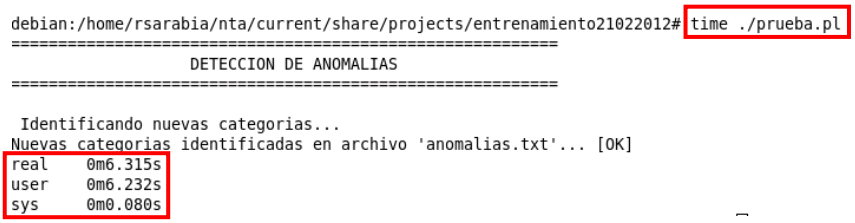


Figura 50. Detección de anomalías utilizando perl.

[5] Script. Programa usualmente simple, que se almacena en un archivo de texto plano.

En las figuras 49 y 50 se resaltan con un recuadro en rojo los tiempos de ejecución de las herramientas utilizadas. El algoritmo desarrollado en perl se ejecutó en menor tiempo, reduciendo en un 42.5 % el tiempo requerido por la herramienta bash.

Después de utilizar el script de perl, se puede concluir que no existe una relación entre el número de vectores de entrada y la capacidad de detección de incidentes, porque el éxito de la detección depende de garantizar que la captura de tráfico incluyera el ataque realizado. En el ejemplo del escaneo de puertos, la mayor capacidad de detección con aproximadamente 12% se obtuvo con un entrenamiento de 500,000 paquetes, tal como se muestra en la figura 51.

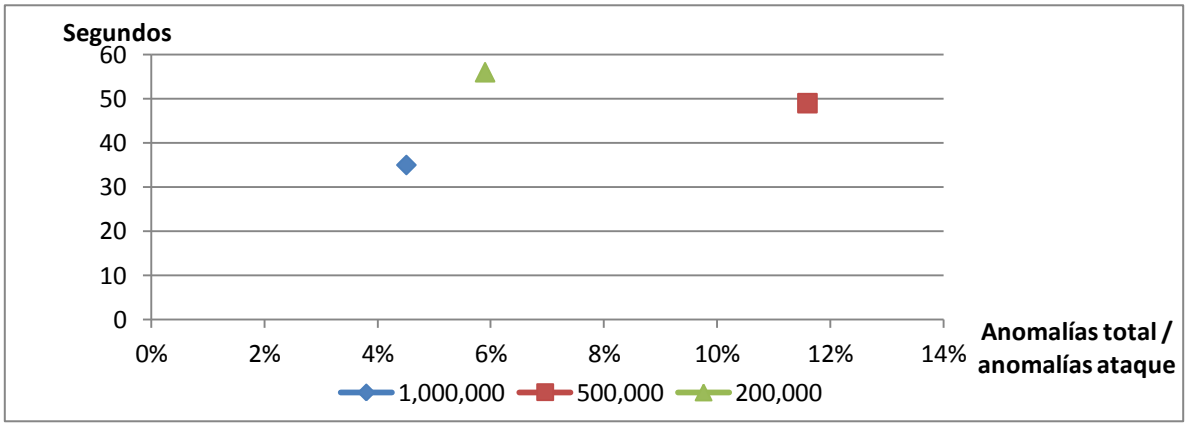


Figura 51. Anomalías detectadas por cantidad de vectores de entrada.

Asimismo, la capacidad de detección no depende del tipo de ataque, en este ejemplo (ver figura 52) se observa que el ataque de escaneo de puertos utilizando hping3 generó un mayor número de anomalías detectadas y no el DoS que generó una mayor cantidad de tráfico de red.

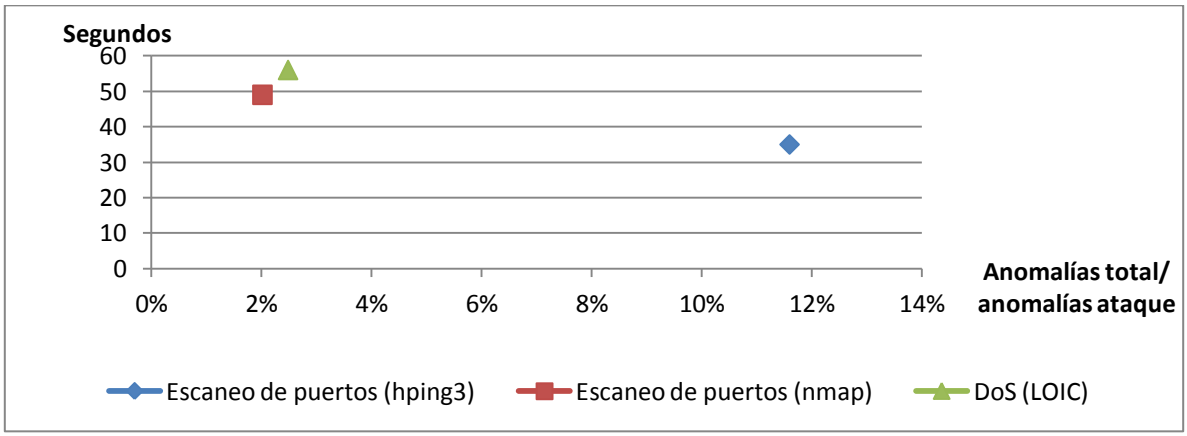


Figura 52. Anomalías detectadas por tipos de ataques.

El resumen de los datos obtenidos de los diversos escenarios de ataques se puede observar en la tabla 10. En la primera columna se encuentra el tipo de ataque (escaneo de puertos o DoS), seguido del número de paquetes utilizados para entrenar la red, la duración de la etapa de entrenamiento, número de paquetes utilizados para las pruebas y el tiempo de duración de las pruebas que como es de esperarse a mayor número de paquetes utilizados mayor el tiempo que toma la red en generar las categorías. Posteriormente, se incluye el total de las anomalías detectadas que se refiere al número de categorías que fueron creadas durante las pruebas y que no aparecen en la relación de categorías del entrenamiento, y se incluyen las anomalías relacionadas con cada tipo de ataque, es decir que, contengan información de los puertos atacados, la dirección IP origen y destino y, finalmente el tiempo que tarda en obtener la información sobre las categorías anómalas.

Ataque	Paquetes entrenados	Duración entrenamiento	Paquetes de prueba	Duración pruebas	Anomalías detectadas	Anomalías relacionadas con ataque	Duración detección
Escaneo de puertos (hping3)	1,000,000	234:18:55	1,000,000	140:38:03	17080	769	00:00:35
Escaneo de puertos (hping3)	500,000	65:37:54	1,000,000	56:45:42	7585	879	00:00:49
Escaneo de puertos (hping3)	200,000	23:54:40	1,000,000	57:30:31	4412	260	00:00:56
Escaneo de puertos (nmap)	500,000	65:37:54	1,000,000	53:46:17	12462	251	00:01:09
DoS	500,000	65:37:54	1,000,000	54:03:35	24158	599	00:01:29

Tabla 10. Análisis de resultados de los ataques realizados.

5.4 Identificación de incidentes

Después de detectar las categorías anómalas se implementó el algoritmo desarrollado en la sección 4.2.4.4 para identificar los incidentes de seguridad, es decir, obtener información referente a los posibles ataques que permitiera al administrador de red tomar las medidas necesarias para su mitigación.

El primer ataque fue un escaneo de puertos utilizando la herramienta hping3. Se implementaron tres escenarios de prueba con diferente número de paquetes de entrenamiento: 1,000,000, 500,000 y 200,000, respectivamente.

En el primer escenario se detectaron un total de 17080 anomalías, de las cuales 769 estuvieron relacionadas con el ataque, por ejemplo, las categorías remarcadas en color rojo 0.274478, 3.2993 y 3.50982 que aparecen en la figura 53, fueron creadas a partir del escaneo de puertos al equipo objetivo 192.168.1.64, lográndose así identificar el equipo que originó el ataque 192.168.1.73 (recuadro azul).

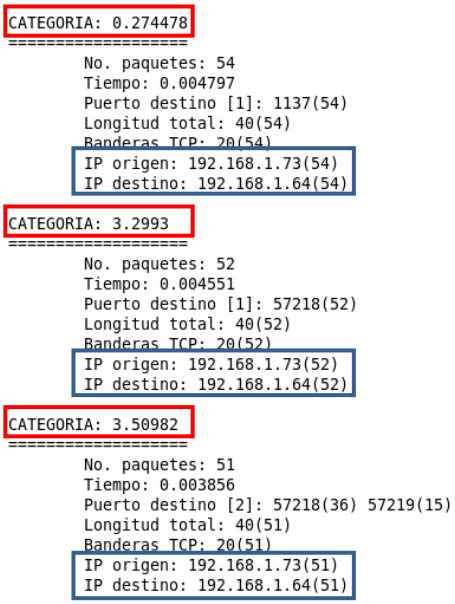


Figura 53. Identificación de un escaneo de puertos con hping3 (escenario 1)

En el segundo escenario se detectaron un total de 7585 anomalías, de las cuales 879 estuvieron relacionadas con el ataque. De los resultados obtenidos se identificó la categoría 14816 que tuvo el mayor número de paquetes asociados con 499,863 (ver figura 54), en el que aparecen 29279 puertos asociados, por lo cual se lanzó el

mensaje de un posible escaneo de puertos ya que se había configurado que a partir de 1000 puertos se debería considerar como un escaneo de puertos. Además, se identificó la dirección IP del atacante 192.168.1.73 y la dirección IP de la máquina atacada 192.168.1.64 (recuadro azul).

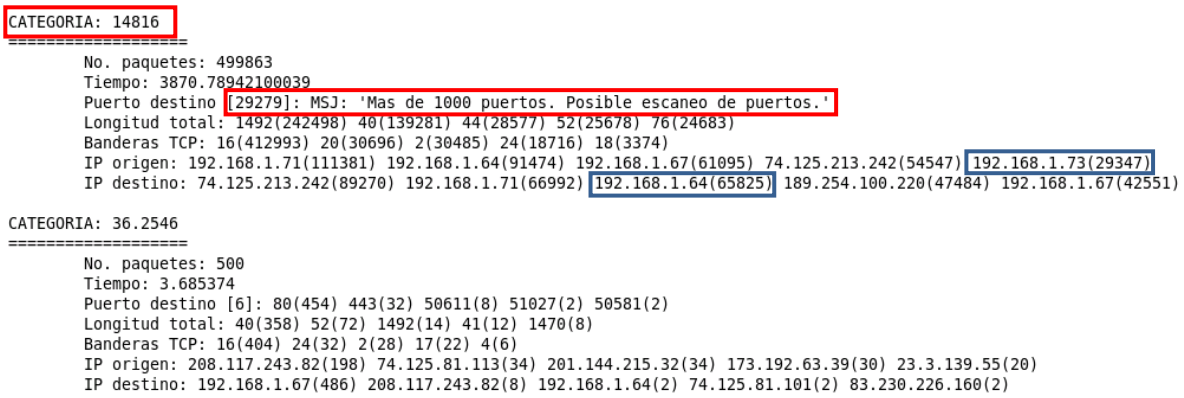


Figura 54. Identificación de un escaneo de puertos con hping3 (escenario 2)

En el tercer escenario se detectaron un total de 4412 anomalías, de las cuales 260 estuvieron relacionadas con el ataque. En la figura 55 se observa la categoría 12226, la cual se encontró hacía referencia a 799,812 paquetes, por lo cual se lanzó el mensaje de un posible escaneo de puertos como en el escenario anterior. Asimismo, se identificó la dirección IP del atacante 192.168.1.73 y la dirección IP de la máquina atacada 192.168.1.64 (recuadro azul).

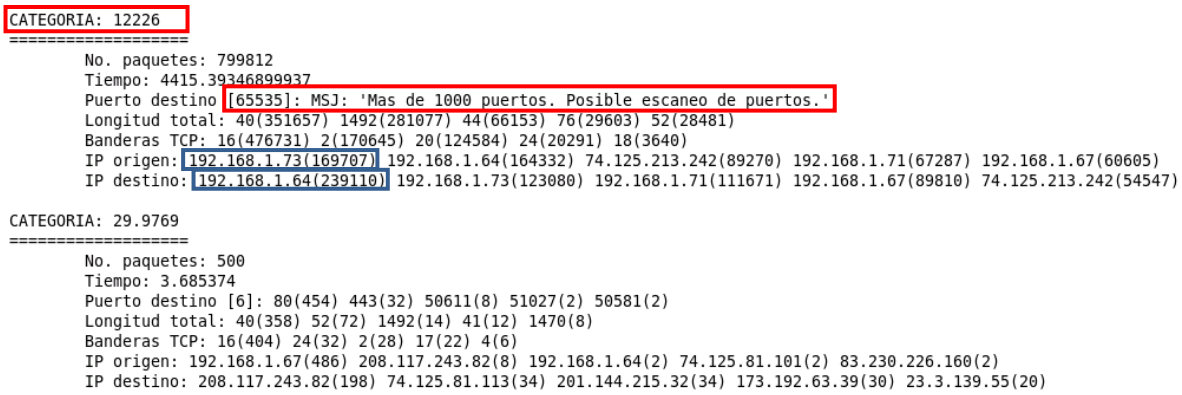


Figura 55. Identificación de un escaneo de puertos con hping3 (escenario 3)

El segundo ataque también fue un escaneo de puertos pero utilizando la herramienta nmap, tal y como se describió en la sección 5.2.1. Se detectaron 12462 anomalías en toda la captura de tráfico de red, de los cuales 251 están relacionadas con el ataque, identificando la categoría 14816 porque estaba asociada a 499,978 paquetes haciendo solicitud a 65529 puertos, que evidentemente no forma parte del comportamiento normal de la red, y por la cantidad excesiva de puertos se puede decir que se trata de un escaneo de puertos tal como se resalta en la figura 56. Además, fue posible identificar los equipos involucrados en el ataque, la dirección IP origen 192.168.1.73 y la dirección IP destino 192.168.1.64.

```
CATEGORIA: 14816
 ==================
        No. paquetes: 499978
        Tiempo: 63.8584490000703
        Puerto destino [62529]: MSJ: 'Mas de 1000 puertos. Posible escaneo de puertos.'
        Longitud total: 40(350118) 44(144202) 1492(4636) 76(547) 52(195)
        Banderas TCP: 2(216308) 18(144209) 4(131936) 16(7302) 24(175)
        IP origen: 192.168.1.73(348227) 192.168.1.64(144300) 189.254.100.211(3667) 192.168.1.70(1985) 74.125.213.213(486)
        IP destino: 192.168.1.64(348405) 192.168.1.73(144201) 192.168.1.70(3668) 189.254.100.211(1983) 192.168.1.71(530)

CATEGORIA: 14095.3
 ==================
        No. paquetes: 325
        Tiempo: 3.861869
        Puerto destino [16]: 51311(157) 54392(149) 51700(3) 51702(3) 56331(2)
        Longitud total: 1492(308) 40(5) 52(5) 236 406
        Banderas TCP: 16(310) 24(11) 18(4)
        IP origen: 74.125.213.213(157) 74.125.213.177(149) 199.59.148.139(6) 66.220.145.41(2) 93.184.216.119(2)
        IP destino: 192.168.1.71(159) 192.168.1.64(149) 192.168.1.72(9) 192.168.1.67(7) 93.184.216.119
```

Figura 56. Identificación de un escaneo de puertos con nmap,

El tercer ataque fue una denegación de servicio utilizando la herramienta LOIC, que por su facilidad de uso es una de las herramientas más populares en la actualidad, de ahí su importancia de poder identificar el incidente. De esta prueba se identificaron 24158 categorías, de las cuales 599 se relacionaron con el ataque de denegación de servicio.

```
CATEGORIA: 14816
 ==================
        No. paquetes: 499979
        Tiempo: 301.38805899998
        Puerto destino [378]: 80(324031) 49339(6695) 52127(4841) 51370(4320) 54392(2447)
        Longitud total: 40(153526) 1500(130767) 52(128085) 1492(20323) 1470(7028)
        Banderas TCP: 16(275956) 24(221326) 4(1267) 18(479) 20(418)
        IP origen: 192.168.1.67(216399) 192.168.1.64(153818) 192.168.1.71(101210) 208.117.238.160(6695) 189.254.100.166(4841)
        IP destino: 192.168.1.64(313474) 192.168.1.67(107664) 192.168.1.71(55562) 192.168.1.70(6706) 208.117.238.160(3852)

CATEGORIA: 14095.3
 ==================
        No. paquetes: 402
        Tiempo: 4.662764
        Puerto destino [13]: 49324(280) 54392(44) 51311(40) 51354(12) 51722(6)
        Longitud total: 1492(370) 1470(8) 52(6) 40(4) 630(2)
        Banderas TCP: 16(382) 24(12) 18(6) 20(2)
        IP origen: 189.254.100.211(280) 74.125.213.177(44) 74.125.213.213(40) 74.125.81.102(22) 199.59.148.139(6)
        IP destino: 192.168.1.70(282) 192.168.1.71(60) 192.168.1.64(46) 192.168.1.67(6) 192.168.1.72(4)
```

Figura 57. Identificación de un DoS utilizando LOIC

En la figura 57 se observa la categoría 14816 que fue identificada como anómala debido a que 499,979 paquetes se encontraban relacionados, 324,031 solicitudes al puerto 80 en un lapso de 0.3 milisegundos y se logró identificar los dos equipos que realizaron el ataque 192.168.1.67 y 192.168.1.71, así como el servidor atacado ubicado en la dirección IP 192.168.1.64.

Finalmente, los resultados obtenidos en los diversos ataques explicados en esta sección muestran que la metodología propuesta para la identificación de incidentes funcionó correctamente logrando así identificar el total de las anomalías analizadas. A continuación se incluirán las conclusiones del trabajo de tesis, aportaciones realizadas y trabajo futuro.

Capítulo 6
Conclusiones

En este capítulo se mencionará la importancia de los conocimientos adquiridos durante la investigación, así como el análisis de cada experimento realizado y los resultados obtenidos para identificar los incidentes de seguridad utilizando el IDS implementado en la plataforma Numenta.

6.1 Resumen del problema

Los sistemas de detección de intrusos (IDS) representan un importante elemento en la protección de las redes de computadoras, son consideradas como una de las primeras defensas contra el comportamiento malicioso y los ataques que puedan poner en peligro la información contenida en los sistemas.

En la actualidad existe un gran número de enfoques innovadores y nuevos modelos de IDS creados tanto en hardware como en software, entonces ¿por qué se utilizó HTM para implementar un IDS basado en anomalías? La respuesta está en que las redes HTM capturan las propiedades algorítmicas y estructurales de neocórtex, es decir, intenta capturar la forma en el que el cerebro humano aprende e infiere de su entorno [7], y por lo tanto, es una oportunidad para resolver este problema con rapidez y precisión, al igual que los seres humanos.

En este trabajo de tesis se implementó un IDS basado en anomalías que permitiera al administrador de red identificar los incidentes ocurridos dentro de una red de datos, para ello se tomaron como referencia trabajos previos tanto nacionales como internacionales sobre el uso de esta nueva plataforma, que a pesar de que eran capaces de detectar las anomalías no así, poder identificar los incidentes.

La red HTM se implementó en un equipo portátil core duo de 500 GB de disco duro, 4GB de RAM. Se configuró la red utilizando un modelo de tres niveles de entrenamiento, el primer nivel aceptaba de entrada vectores de 33 elementos que contenían información de las cabeceras TCP/IP y Ethernet. Como resultado se obtuvo un archivo que incluía el identificador del paquete analizado y la categoría asignada.

Posteriormente, se diseñó un ambiente de pruebas utilizando tanto equipos Windows como Linux, en el que se incluyó un servidor web el cual fue atacado por otros equipos utilizando diversas herramientas para el escaneo de puertos y denegación de servicio, que por sus características pueden ser considerados como anomalías.

6.2 Interpretación de resultados

La evaluación de la red HTM como una opción para la predicción y comprensión del ciberespacio no resulta viable en cuestión de tiempo; sin embargo, es capaz de aprender los patrones espacio-temporales y detectar comportamientos anómalos de forma autónoma y sin requerir grandes capacidades de cómputo a pesar de realizar una amplia cantidad de cálculos.

Con el objetivo de construir una red que modelara un comportamiento preciso de un ciberespacio “normal” se entrenó con 200,000, 500,000 y un millón de paquetes. Sin embargo, entrenar esta cantidad de vectores requería de días completos, por lo que fue necesario modificar algunas funciones durante el entrenamiento y las pruebas para aprovechar las características del equipo de cómputo utilizado mediante el cómputo de alto desempeño, obteniéndose una reducción de aproximadamente del 40% en el tiempo de ejecución durante el entrenamiento y pruebas.

Posteriormente se capturó tráfico malicioso para verificar el funcionamiento de la red HTM, y fue necesario desarrollar una metodología que permitiera identificar los incidentes. Primero, se consideró la opción de modificar la programación en NuPIC para que el resultado no sólo fuera el ID y la categoría por cada vector de entrada, sino que incluyera más información; sin embargo, no fue posible porque fueron utilizadas funciones ya definidas y se hubiera requerido modificar el código fuente, por lo que fue más viable desarrollar un script para recopilar información de la anomalía.

Para ello, se utilizó la correlación de información entre los diversos archivos generados y un método de búsqueda lineal utilizando el lenguaje de programación de perl, considerando el archivo de entrada al sensor, y los resultados de las categorías de entrenamiento y pruebas. Como resultado se obtuvo un archivo con las categorías anómalas e información sobre el no. de paquetes, tiempo, puerto destino, longitud total, banderas, IP origen y destino.

En conclusión, el IDS implementado en la Plataforma Numenta fue capaz de detectar los ataques de escaneo y denegación de servicio como anomalías haciendo uso de diferentes herramientas, así como identificar acertadamente los equipos involucrados en los diversos ataques.

6.3 Alcances y limitaciones

Todo lo que uno hace en Internet, involucra paquetes de información. Por ejemplo, cada página web que uno recibe, viene de una serie de paquetes, cada correo que uno manda, sale como una serie de paquetes. Es demasiada información la que se genera en una red de datos, por lo que se decidió filtrar sólo aquellos paquetes TCP/IP porque es un conjunto de protocolos orientados a conexión, es decir, permite que dos máquinas controlen el estado de transmisión, de ahí que es posible obtener información para poder identificar el incidente.

Debido a que los grandes logros en la tecnología han traído nuevos esquemas de ataques, fue necesario limitar el alcance de este trabajo, porque a pesar de que la herramienta puede ser capaz de identificar cualquier modificación en el comportamiento normal de la red, sólo se implementaron dos tipos de ataques, que por su naturaleza pueden identificarse como anomalías, el escaneo y denegación de servicio.

Existe otro tipo de ataque detectado por los IDS, ataques de penetración, que consisten en la adquisición no autorizada y/o alteración de los privilegios, recursos o datos del sistema, lo que implica un comportamiento no normal de la red. Sin embargo, quedaría pendiente su comprobación con la red HTM desarrollada.

Uno de los principales problemas que presentó la red HTM implementada fue el tiempo requerido para el procesamiento de los datos durante el entrenamiento y la ejecución de pruebas. Se logró reducir el tiempo mediante el uso de cómputo de alto desempeño haciendo uso de los dos procesadores del equipo de cómputo seleccionado; sin embargo, en el mercado existen equipos con mayor capacidad de procesamiento, lo que podría reducir aún más los tiempos de ejecución.

Cabe mencionar que no se recomienda utilizar el IDS desarrollado en un ambiente de producción en tiempo real debido a que los resultados no se podrían dar de manera simultánea, por el tiempo requerido para clasificar todo el tráfico de red.

Adicionalmente, al ser una plataforma todavía en desarrollo no es posible utilizarlo en cualquier sistema operativo, al inicio de este trabajo se planteó el uso de Debian 6.0, la última versión disponible, pero no fue posible instalar NuPIC debido a que requería varias librerías que se encontraban en la versión estable Debian 5.0.

6.4 Trabajo futuro

HTM es un nuevo y poderoso paradigma de cómputo que a la larga puede tener la misma importancia que las computadoras programables tradicionales. ¿Qué hace a HTM diferente de otros enfoques de máquinas de aprendizaje? Las redes HTM son únicas porque combinan lo mejor de diversas técnicas existentes que captura las propiedades algorítmicas y estructurales del neocórtex, es decir, es capaz de programar computadoras inteligentes.

Como parte del desarrollo de esta tesis se solicitó participar en la versión beta de Numenta; sin embargo, me indicaron que durante los últimos meses sus clientes han experimentado con su producto y, como se esperaba, han aprendido mucho de sus necesidades, por lo que han decidido ajustar el producto y no expandir su programa de desarrollo hasta dentro de unos meses. Se encuentran trabajando en algo que llaman “el próximo paso” de los problemas de predicción, con el que pretenden resolver problemas de predicción que al día de hoy no se pueden, por ejemplo, están trabajando en agregar la detección de anomalías: dada una entrada actual en el contexto de entradas pasadas, ¿qué tan inusual es la entrada actual?

HTM aún se encuentra en desarrollo, por lo que tomará tiempo en que sea conocido y se desarrollen más aplicaciones. Asimismo, la configuración de parámetros de una red HTM puede variar los resultados, por lo que se podrían modificar los vectores utilizados, los tipos de nodos, los niveles definidos o incluso los tipos de ataques analizados. Y como se mencionó previamente, se podrían reducir los tiempos de ejecución utilizando mayor capacidad de cómputo, por ejemplo, redes con múltiples procesadores o clústeres de computadoras.

Finalmente, el procedimiento utilizado para la identificación de incidentes podría ser modificado y en lugar de realizar una correlación de información entre diversos archivos, se podrían modificar las funciones incluidas en NuPIC, es decir, realizar cambios en el código fuente para incluir más información respecto a la categoría.

Referencias

[1] Cisco, «Cisco 2011 Annual Security Report. Highlighting Global Security Threats and Trends,» 2012. [En línea]. Disponible: http://www.cisco.com/en/US/prod/collateral/vpndevc/security_annual_report_2011.pdf. [Último acceso: 03 Marzo 2012].

[2] Symantec Corporation, «2010, Symantec Internet Security Threat Report. Trends for,» Abril 2010. [En línea]. Disponible: https://www4.symantec.com/mktginfo/downloads/21182883_GA_REPORT_ISTR_Main-Report_04-11_HI-RES.pdf. [Último acceso: 10 Febrero 2012].

[3] ESET, «ESET Security Report - Latinoamérica 2011,» 2011. [En línea]. Disponible: http://www.eset-la.com/pdf/prensa/informe/eset-report-security-latinoamerica-2011.pdf. [Último acceso: 10 Febrero 2012].

[4] B. Mukherjee, L. Heberlein y K. Levitt, «Network Intrusion Detection,» *IEEE Network,* vol. 8, nº 3, pp. 26-41, Mayo-Junio 1994.

[5] J. Hawkins y S. Blakeslee, On Intelligence, 1ra. ed., Nueva York: Henry Holt, 2004, p. 272.

[6] J. Hawkins, «Why Can´t A Computer Be More Like a Brain? Or What To Do With Transistors?,» *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International,* pp. 38-41, 2008.

[7] J. Hawkins y D. George, «Hierarchical Temporal Memory. Concepts, Theory, and Terminology,» 27 Marzo 2007. [En línea]. Disponible: http://www.numenta.com/htm-overview/education/Numenta_HTM_Concepts.pdf. [Último acceso: 10 Febrero 2012].

[8] Numenta, «Hierarchical Temporal Memory including HTM Cortical Learning Algorithms,» Septiembre 2011. [En línea]. Disponible: http://www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf. [Último acceso: 10 Febrero 2012].

[9] D. George, «How to Make Computers that Work Lke a Brain,» *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE,* pp. 420-423, 28 Agosto 2009.

[10] Y. J. Hall y E. R. Poplin, «Using Numenta's hierarchical temporal memory to recognize CAPTCHAs,» 2007. Disponible: http://www.pembrokeballet.com/10701-HTM_CAPTCHA.pdf. [Último acceso: 10 Febrero 2012].

[11] D. George y B. Jaros, «The HTM Learning Algorithms,» 1 Marzo 2007. [En línea]. Disponible: http://www.numenta.com/htm-overview/education/Numenta_HTM_Learning_Algos.pdf. [Último acceso: 12 Febrero 2012].

[12] D. Admassu Teffera, «A Proposal and Implementation of a Neural Network Based Hierarchical Temporal Memory to Realize Cognitive Functions,» [En línea]. Disponible: http://etd.aau.edu.et/dspace/bitstream/123456789/1539/1/Daniel%20Admassu%20.pdf. [Último acceso: 20 Febrero 2012].

[13] J. Hawkins, «Advances in Modeling Neocortex and Its Impact on Machine Intelligence,» 11 Diciembre 2010. [En línea]. Disponible: http://www.beckman.illinois.edu/gallery/video.aspx?destinationID=o2RWiAWUQEKPgd_8QBTYOA&contentID=fYhfoB6NFE2ytFPl7XLnTA. [Último acceso: 10 Febrero 2012].

[14] D. George, «How the brain might work: A hierarchical and temporal model for learning and recognition,» Junio 2008. [En línea]. Disponible: http://www.numenta.com/htm-overview/education/DileepThesis.pdf.

[15] J. Hawkins, D. George y J. Niemasik, «Sequence memory for prediction, inference and behaviour,» 2009. [En línea]. Disponible: http://rstb.royalsocietypublishing.org/content/364/1521/1203.full.pdf+html. [Último acceso: 10 Febrero 2012].

[16] Numenta, «Hierarchical Temporal Memory. Comparison with Existing Models,» Marzo 2007. [En línea]. Disponible: http://www.numenta.com/htm-overview/education/HTM_Comparison.pdf. [Último acceso: 10 Febrero 2012].

[17] C. Kruegel, D. Mutz, W. Robertson y F. Valeur, «Bayesian Event Classification for Intrusion Detection,» *Computer Security Applications Conference, 2003. Proceedings. 19th Annual,* pp. 14-23, 2003.

[18] C. R. Tosh y G. D. Ruxton, Modelling Perception with Artificial Neural Networks, 1a. ed., Reino Unido: Cambridge University Press, 2010.

[19] S. H. Hedberg, «In the News. Brigging the Gap between Neuroscience and AI,» *Intelligent Systems, IEEE,* vol. 22, nº 3, pp. 4-7, 2007.

[20] Numenta Inc., «Getting Started With NuPIC,» Septiembre 2008. [En línea]. Disponible: http://www.numenta.com/archives/education/nupic_gettingstarted.pdf. [Último acceso: 10 Febrero 2012].

[21] N. C. Schey, «Song identification using the Numenta Platform for Intelligent Computing,» The Ohio State University, 2008.

[22] Numenta, «Advanced NuPIC Programming,» Septiembre 2008. [En línea]. Disponible: http://www.numenta.com/archives/education/nupic_prog_guide.pdf. [Último acceso: 27 Febrero 2012].

[23] J. Hawkings, «Learn Like A Human Why Can't A Computer Be More Like a Brain?,» Abril 2007. [En línea]. Disponible: http://spectrum.ieee.org/computing/hardware/learn-like-a-human/6. [Último acceso: 05 Abril 2012].

[24] G. Khangamwa, «Network Intrusion Detection, based on online traffic measurements using Hierarchical Temporal Memory Networks.,» Septiembre 2009. [En línea]. Disponible: http://www.aitdspace.gr/xmlui/bitstream/handle/123456789/244/Thesis_Gift_Khangamwa_final_MSITT.pdf?sequence=1. [Último acceso: 15 Julio 2012].

[25] G. M. Bonhoff, «Using Hierarchical Temporal Memory for Detecting Anomalous Network Activity,» 2008. [En línea]. Disponible: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA482820. [Último acceso: 15 Julio 2012].

[26] J. R. Sánchez Soledad, *Sistema de Detección de Intrusos Basado en Anomalías de Red Usando la Plataforma Numenta para Cómputo Inteligente,* México, D.F., 2010.

[27] N. De Luna Mendoza, *Sistema para Detección de Intrusos en Línea basado en anomalías de red, usando redes de Memoria Temporal Jerárquica,* México, D.F., 2011.

[28] A. Fourati y K. Al Agha, «An IDS First Line of Defense for Ad Hoc Networks,» de *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, 2007.

[29] National Institute of Standars and Technology, «NIST SP 800-94 Guide to Intrusion Detection and Prevention Systems (IDPS),» [En línea]. Disponible: http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf. [Último acceso: 11 Marzo 2012].

[30] O. D. Oryspayuli, «What intrusion detection approaches work well if only TCP/IP packet header information is available?,» Agosto 2006. [En línea]. Disponible: http://www.utwente.nl/ewi/dacs/assignments/completed/master/reports/MSc-Thesis-Dossay%20Oryspayev.pdf. [Último acceso: 31 Marzo 2012].

[31] D. Dasgupta y F. González, «An immunity-based technique to characterize intrusions in computer networks,» *IEEE Transactions on Evolutionary Computation,* vol. 6, nº 3, pp. 281-291, 2002.

[32] B. Singh, Network Security and Management, 2a. ed., Nueva Delhi: Asoke K. Ghosh, PHI Learning Private Limited, 2009, pp. 113-114.

[33] R. Bace y P. Mell, «NIST Special Publication on Intrusion Detection Systems,» 2003. [En línea]. Disponible: http://www.21cfrpart11.com/files/library/reg_guid_docs/nist_intrusiondetectionsys.pdf. [Último acceso: 11 Marzo 2012].

[34] Wikipedia, «Escáner de puertos,» 01 Febrero 2012. [En línea]. Disponible: http://es.wikipedia.org/wiki/Esc%C3%A1ner_de_puertos. [Último acceso: 16 abril 2012].

[35] Nmap, «Guía de referencia de Nmap (Página de manual),» [En línea]. Disponible: http://nmap.org/man/es/. [Último acceso: 16 abril 2012].

[36] S. Sanfilippo, «hping,» 2006. [En línea]. Disponible: http://www.hping.org/. [Último acceso: 28 julio 2012].

[37] Kaspersky, «DDoS attacks in H2 2011,» 22 Febrero 2012. [En línea]. Disponible: http://www.securelist.com/en/analysis/204792221/DDoS_attacks_in_H2_2011#p22. [Último acceso: 01 Abril 2012].

[38] Carnegie Mellon Software Engineering Institute, «Denial of Service Attacks,» 4 Junio 2001. [En línea]. Disponible: http://www.cert.org/tech_tips/denial_of_service.html#3. [Último acceso: 31 Marzo 2012].

[39] ESET. Blog de Laboratorio, «Consejos para evitar un ataque de denegación de servicio,» 28 marzo 2012. [En línea]. Disponible: http://blogs.eset-la.com/laboratorio/2012/03/28/consejos-ataque-denegacion-servicio/#more-19288. [Último acceso: 16 abril 2012].

[40] Praetox, «LOIC,» 29 01 2012. [En línea]. Disponible: http://sourceforge.net/projects/loic/. [Último acceso: 28 07 2012].

[41] S. Zaman y F. Karray, «Lightweight IDS based on Features Selection and IDS Classification Scheme,» de *International Conference on Computational Science and Engineering*, 2009.

[42] P. Innella y O. McMillan, «An Introduction to IDS,» 6 Diciembre 2001. [En línea]. Disponible: http://www.symantec.com/connect/articles/introduction-ids. [Último acceso: 31 Marzo 2012].

[43] P. Sangkatsanee, N. Wattanapongsakorn y C. Charnsripinyo, «Real-time Intrusion Detection and Classification,» [En línea]. Disponible: http://inms.in.th/inmsweb/paper/162Realtime_Intrusion_Detection_and_Classification.pdf. [Último acceso: 06 Marzo 2012].

[44] M. V. Mahoney y P. K. Chan, «Detecting Novel Attacks by Identifying Anomalous Network Packet Headers,» 2001. [En línea]. Disponible: http://www2.cs.fit.edu/~mmahoney/paper2.pdf. [Último acceso: 25 marzo 2012].

[45] Fyodor, «Nmap Network Scanning,» 2005. [En línea]. Disponible: http://nmap.org/man/es/. [Último acceso: 28 07 2012].

[46] AMIPCI, «Hábitos de los Usuarios de Internet en México,» 17 Mayo 2011. [En línea]. Disponible: http://www.amipci.org.mx/temp/Habitos2011AMIPCIPrensa comprimida-00109590013056463170B.pdf. [Último acceso: 10 Febrero 2012].

[47] B. Elmer Mode, Elementos de probabilidad y estadística, España: Reverté, S.A., 2008.

Glosario

Algoritmos de aprendizaje corticales HTM	Conjunto de funciones de agrupación espacial, temporal, aprendizaje y olvido que comprenden una región HTM, también conocida como algoritmos de aprendizaje HTM.
Aprendizaje	Un nodo se encuentra en el estado de aprendizaje cuando está recibiendo las entradas, midiendo las estadísticas de los insumos, y realiza modificaciones en sus estructuras internas para representar las estadísticas de las entradas.
Categoría	El nivel superior, clase distinta a la que entidades o conceptos pertenecen.
Causa	Un objeto en el mundo. Desde la perspectiva de HTM, lo que es importante acerca de los objetos del mundo es que tienen persistencia, es decir, que existen en el tiempo. Una causa no es necesariamente un objeto físico.
Clasificación	La clasificación se desarrolla durante el entrenamiento: el sistema HTM se presenta con un archivo de categorías. Después de eso, el sistema HTM es presentado con nuevos datos y puede decidir en la categoría más cercana para cada patrón.
Células	Equivalente en HTM a una Neurona.
Coincidencia	Una coincidencia es una notable asignación de dos o más eventos o circunstancias sin relación de causa evidente. En el contexto de red HTM, una combinación específica de patrones que pueden ocurrir juntos en un punto en el tiempo.
CPU	Un CPU es un núcleo de un procesador único. Un procesos de doble núcleo tiene dos CPUs, y un clúster de cinco sistemas, cada uno con dos CPUs de doble núcleo, tiene 20 CPUs.

Creencia	En el contexto de un HTM, una creencia es la distribución de probabilidad de una causa o conjunto de causas. En concreto, la creencia se refiere a la distribución de más de un conjunto de posibles causas, una vez que todas las pruebas de arriba hace abajo, de abajo hacia arriba y lateral han sido considerados.
Detección de coincidencias	El proceso de detección de coincidencias frecuentes entre los patrones de entrada.
Detección de intrusos	El proceso de seguimiento de los eventos que ocurren en un sistema informático o red y su análisis para detectar signos de intrusiones, definidos como intentos de comprometer la confidencialidad, integridad o disponibilidad de un equipo de cómputo o red.
Efector	Efectores son nodos que reciben la salida del nodo superior como entrada. El efector puede enviar la salida a un archivo o dispositivo.
Enlace	Conexión entre nodos en una red HTM.
Entrada	Cualquier nodo puede recibir entradas de todos los nodos a los que está vinculado. Un nodo puede tener múltiples entradas.
Entrenamiento	Para entrenar una red HTM, se invoca el NRE con la configuración de red y los datos de entrenamiento. Durante el entrenamiento, los nodos de la red HTM realizan el aprendizaje y la inferencia.
Grupo	Un conjunto de patrones de coincidencias que es probable que ocurra muy junto en el tiempo.
HTM	Memoria Temporal Jerárquica. Teoría que describe las propiedades estructurales y computacionales del neocórtex.

Incidente	Una amenaza inminente de violación o violación de las políticas de seguridad informática, las políticas de uso aceptable o prácticas estandarizadas de seguridad.
Inferencia	La inferencia es el acto o proceso de derivar una conclusión basada exclusivamente en lo que uno ya sabe. En el contexto de plataforma Numenta, puede significar que durante el entrenamiento, los nodos pueden inferir por ejemplo, la probabilidad de que un determinado elemento es el siguiente de una secuencia basada en otras secuencias que se han visto. Después de que la red HTM se ha entrenado, se pueden alimentar nuevos datos y HTM puede inferir la categoría correspondiente (como un patrón estadístico).
Jerarquía	Una red de elementos conectados donde las conexiones entre los elementos son únicamente identificados como Feed-Forward o Feedback
Memoria Temporal Jerárquica (HTM)	Una tecnología que replica algunas de las funciones estructurales y algorítmicas del neocórtex.
Neurona	Una célula de procesamiento de información en el cerebro.
Nivel	Una región HTM en el contexto de la jerarquía.
Nodo	Un nodo es la unidad básica computacional de una red HTM. Los tipos de nodos incluyen el nodo sensor, efector y de aprendizaje. Un nodo de aprendizaje aprende y representa las estadísticas espaciales y temporales de las entradas a los que está expuesta.
NP (Nodo procesador)	Componente de software que se encarga de la gestión y programación de una parte de una red HTM.
NRE	Numenta Runtime Engine. Software ejecutable requerido para el funcionamiento de las redes HTM.
NuPIC	Acrónimo de Plataforma Numenta para Cómputo Inteligente.

Paralelo	El término “ejecución en paralelo” significa que “se ejecutan simultáneamente en más de un CPU” con el objetivo de reducir el tiempo de ejecución global.
Predicción	Las células de activación (en un estado predictivo) que probablemente estará activo en el próximo futuro debido a la entrada FeedForward Una región HTM frecuentemente predice muchas posibles entradas en el futuro al mismo tiempo.
Proceso	Un proceso es un proceso del sistema operativo ya sea MacOS, Linux o Microsoft Windows. NRE no utiliza hilos, de modo que la aceleración de la ejecución en paralelo siempre se logra mediante el uso de varios procesos.
Red HTM	Un conjunto de nodos, sensores y efectores conectados para realizar una función específica. Sirve como la estructura de HTM que se calcula por el NRE.
Región HTM	La unidad principal de memoria y predicción en una red HTM. Una región se compone de una capa de células altamente interconectadas dispuestas en columnas. Una región HTM tiene una sola capa de células mientras que en el neocórtex (y en última instancia), una región tiene múltiples capas de células. Cuando se refiere al contexto de su posición jerárquica, una región puede ser denominada como un nivel.
Representación distribuida dispersa	Una representación compuesta de muchos bits en el que un pequeño porcentaje está activo y donde un solo bit no es suficiente para transmitir el significado.
Salida	La salida del nodo es la parte del estado del nodo que es accesible por otros nodos. Las salidas pueden ser datos diversos tipos. Cada nodo puede tener varias salidas. Cuando un nodo está en modo de inferencia, esto vuelve disponibles las salidas a otros nodos.

Sensor	Una fuente de entrada a las redes HTM. Los sensores son la interface a archivos externos, dispositivos de hardware, etc. y dan formato a los datos para la entrada a otros nodos.
Supervisor	Parte de NRE responsible de coordinar una red HTM, y para la comunicación con aplicaciones externas (por ejemplo, las herramientas)

Anexos

A. Generación de datos de entrada al sensor

```
#!/usr/bin/perl -w

print "=========================================\n";
print "BIENVENIDOS A LA RED HTM\n";
print "=========================================\n";
print "Directorio raiz de los archivos de trafico de red: ";
$DIR = `pwd`;
$DIR =~ s/\n//g;
$DIRFILE = $DIR."/archivos";
print $DIRFILE;


##### Obtener el directorio donde se encuentran los archivos #####
#################################################################

$date = `date +%d%m%y`;
chop $date;
print "\nNombre del directorio a utilizar [trafico_$date]: ";
$directory = <STDIN>;
chop $directory;

if ($directory eq "") { $directory = "trafico_".$date; }

do
{
      if (-e "$DIRFILE/$directory") {
            chdir "$DIRFILE/$directory";
            print `ls`;
            $a = "fin"; break; }
      else {
            print "El directorio no existe, desea crearlo [S/N]? ";
            $cfg = <STDIN>;
            chop ($cfg);

            if (($cfg eq "N" ) || ($cfg eq "n")) {
                  print "Favor de introducir el nombre del directorio
nuevamente: ";
                  $directory = <STDIN>;
                  chop $directory; }
            else {
                  `mkdir $DIRFILE/$directory`;
                  print "El directorio $directory fue creado\n"; }
      }
} while ( $a ne "fin" );
```

```
##### Leyendo el archivo de tráfico de red #####
################################################

print "Existe el archivo de trafico de red [S/N]? ";
$cfg = <STDIN>;
chop ($cfg);

if (($cfg eq "S" ) || ($cfg eq "s")) {
      print "Nombre del archivo de trafico de red [.tcpdump]: ";
      $filename = <STDIN>;
      chop $filename;
      do
      {
            if (-e "$DIRFILE/$directory/$filename") { $b = "fin"; break;
}
            else {
                  print "Favor de introducir el nombre del archivo
nuevamente: ";
                  $filename = <STDIN>;
                  chop $filename; }
      } while ( $b ne "fin" );

      $filesize = -s "$DIRFILE/$directory/$filename";
      print "Tamano del archivo $filename: $filesize [bytes]\n";
      print "Prefijo de los archivos a generar [salida]: ";
      $prefijo = <STDIN>;
      chop $prefijo;

      if ($prefijo eq "") { $prefijo = "salida"; }
      print "Tamano de los archivos a generar en MB [1]: ";
      $size = <STDIN>;
      chop $size;

      if ($size eq "") { $size = 1; }
      $numfiles = int($filesize / ($size * 1000000)) + 1;
      `sudo tcpdump -r $DIRFILE/$directory/$filename -w $prefijo -C
$size`;
      print "Se generaron $numfiles de $size [MB]... [OK]\n\n";

      ##### Filtrando archivo TCP/IP para entrada a sensor #####
      ##########################################################
      print "Generando archivos de entrada para el sensor... \n";
      for($i = 0; $i < $numfiles; $i++){
                  if ( $i == "0" ) { $pref = $prefijo; }
            else { $pref = $prefijo.$i; }
            $output = $pref.".tmp";

            `sudo tcpdump -r $pref -nn -ttt -xx 'tcp and ip' | egrep
"IP|0x0000|0x0010|0x0020|0x0030" > $output`;
            sensorInput ($output);
            `rm $output`;
      }
}
```

```
##### Generando archivos de trafico de red #####
################################################
else {
      print "Numero de archivos de trafico de red a crear [1]: ";
      $nofiles = <STDIN>;
      chop $nofiles;
      if ($nofiles eq "") { $nofiles = "1"; }

      print "Numero de tramas TCP/IP tendra [2500]: ";
      $tramas = <STDIN>;
      chop $tramas;
      if ($tramas eq "") { $tramas = "2500"; }

      print "Numero de captura de inicio [1]: ";
      $capinicio = <STDIN>;
      chop $capinicio;
      if ($capinicio eq "") { $capinicio = "1"; }

      $capfinal = $capinicio + $nofiles;

      print "Prefijo de los archivos a generar [salida]: ";
      $prefijo = <STDIN>;
      chop $prefijo;
      if ($prefijo eq "") { $prefijo = "salida"; }


      ##### Filtrando archivo TCP/IP para entrada a sensor #####
      ##########################################################

      print "Generando $nofiles con $tramas tramas TCP/IP ... \n";
        for($i = $capinicio; $i < $capfinal; $i++){
                $pref = $prefijo.$i;
                $output = $pref.".tmp";

                `sudo tcpdump -i wlan0 -v -c $tramas -nn -ttt -xx 'tcp
and ip' | egrep "IP|0x0000|0x0010|0x0020|0x0030" > $output`;
                sensorInput ($output);
           #`rm $output`;
        }
}

print "Proceso completado ..................... [OK]\n";
#$redHTM = $DIR."/LaunchNetwork.py";
#`python $redHTM`;
```

```
########## Función para generar entrada para sensor ##########
################################################################

sub sensorInput {
      $filetmp = shift;
      open (FH,"<$filetmp");
      @linea = <FH>;
      $file = substr($filetmp,0,length($filetmp)-4);
      my $archivo_salida = $file.".snr";
      open (FH1,">$archivo_salida");

      $inicia=0;
      $delta="";
      $ID=1;
      my $cadena="";
      my $pa="";
      print FH1 "33\n";

      foreach $paquete(@linea)
      {
            if ($paquete=~/IP/) {
                  $delta = "0.".substr($paquete,0,6);
                  $inicia=1;
                  $cadena=$ID." ".$delta;
                  $ID++;
            }
            else {

                  if (($inicia >= 1) && ($inicia <= 3)) {
                            chop($paquete);
                              @pack=split(/:/,$paquete);
                              $pack[1] =~s/\s*//g;
                              $pa=$pa.$pack[1];
                       }

                  if ($inicia == 4) {
                              chop($paquete);
                              @pack=split(/:/,$paquete);
                              $pack[1] =~s/\s*//g;
                              $pa=$pa.$pack[1];

      # 2-7
                  $macorigen = hex(substr($pa,12,2))."
                  ".hex(substr($pa,14,2))." ".hex(substr($pa,16,2))."
                  ".hex(substr($pa,18,2));
                  $macorigen .= " ".hex(substr($pa,20,2))."
                  ".hex(substr($pa,22,2));
      # 8-13
                   $macdestino = hex(substr($pa,0,2))."
                  ".hex(substr($pa,2,2))." ".hex(substr($pa,4,2))."
                  ".hex(substr($pa,6,2));
                  $macdestino .= " ".hex(substr($pa,8,2))."
                  ".hex(substr($pa,10,2));
```

```
      # 14
                  $tipo = substr($pa,24,2).substr($pa,26,2);
      # 15
                  $iporigen = hex(substr($pa,52,2))."
                  ".hex(substr($pa,54,2))." ".hex(substr($pa,56,2))."
                  ".hex(substr($pa,58,2));
      # 16-19
                   $ipdestino = hex(substr($pa,60,2))."
                  ".hex(substr($pa,62,2))." ".hex(substr($pa,64,2))."
                  ".hex(substr($pa,66,2));
      # 20-23
                  $longitud = substr($pa,32,2).substr($pa,34,2);
      # 24
                  $identificacion = substr($pa,36,2).substr($pa,38,2);
      # 25
                  $protocolo = substr($pa,46,2);
      # 26
                  $banderasip = substr($pa,40,2);
      # 27
                  $puertoorigen = substr($pa,68,2).substr($pa,70,2);
      # 28
                  $puertodestino = substr($pa,72,2).substr($pa,74,2);
      # 29
                  $secuencia =
                  substr($pa,76,2).substr($pa,78,2).substr($pa,80,2).subs
                  tr($pa,82,2);
      # 30
                  $ack =
                  substr($pa,84,2).substr($pa,86,2).substr($pa,88,2).subs
                  tr($pa,90,2);
      # 31
                  $banderastcp = substr($pa,94,2);
      # 32
                  $ventana = substr($pa,96,2).substr($pa,98,2);;
      # 33
                  print FH1 $cadena." ".$macorigen." ".$macdestino."
".hex($tipo)." ".$iporigen." ".$ipdestino." ";
                  print FH1 hex($longitud)." ".hex($identificacion)."
".hex($protocolo)." ".hex($banderasip)." ";
                  print FH1 hex($puertoorigen)." ".hex($puertodestino)."
".hex($secuencia)." ".hex($ack)." ";
                  print FH1 hex($banderastcp)." ".hex($ventana)."\n";

                  $pa = "";
                  } #if
                $inicia++;
            } #else
      } #foreach
      print "Archivo $archivo_salida creado .....................
[OK] \n";
} #function
```

B. Identificación de incidentes

```
#!/usr/bin/perl –w

print "================================================================\n";
print "                    IDENTIFICACION DE ANOMALÍAS                  \n";
print "================================================================\n";

print "\n Identificando nuevas categorias...\n";

open (OUTPUT, ">anomalias.txt");
open (INC, ">incidentes.txt");
my %training = ();
my %test = ();
my %data = ();
my @s = ();

open (TR, "training_results.txt") or die "El archivo no se puede leer:
$!";
    while( <TR> ) {
      my @a = split;
        $training{$a[0]} ++; }
close TR;

open (TE, "test_results.txt") or die "El archivo no se puede leer: $!";
     while( <TE>) {
      my @b = split;
            $test{$b[0]} ++;
      push ( @{ $testing{$b[0]} }, $b[1]); }
close TE;

for my $w ( keys %training ) { delete($test{$w}); }
for my $t ( sort { $test{$b} <=> $test{$a} || $a <=> $b } keys %test) {
print OUTPUT "$t\n"; } #Anomalias detectadas
print " Nuevas categorias identificadas en archivo 'anomalias.txt'...
[OK]\n";


print "================================================================\n";
print "                  IDENTIFICACION DE INCIDENTES                  \n";
print "================================================================\n";

print " Leyendo datos de origen ... ";
open (TD, "data/vectorForm/test_data.snr") or die " El archivo no puede
leer: $!";
     while( <TD>) {
      my @c = split;
        $data{$c[0]} = $_;
      #print "$_\n";
      }
close TD;
```

```
print "[OK]\n Obteniendo informacion... ";
for my $t ( sort { $test{$b} <=> $test{$a}  || $a <=> $b } keys %test) {
      foreach (@{ $testing{$t} }) {
         @s = split(/\s+/, $data{$_});
            $delta{$t} += $s[1];
            $pto_destino{$t}{$s[28]}++;
            $IP_size{$t}{$s[23]}++;
            $flags_TCP{$t}{$s[31]}++;
            $IP_origen{$t}{$s[15].".".$s[16].".".$s[17].".".$s[18]}++;
            $IP_destino{$t}{$s[19].".".$s[20].".".$s[21].".".$s[22]}++;
      }


last if $test{$t} eq '4';
print INC "CATEGORIA: $t\n";
print INC "==================";
print INC "\n\tNo. paquetes: $test{$t}\n";
print INC "\tTiempo: $delta{$t}\n";


my $nopto = scalar keys %{ $pto_destino{$t}};
print INC "\tPuerto destino [$nopto]: ";
my $pto = 0;
if ( $nopto > '1000' ) {
      print INC "MSJ: 'Mas de 1000 puertos. Posible escaneo de
      puertos.'";}
else {

for $p (sort { $pto_destino{$t}{$b} <=> $pto_destino{$t}{$a} } keys %{
$pto_destino{$t}}) {
      $pto++;
      if ($pto_destino{$t}{$p} eq "1") {
            printf INC "$p "; }
      else { printf INC ("%s(%d) ", $p, $pto_destino{$t}{$p}); }
      last if $pto eq '5';
      }
}


print INC "\n\tLongitud total: ";
my $sz = 0;
for my $q ( sort { $IP_size{$t}{$b} <=> $IP_size{$t}{$a} } keys %{
$IP_size{$t}}) {
      $sz++;
      if ($IP_size{$t}{$q} eq "1") {
            printf INC ("%d ", $q); }
      else { printf INC ("%d(%d) ", $q, $IP_size{$t}{$q}); }
      last if $sz eq '5';
      }
```

```
print INC "\n\tBanderas TCP: ";
my $flg = 0;
for my $r ( sort { $flags_TCP{$t}{$b} <=> $flags_TCP{$t}{$a} } keys %{
$flags_TCP{$t}}) {
      $flg++;
      if ($flags_TCP{$t}{$r} eq "1") { printf INC ("%d ", $r); }
      else { printf INC ("%d(%d) ", $r, $flags_TCP{$t}{$r}); }
      last if $flg eq '5';
      }


print INC "\n\tIP origen: ";
my $ipo = 0;
for my $u ( sort { $IP_origen{$t}{$b} <=> $IP_origen{$t}{$a} } keys %{
$IP_origen{$t}}) {
      $ipo++;
      if ($IP_origen{$t}{$u} eq "1") { print INC "$u"; }
      else { printf INC ("%s(%d) ", $u, $IP_origen{$t}{$u}); }
      last if $ipo eq '5';
      }


print INC "\n\tIP destino: ";
my $ipd = 0;
for my $v ( sort { $IP_destino{$t}{$b} <=> $IP_destino{$t}{$a} } keys %{
$IP_destino{$t}}) {
      $ipd++;
      if ($IP_destino{$t}{$v} eq "1") { print INC "$v "; }
      else { printf INC ("%s(%d) ", $v, $IP_destino{$t}{$v}); }
      last if $ipd eq '5';
      }

print INC "\n\n";
}

print "[OK]\n Informacion detallada de las anomalias en archivo
'incidentes.txt'... [OK]\n";
```

C. Propagación de creencia en redes HTM

Esta sección describe cómo se implementa la propagación de creencias en las redes HTMs (Belief Propagation, BP) [11] [14]. Las ecuaciones de la propagación de creencias para las redes HTM se derivan de la modificación de las ecuaciones de la propagación de creencias de las redes bayesianas. Al igual que las redes bayesianas, las redes HTM pueden ser entendidas como la codificación de las relaciones entre las variables aleatorias. En nuestro ejemplo, estas variables aleatorias corresponden a la coincidencia de patrones y las cadenas de Markov aprendidas en los múltiples niveles de la red.

Una entrada a la red suele denominarse evidencia. Por ejemplo, si se presenta una nueva imagen en la red para el reconocimiento, esta imagen es la evidencia para la red. La red propaga esta evidencia en la jerarquía para ajustar los estados de creencia en cada nodo de la red. Los estados de creencia pueden ser pensados como el grado de creencia que cada nodo asigna a sus coincidencias.

BP puede ser considerado como un mecanismo para calcular el efecto de una evidencia en los estados de creencia de una red utilizando cálculos locales. Existen dos variantes de algoritmos de propagación de creencias y pueden responder diferentes consultas en el mismo modelo. Una variante se llama sum-prop, calcula los estados de creencias de cada nodo dada la evidencia. El algoritmo de sum-prop es generalmente lo que la gente quiere decir cuando habla del algoritmo de propagación de creencias.

Otra variante del algoritmo BP se llama max-prop. También es conocido como Revisión de Creencias. Este algoritmo encuentra la mejor combinación posible de las asignaciones de los estados de los nodos en una red HTM, dada la evidencia.

La diferencia entre estas dos consultas puede ser sutil. En el caso de sum-prop, la creencia de los estados calculados en un nodo es el grado de creencia calculado sin suponer que los otros nodos han sido comprometidos a algún estado en particular. En el caso de max-prop, los estados de creencia corresponden al grado de creencia del estado de un nodo que forma parte de la mejor configuración posible. Esto supone el compromiso de los otros nodos a los estados particulares.

Se utiliza la siguiente notación para describir los mecanismos de propagación de creencias en las redes HTM. La mayor parte de esta notación es estándar y le resultará familiar a cualquier persona con una formación básica en la teoría de probabilidad.

C.1 Notación

C^K	Variable aleatoria que representa los patrones de coincidencia del nodo k^{th}. Dependiendo del contexto, C^k también se utiliza para representar el conjunto de coincidencia de patrones en el nodo k-ésimo.
c_i^k	La i-ésima coincidencia en el nodo k-ésimo.
G^k	La variable aleatoria representa las cadenas de Markov o grupos temporales del nodo k-ésimo. Dependiendo del contexto, también puede ser utilizado para representar el conjunto de grupos temporales del nodo k-ésimo.
g_i^k	La i-ésima cadena de Markov o grupo temporal en el nodo k-ésimo.
e^- y ^-e	La evidencia desde abajo, desde el punto de vista de un nodo. Este se compone de toda la evidencia observada por todos los descendientes de este nodo.
e^+ y ^+e	La evidencia de arriba, desde el punto de vista de un nodo. Este se compone de toda la evidencia observada por todos los nodos que no descendientes de este nodo.
$P(e^-\|G^k)$	La probabilidad de evidencia desde abajo teniendo en cuenta las cadenas de Markov o grupos temporales en el nodo *k*. Este es un vector de longitud igual al número de grupos.
$P(e^-\|g_i^k)$	La probabilidad de la evidencia desde abajo dado el grupo i-ésimo en el nodo *k*. Este es un escalar.

$P(e^-|C^k)$ La probabilidad de la evidencia de abajo dados los patrones de coincidencia en el nodo *k*. Este es un vector de longitud igual al número de patrones de coincidencia en el nodo *k*.

$P(e^-|c_i^k)$ La probabilidad de la evidencia de abajo dado el i-ésimo patrón de coincidencia en el nodo *k*. Este es un escalar.

$P(C^k|G^k)$ Las filas de esta matriz corresponden a las cadenas de Markov o grupos temporales y las columnas corresponden a las coincidencias. La entrada (i,j)-ésimo de esta matriz, $P(c_j|g_j)$, da la probabilidad de ver la coincidencia c_j dado que se ha visto la cadena de Markov o grupo temporal g_i.

λ^k El mensaje que envía el nodo k-ésimo a su padre. Este es un vector y es proporcional a $P(e^-|G^k)$. $\lambda^k(r)$ es el componente r-ésimo de este vector y es proporcional a $P(e^-|g_r^k)$.

y^k El mensaje de que el agrupador especial del k-ésimo nodo envía al agrupador temporal de ese nodo. Este es un vector de igual longitud al número de coincidencias y es proporcional a $P(e^-|C^k)$.

C.2 Memoria aprendida de un nodo

Las coincidencias y grupos forman la memoria básica del nodo en el cual opera la propagación de creencias.

El conjunto de coincidencias *C* se almacena dentro del agrupador espacial del nodo. Una coincidencia particular c_i, define la co-ocurrencia de los grupos temporales de su nodo hijo. Se puede considerar como un vector $[r_{m_1}, \ldots, r_{m_M}]$, donde las r's corresponden a los índices de los grupos de sus nodos dijos. Por ejemplo, si $M = 2$ y c_4 es la coincidencia del grupo 2 del nodo hijo m_1 y el grupo 5 del nodo hijo m_2, entonces $c_4 = [2,5]$. *C* es el conjunto de todos los c_i's.

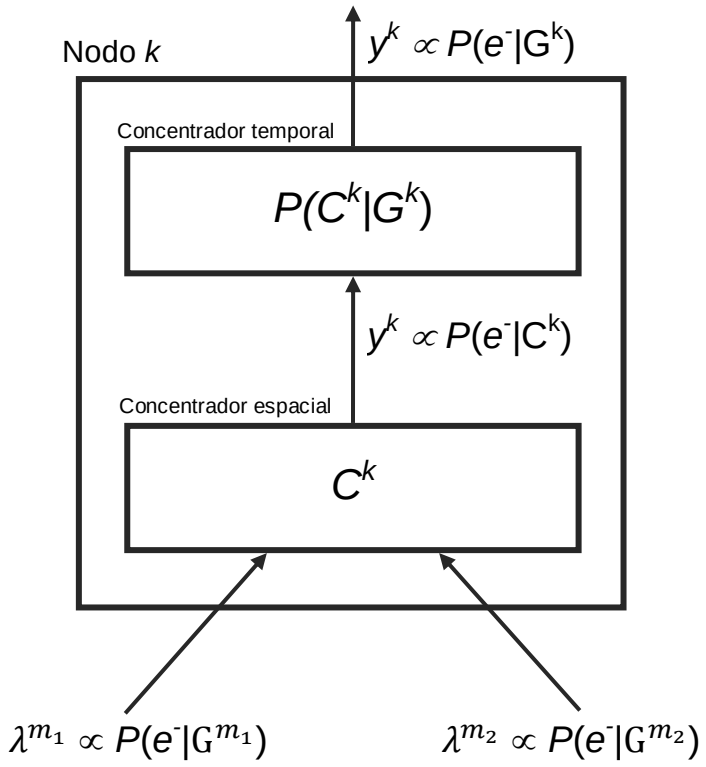


Figura C.1. Cálculos de la propagación de creencias dentro de un nodo

Para propósito de la propagación de creencias, necesitamos una matriz de probabilidad condicional que se pueden derivar de los grupos temporales. Esta matriz se denota como *P*(*C*|*G*). La i-ésima fila de esta matriz corresponde al grupo *i* en el nodo y la columna j de la matriz corresponde a las coincidencias *j*.

Esta matriz está formada por la combinación de los grupos temporales de información, ambos obtenidos mediante el aprendizaje. La fila *i* de la matriz *P*(*C*|*G*) se calcula de la siguiente manera. Las posiciones correspondientes a las coincidencias que pertenecen a este grupo están ocupadas por sus frecuencias de ocurrencia. El resto de las posiciones se llenan con ceros. Esta fila se normaliza para obtener la fila *i* de *P*(*C*|*G*).

C y *P*(*C*|*G*) constituyen la memoria de un nodo de HTM. Estas memorias se ven afectadas sólo a través del aprendizaje.

C.3 Cálculo de la propagación de creencias

La figura 61 muestra un segmento de una red HTM con 3 nodos. Se describen los cálculos de propagación de creencias con el nodo *k* como referencia. Este nodo tiene dos nodos hijos – nodo m_1 y m_2 –. Para la generalidad asumimos que este nodo tiene *M* hijos. Usamos m_1, m_2, …, m_M como los índices de los nodos hijos.

Este nodo recibe *M* mensajes de sus nodos hijos. El mensaje desde el nodo m_i será λ^{m_i} donde,

$$\lambda^{m_i} \propto P(e^-|\mathrm{G}^{m_i}).$$

Este mensaje es proporcional a la probabilidad de evidencia desde abajo dado los grupos temporales G^{m_i} en el nodo m_i. Este es un vector de longitud igual al número de grupos del nodo m_i.

El agrupador espacial calcula su salida *y* basado en las entradas. Esta salida es proporcional a $P(e^-|C)$. Esta salida es un vector de longitud N_c donde N_c es el número de coincidencias en el agrupador espacial. El componente i-ésimo de este vector corresponde a la coincidencia c_i. En general, esta coincidencia se puede pensar como se represente un vector de *M* números $[r_{m_1}, r_{m_2}, \ldots, r_{m_M}]$ donde la *r* representa el grupo de índices de sus hijos que constituyen esta coincidencia. La i-ésima componente de *y* se calcula como

$$y(i) = \alpha_1 \prod_{j=1}^{M} \lambda^{m_i}(r_{m_i})$$

Donde α_1 es una constante de escala arbitraria. Esta constante de escala por lo general se establece en un valor para que los mensajes no se encuentren con desbordamiento de punto flotante.

Dado que, $P(e^-|\mathrm{c}_i) = \prod_{j=1}^{M} P(e^-|g_{r_{m_i}}^{m_i})$ y dado que λ^{m_i} son proporcionales a $P(e^-|\mathrm{G}^{m_i})$ para toda *i*, este asegura que *y* es proporcional a $P(e^-|C)$.

El agrupador temporal calcula su salida basándose en la entrada del agrupador espacial. Esta salida es proporcional a $P(e^-|G)$ y es un vector de longitud N_g, el número de grupos temporales dentro del nodo. Este se calcula como sigue. El componente i-ésimo de este vector se calcula como,

$$\lambda(i) = \sum_{j=1}^{N_c} P(c_j|g_i) y(j)$$

Dado $P(e^-|g_i) = \sum_{j=1}^{N_c} P(c_j|g_i) P(e^-|c_j)$, el cálculo anterior asegura que λ es proporcional a $P(e^-|G)$.

La salida del agrupador espacial es la salida del nodo. Los cálculos en el nodo *k* son mostrados en la figura 61.