Difusión Fiable con Anonimato y Homonimia

Torres Rommel[1]; Jaramillo Danilo[1]; Jiménez Ernesto[2]

[1]Universidad Técnica Particular de Loja, Departamento de Ciencias de la Computación y Electrónica, Loja, Ecuador
[2]Universidad Politécnica de Madrid, Madrid, España

Resumen: Un sistema distribuido es un conjunto de procesos que intercambian mensajes a través de un conjunto de enlaces. La presente investigación propone un modelo de sistema distribuido con la capacidad de anonimato, que permita el intercambio de mensajes sin poder identificar el emisor ni el receptor. En este sentido se diseña e implementa una capa adicional de acuerdo al modelo de referencia OSI para el intercambio de información. Esta nueva capa denominada de Difusión Fiable permite la distribución de mensajes en el sistema distribuido cumpliendo con la característica de anonimato.

Palabras clave: sistemas distribuidos, anonimato, computación en la nube, difusión fiable.

Reliable Anonymous and Eponymous Diffusion

Abstract: A distributed system is a set of processes that exchange messages through a set of links. This research proposes a model of distributed system with the capability of anonymity, that allows the exchange of messages without being able to identify the sender nor the receiver. In this sense an additional OSI layer is designed and implemented for the interchange of information. This new layer called the Reliable Broadcast allows the distribution of messages in distributed system complying with the feature of anonymity.

Keywords: distributed systems, anonymity, cloud computing, reliable broadcast.

1. INTRODUCCIÓN

Un sistema de computación distribuido (SDC) se considera al conjunto de procesos que se comunican entre sí a través de enlaces de comunicación, los equipos pueden estar localizados físicamente en áreas dispersas.

El anonimato dentro de los sistemas distribuidos se produce cuando las entidades de computación (procesos, agentes, sensores, etc.) no tienen identificación, y por lo tanto no se puede distinguir unas de otras(Bonnet y Raynal, 2010). El anonimato es una propiedad de elevada relevancia, tan pronto como se está interesado en garantizar la privacidad. El anonimato se define como la cualidad de que un sujeto (equipo, proceso) no puede ser identificado de un conjunto de sujetos (conjunto anónimo)(Pfitzmann y Hansen, 2008). La homonimia es una estrategia para lograr anonimato, en la cual todos los integrantes del conjunto de sujetos comparten la misma identificación.

La computación distribuida trabaja sobre una topología dinámica en la cual los enlaces o dispositivos sobre los que se ejecutan los procesos son propensos a fallos. Por lo tanto enfrentar estos factores requiere un diseño de un protocolo fiable de comunicación de datos.

Existen dificultades en el diseño de algoritmos para computación distribuida usando anonimato y homonimia. En primer lugar los procesos pueden fallar, por lo tanto si no hay una forma de identificación es muy complejo determinar qué proceso específicamente falló y desde donde se deben realizar procesos de actualización de la información.

En segundo lugar suponiendo que trabajamos con enlaces fiables existe el problema de la sincronía. Ya que trabajamos con comunicaciones asíncronas en los cuales no se puede determinar el tiempo de transmisión de la información.

Para el primer y segundo punto existen soluciones que en general tratan de lograr un consenso entre los procesos a través de la elección de un proceso líder que medie la actualización y precisión de la información distribuida. En (Bonnet y Raynal, 2010) se realiza el estudio de consenso en los sistemas anónimos.

En tercer lugar los dispositivos para comunicarse entre sí, generalmente necesitan una identificación unívoca, que permita determinar el origen y destino de la información. En este sentido colocar anonimato y/o homonimato rompe la mayor parte de los modelos de comunicación actualmente utilizados.

La comunicación anónima puede ser utilizada como una alternativa para acceder los sistemas de comunicación y sistemas informáticos sin dejar rastro(Mansfield-Devine, 2011) para tratar de vulnerarlos. Por otro lado, la comunicación anónima es necesaria en sistemas donde la capacidad de procesamiento y almacenamiento

rovitor@utpl.edu.ec

es limitada como por ejemplo una red de sensores, en donde es importante tener la información generada por los sensores más que identificar de donde proviene. Otro ejemplo se puede dar en sistemas de voto electrónico donde para garantizar la imparcialidad de los resultados es necesario que la preferencia de cada votante se mantenga anónima. Otro ejemplo se da cuando los internautas no desean que sus preferencias de navegación sean usadas por terceros para inundarlos de información basura(Zhong y Zhang, 2015). Inclusive Facebook lanzó una aplicación que permite discusiones anónimas de grupos con intereses similares. (Lin, Sherr, y Loo, 2016).

Esta investigación muestra una solución inicial a la primera dificultad, implementando un protocolo de comunicación de difusión fiable con un primer acercamiento a la comunicación anónima entre procesos.

La presente investigación esta organizada de la siguiente manera. En el apartado 2 se hace una revisión del estado de la cuestión determinado cuales han sido las investigaciones y trabajos relacionados con el tema. En el apartado 3 se define y analiza el modelo del sistema distribuido que se pretende alcanzar. Luego en el apartado 4 se detalla la propuesta de difusión fiable para el modelo del sistema distribuido. En el apartado 5 se detallan los resultados obtenidos y los principales problemas en la implementación de nuestra propuesta. Finalmente se muestran las conclusiones y trabajo futuro.

2. ESTADO DE LA CUESTIÓN

Para que esta investigación sea autocontenida se hace una introducción a los conceptos relacionados.

2.1 Sistemas distribuidos

En (Ozalp Babaoglu, 1993) define a un sistema distribuido en una colección de procesos secuenciales p1, p2...pn y una red capaz de implementar canales de comunicación unidireccionales entre pares de procesos para el intercambio de mensajes. Una de las aseveraciones que se considera, es sobre la fiabilidad de los canales, pero se pueden enviar mensajes fuera de orden, además que todos los procesos se pueden comunicar con cualquier otro proceso. Se supone que la red de comunicación de estar fuertemente conectada (pero no necesariamente completamente conectada).

Otra definición, lo señala como un conjunto de computadores o procesos separados físicamente conectados entre sí por una red de comunicaciones. Generalmente son utilizados para la resolución de problemas complejos que necesitan la interconexión de servicios.

En la literatura se menciona como características generales de los sistemas distribuidos, las siguientes:

- Múltiples ordenadores conectados, pueden ser sistemas independientes los que a su vez pueden tener varios periféricos conectados al mismo.
- Estado compartido, puesto que al ser un conjunto de computadores compartiendo recursos, el estado para realizar una determinada tarea debe ser conocida por todos los componentes o procesos del sistema.
- Interconexión entre componentes, puesto que en el sistema distribuido estos deben estar conectados y de hecho debe existir una sincronización de procesos entre los mimos.

Estas características generales, pueden agrupar algunas características más específicas. Como parte de la característica de estado compartido se encuentra una sub característica mas específica, que hace relación a la consistencia de datos.

La consistencia de los datos en un sistema distribuido único compuesto por una variedad de máquinas independientes, debe tener un solo estado general o global conocido por todos. El estado global mantiene valores para los componentes como la hora del sistema, los datos compartidos para todos, tablas de procesos, tablas de mantenimiento del sistema, etc. Cabe señalar que la inconsistencia en los sistemas distribuidos es uno de los aspectos de mayor importancia interés de la comunidad científica.

La característica de interconexión puede agrupar en las siguientes subcaracterísticas de:

- Compartición de recursos, esta característica hace referencia a la principal función del sistema distribuido que permite la utilización de recursos y datos a los diferentes participantes del sistema, permitiendo el intercambio de información.
- Sistemas abiertos, esta característica hacer referencia a la posibilidad de agregar nuevos recursos tanto de hardware como de software sin modificar la arquitectura inicial.
- Tolerancia a fallos, el sistema debe ser capaz de recuperar los datos y dejarlos tal cual estuvieron de forma permanente en el estado consistente anterior al fallo.
- Escalabilidad, donde el sistema puede ser ampliado fácilmente, sin la necesidad de que se afecte la eficiencia del sistema.
- Seguridad, al contar con varios componentes conectados es necesario proteger tanto los recursos y datos de un acceso no autorizado al sistema.

2.2 Anonimato

“Carácter o condición de anónimo", esta definición consta en el diccionario de la Real Academia de la Lengua (RAL). Donde también anónimo, se ha definido como "secreto del autor que oculta su nombre". De esta manera la condición de

anonimato en comunicaciones debe permitir ocultar tanto en el nombre del emisor y receptor de un mensaje determinado.
sección debe reflejar el marco referencial que sustente el contenido del artículo, puede incluir: marco teórico, metodología, materiales o métodos y/o consideraciones generales. El título de esta sección puede ajustarse de acuerdo a las necesidades de cada artículo.

La comunicación anónima en redes de computadoras ha sido discutida por algunos autores. En su estudio (Pfitzmann y Hansen, 2008) hace una definición de anonimato, indetectabilidad, inobservabilidad, así como también describe al uso de seudoanonimato como mecanismos para lograr la comunicación anónima. Los autores de (Ren y Wu, 2010) hacen un análisis de la comunicación anónima en redes de computadores enfocándose en aspectos de seguridad, considerando la necesidad de protección de los datos.

2.3 Homonimia

Según la RAL, se define a homonimia como una cualidad del homónimo. En el mismo diccionario se define a homónimo como "Dicho de dos o más personas o cosas: Que llevan un mismo nombre". En sistemas de comunicación se presenta como la condición de identificar a todos los elementos de un sistema distribuido con una misma nomenclatura. En (Arevalo, Anta, Imbs, Jimenez, y Raynal, 2012) se detalla que homonimia como una generalización de dos casos extremos: (1) los procesos que tiene identificadores únicos y (2) los procesos que tiene el mismo identificador para todos (anonimato).

2.4. Avances y Retos en coordinación distribuida

Los principales avances y retos a resolver dentro de la coordinación distribuida en la computación en nube son:

2.4.1. Acuerdo en múltiples valores y detectores de fallos

El problema de acuerdo en múltiples valores, k-set-agreement, es un problema de coordinación de n procesos en un sistema distribuido donde los procesos pueden fallar (por fallos de caída-parada)(Chaudhuri, 1992). Este problema permite analizar la relación entre el número de procesos caídos y el mínimo número de valores distintos que los procesos pueden decidir. El problema de k-set-agreement se define de forma que de n valores propuestos, como mucho k valores distintos pueden ser decididos. Más formalmente, cada proceso propone un valor y todo proceso que no falle (llamado proceso correcto) tiene que decidir un valor (propiedad de terminación), de tal forma que cualquier valor decidido debe ser uno de los valores propuestos (propiedad de validez), y no pueden existir más de k valores distintos que hayan sido decididos (propiedad de acuerdo). El parámetro k define el grado de coordinación: k = 1 es el caso más restrictivo (normalmente llamado consenso), mientras que k=n-1 es el caso menos restrictivo (normalmente a este caso se le llama set agreement. Para los sistemas distribuidos asíncronos se ha adoptado un nuevo enfoque aumentando al sistema asíncrono un detector de fallos.

Un detector de fallos (Chandra y Toueg, 1996) es una herramienta distribuida que cada proceso puede invocar para obtener información acerca de los fallos de los procesos. Hay muchas clases de detectores de fallos en función de la calidad y el tipo de la información devuelta.

En (Arevalo y cols., 2012) se presentan nuevos tipos de detectores de fallos: detectores de fallos para homónimos (es decir, un mismo identificador puede ser compartido por más de un proceso). La homonimia es una extensión del anonimato (es decir, cuando los procesos no tienen identificadores). Estos nuevos detectores de fallos se definen, comparándolos entre sí (para saber la relación "más débil que"), e implementándolos.

2.4.2. Consenso

Es un servicio que permite acordar un mismo valor entre todos los equipos de la nube. Es un caso particular de **k- set-agreement.** Concretamente, consenso es el caso cuando $k = 1$. La consecución de este consenso se hace mucho más difícil de conseguir cuando los equipos pueden fallar (por ejemplo porque un equipo o proceso que ha dejado de funcionar).

Resolver el problema del consenso es muy importante porque se encuentra en el núcleo de muchas aplicaciones distribuidas. El consenso puede ser definido de forma que cada proceso propone un determinado valor, y todos los procesos correctos (es decir, todos los procesos que no fallen por caerse o dejar de funcionar) tienen que estar de acuerdo en tomar una misma decisión. Esta decisión tiene que ser uno de los valores propuestos por alguno de los procesos. Más formalmente, en el problema de consenso cada proceso propone un valor, y cada proceso correcto tiene que decidir un mismo valor v (propiedad de terminación), de tal forma que el valor decidido v es uno de los valores propuestos (propiedad de validez), y no pueden existir dos procesos que decidan de forma diferente del valor v (propiedad de acuerdo).

Es un resultado ampliamente conocido en la literatura el hecho de que es imposible alcanzar consenso cuando el sistema distribuido es asíncrono (Fischer, Lynch, y Paterson, 1985). Este hecho es el que ha llevado a la creación de los detectores de fallos como una forma de salvar este resultado de imposibilidad.
En (Arevalo y cols., 2012) se implementa un protocolo para resolver consenso, en el cual se utiliza un detector de fallos con homónimos (llamado HΩ, es decir, **homonymous-omega**). Este trabajo es muy importante porque resuelve consenso en sistemas distribuidos con características muy avanzadas: membresía desconocida y existencia de procesos homónimos.

Un reto a resolver actualmente es diseñar nuevos protocolos para acordar valores comunes que permitan sincronizar los recursos que están utilizando de forma distribuida los equipos pese a estar trabajando en la nube de forma que se respete la privacidad de los usuarios (anonimia y homonimia).

3. SISTEMA DISTRIBUIDO CON ANONIMATO/HOMONIMIA

Esta investigación pretende avanzar en aspectos de la computación en la nube (cloud-computing) que son objeto de estudio a nivel internacional; el coordinar elementos de forma distribuida sin poder identificar a los elementos del sistema de forma unívoca (o sin poder identificarlos de ninguna manera). Para lograrlo se ha definido los componentes de la propuesta basándonos en un esquema cliente servidor. La granja de servidores atiende las solicitudes de los clientes. Y los clientes acceden a un servicio proporcionado por la granja de servidores.

Como se muestra en la Figura 1 existen dos componentes principales: a) la granja de servidores, y, b) los clientes que acceden al servicio.

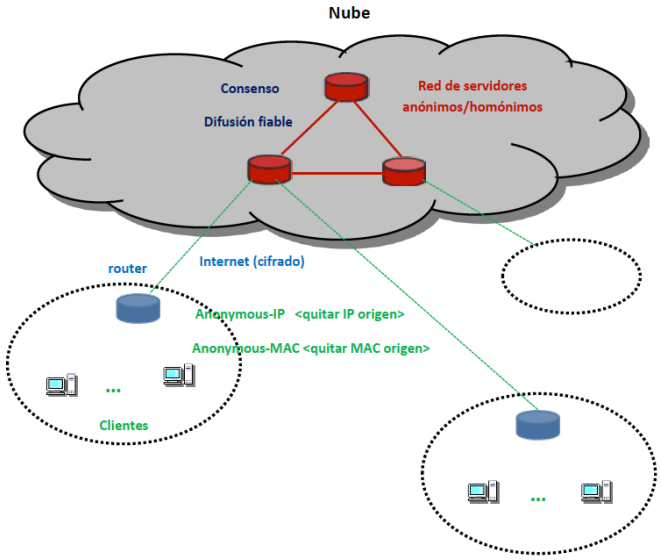

Figura 1. Comunicación anónima en un sistema distribuido

general, se recomienda que tanto el análisis como la discusión de resultados se presenten de manera integrada. Es necesario que los resultados queden clasificados dentro de arreglos significativos: tablas, cuadros, gráficos, diagramas figuras, estadísticas, etc., que permitan reflejarla coherencia y la argumentación de los mismos. Cada representación de datos debe ir seguida de un texto que explique, interprete, contraste, amplíe… lo que los datos significan.

3.1 La granja de servidores

La granja de servidores esta compuesta por un conjunto de procesos, que se comunican entre sí (intercambiando mensajes) a través de un conjunto de enlaces. En esta propuesta todos los servidores ejecutan el mismo proceso en forma coordinada. Un proceso es un conjunto de instrucciones que se ejecutan secuencialmente. Por lo tanto, un proceso puede fallar en cualquier momento debido por ejemplo a que el sistema anfitrión dejo de funcionar por causas físicas (desconexión, falla de energía) como lógicas (sobreprocesamiento, mal diseño o implementación de algoritmos). Esta característica de funcionamiento define a su vez dos subconjuntos de procesos: a) procesos fallidos b) procesos correctos.

Otro componente de la granja de servidores son los enlaces. Se trabaja con enlaces que son propensos a fallos y que trabajan en forma asíncrona. Estás características, sobre la cual se ha planteado la presente investigación, hacen que el entorno de comunicación no sea fiable. Para lograr la fiabilidad se ha definido un proceso de difusión anónima, el mismo que es descrito en el apartado 4.

Las cualidades que se pretende cumplir en la granja de servidores son el anonimato y la fiabilidad.

La cualidad de anonimato se refiere a que el intercambio de mensajes en la granja de servidores se lo hace sin identificar el origen ni el destino. Se utiliza una estrategia de homonimia. En nuestro caso identificamos el destino con una dirección de broadcast y el origen con un dirección preasignada que adicionalmente es la misma para todos los mensajes que se intercambian en el sistema.

La cualidad de fiabilidad involucra que los mensajes que envían los procesos correctos son recibidos por un subconjunto de procesos correctos. Se ha definido que la granja de servidores es fiable si la mayoría de procesos correctos procesan el mensaje correctamente.

Para abordar la complejidad de la interrelación de los componentes de esta propuesta y cumplir con las cualidades requeridas por la granja de servidores se definieron tres estrategias que se detallan a continuación.

- Consenso en un sistema anónimo
- Detector de fallos para el sistema anónimo
- Difusión anónima

3.1.1 Consenso en el sistema anónimo

El consenso definido como la capacidad de decidir sobre un valor. En computación distribuida los procesos deben acordar valores de tal forma que la información sobre la que trabajan sea confiable y actualizada. Por ejemplo el servicio brindado por una aplicación de almacenamiento distribuido, tal como google drive o dropbox, en la cual los clientes pueden acceder al mismo archivo simultáneamente, el sistema distribuido debe consensuar cual es la última versión del archivo en la que están trabajando los clientes.

En esta propuesta además, los procesos son anónimos, no se pueden identificar, esto plantea algunas incidencias que debemos resolver: a) Conocer cuál es el número de procesos servidores correctos. 2) Basado en la cantidad de procesos correctos definir cuándo el sistema distribuido es estable de tal forma que pueda consensuar un valor.

3.1.2 Detección de fallos para el sistema anónimo

Los procesos pueden fallar, por lo que es importante de terminar de alguna forma cuales son los procesos que han fallado y como afectan al sistema distribuido. En nuestro caso al no tener identificación es más complejo determinar específicamente que proceso falló. Para resolver esta incidencia se ha definido una estrategia basada en grupo, es decir no conocemos que proceso específicamente falló si no más bien que cantidad de procesos son los correctos. Tanto en el proceso de consenso y de detección de fallos usamos un proceso líder. El papel del proceso líder es obtener la información del estado del sistema distribuido e informar a los demás procesos servidores.

3.1.3 Difusión anónima

Para la granja de servidores se ha propuesto una comunicación anónima entre los procesos. Como se ha mencionado anteriormente, nuestra capa modifica los direcciones de origen y destino de cada paquete que se intercambia en el sistema. Se usa dirección de grupo, conocida como dirección de broadcast para identificar el destino. La difusión mediante broadcast y multicast es un problema ampliamente revisado en la literatura (Défago, Schiper, y Urban, 2004). De la misma forma se usa una dirección de fija preseleccionada para identificar al origen.

3.2. Comunicación con clientes

Para la interacción de los clientes con la granja de servidores, Figura 1 se han definido dos componentes

- *Acceso a la granja de servidores.* Los clientes, que pueden estar ubicados en red diferente, deben acceder al servicio a través de un proceso servidor que servirá de pasarela a la granja de servidores. Este proceso servidor debe mantener un registro de la petición del cliente de tal forma que pueda pasar su requerimiento y luego, una vez procesado por la granja de servidores, recibir la respuesta respectiva.

- *Comunicación anónima del cliente a la granja de servidores.* La comunicación de los clientes con la granja de servidores debe garantizar que las peticiones y la interacción no se puedan rastrear, sea anónima. Existen algunas estrategias que pueden garantizar en parte el anonimato, por ejemplo: envío a través de direcciones Multicast, el uso de sistemas de terceros como el propuesto por The Onion Router, TOR (Haraty y Zantout, 2014), o técnicas de encapsulación a través Generic routing Encapsulation, GRE (Hanks, Li, y Traina, 1994), inclusive se puede utilizar traducción de direcciones, NAT. Estas estrategias requieren cambios en la equipos de comunicación intermedios entre el cliente y la granja de servidores. Todas estas estrategias se pueden complementar usando técnicas de criptografía para mejorar la seguridad y el anonimato.

4. DIFUSIÓN FIABLE CON ANONIMATO

La presente investigación está enfocada en difusión anónima en la granja de servidores. Nuestra propuesta desde un punto de vista de capas, incluye una nueva capa ubicada entre la capa de enlace de enlace de datos y la capa de transporte de acuerdo al modelo de referencia OSI.

La Figura 2 muestra la interacción del emisor y el receptor de acuerdo al modelo de capas propuesto. La capa que se propone se la denomina *Capa de difusión fiable*, que brinda servicios a las capas superior e inferior a través de primitivas.

En el emisor la capa superior utiliza el servicio de difusión fiable a través de la *rb_send*(*msji*). A su vez se utiliza la primitiva *send(pkt)* de la capa inferior para la envío del paquete o mensaje que se ha generado o preparado en la capa propuesta.

En el receptor nuestra propuesta toma el paquete de la capa subyacente a través de la primitiva *recibir(pkt)* y pretende entregar a la capa superior la información en forma fiable, ordenada, completa y sin errores a través de la primitiva *rb_del*(*mi*).

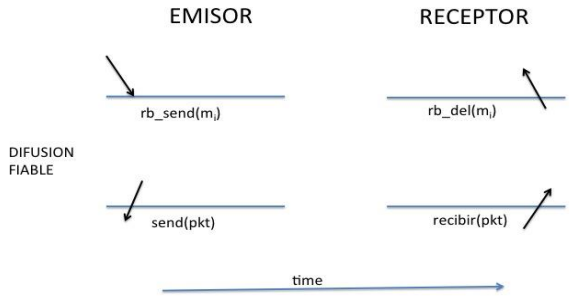

Figura 2. Modelo de capas para la difusión fiable

Para detallar el funcionamiento de nuestra propuesta se propone una máquina de estado finito tanto para el receptor como para el emisor.

El receptor, como se puede ver en la Figura 3 puede estar en dos estados *Esperar llamada de abajo mi* y *Esperar converger i*. En el estado de espera el receptor esta escuchando por un paquete. En el momento que llega el mensaje envía una respuesta de confirmación del mensaje a la red. En nuestro modelo para garantizar el anonimato todos los paquetes se envían mediante una dirección de difusión. Debido a que actualmente los enrutadores no permiten pasar mensajes de difusión (broadcast) para que nuestro modelo funcione se les debe incorporar el software necesario para gestionar los mensajes con direcciones de difusión de nuestro modelo. Una vez que ha enviado el mensaje de confirmación el receptor cambia a un estado de convergencia, para retornar al estado de espera el receptor espera mensajes de confirmación de que el sistema distribuido es estable y que el emisor ha procesado correctamente el mensaje. Con esta información el receptor entrega en forma fiable el mensaje a la capa superior.

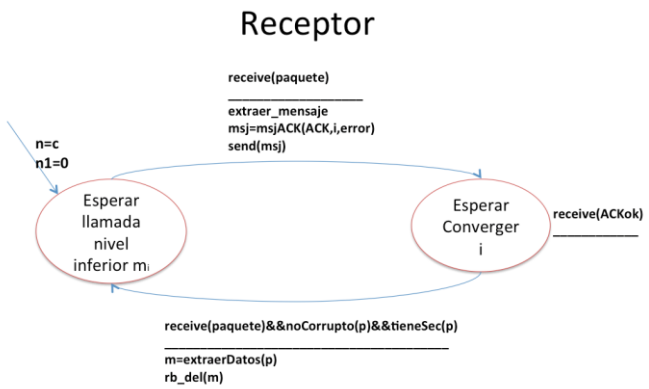

Figura 3. FMS Receptor

El emisor como se ve en la Figura 4 puede estar en tres estados: a) *Esperar llamada i de arriba* b) *Esperar ACK de i* y c) *Esperar converger*. Cuando se inicia el sistema distribuido los nodos (procesos, equipos o servidores) generan un número randómico *i* para identificar cada mensaje a enviar (Para mantener un anonimato suave, pseudoanonimato, generamos estos identificadores en forma aleatoria y no secuencial) y para determinar la estabilidad del sistema distribuido comenzamos suponiendo que existen dos servidores correctos, esta valor se actualizará en forma dinámica con las interacciones subsecuentes entre los procesos.

El emisor activa su estado *Esperar llamada i de arriba* cuando la primitiva *rb_send(m,i)* es usada por la capa superior. En este estado se crea un mensaje, con identificación homónima en la dirección de origen y identificación de grupo (broadcast) en la dirección de destino. El mensaje contiene la identificación del mensaje *i*, el mensaje e información para el control de errores, es entregado a la capa inferior a través de la primitiva *send(pkt)*. Al mismo tiempo el emisor inicia un *timer* usado para dos fines: 1) determinar la pérdida de paquetes, en una estrategia similar a TCP, y; 2) determinar que el sistema distribuido es estable y puede tomar decisiones de consenso.

Una vez que se ha enviado el mensaje el emisor cambia su estado a *Esperar ACK de i*. Una vez recibido el mensaje de confirmación, el emisor cambia al estado de *Esperar converger*. En este estado registra el número de mensajes de confirmación, ACK, que han llegado del sistema distribuido en el tiempo definido en *timer*. Con esta información se actualiza el número de procesos correctos que tiene en ese momento el sistema distribuido. Una vez que el emisor ha recibido mensajes de confirmación de la mayoría de procesos correctos entonces se cambia al estado *Esperar llamada i de arriba*, caso contrario vuelve al estado de *Esperar ACK de i*.

5. RESULTADOS Y DISCUSIÓN

Una vez definidos los algoritmos de comunicación para el emisor y receptor para la difusión fiable. Se procede a la implementación. La tabla 1 muestra los parámetros y los valores utilizados para definir los escenarios de implementación para la difusión fiable con anonimato.

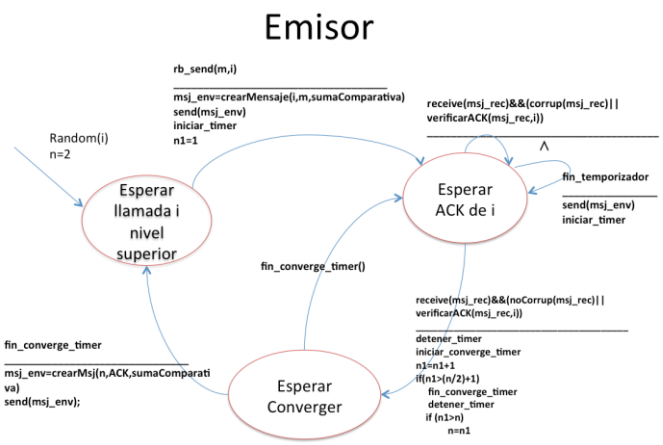

Figura 4. FMS Emisor

En una primera fase se seleccionó como plataforma de desarrollo sistemas operativos basados en Windows Sin embargo luego de probar con algunas interfaces de desarrollo, tales como, udp sockets, raw sockets, winpcap, no se pudo obtener anonimato en el intercambio de mensajes.

Con este antecedente se implementó en plataformas basadas en linux, se utilizó la interfaz libpcap. El resultado exitoso de las pruebas de la implementación se muestra en la Figura 5 que corresponde a una captura realizada con el analizador de paquetes Wireshark (http://www.wireshark.org).

Lo importante del resultado es que se logró homonimia y anonimato tanto en la capa de enlace de datos como en la capa de red. En otras palabras la dirección física y lógica de origen están con un valor que se ha prefijado para los paquetes y tramas que se intercambian en el sistema. En el mismo sentido la dirección física y lógica de destino de todos los paquetes es una dirección de broadcast.

6. CONCLUSIONES Y TRABAJO FUTURO

En esta investigación se pretende avanzar en unos aspectos de la computación en nube (cloud-computing) que son objeto de estudio a nivel internacional, coordinar elementos de forma distribuida sin poder identificar a los elementos del sistema de forma unívoca (o sin poder identificarlos de ninguna manera). De esta forma se quiere avanzar hacia servicios en nube donde la privacidad sea un aspecto muy relevante entre los servicios a proporcionar por el sistema.

Por lo tanto esta investigación propone un modelo de sistema distribuido que permite el intercambio anónimo de información entre clientes y la granja de servidores.

La principal contribución de esta investigación es el desarrolló una API que implementa los servicios de difusión fiable en la granja de servidores. Esta API en su versión inicial trabaja con elementos anónimos u homónimos en un mismo dominio de broadcast bajo plataforma linux en lenguaje de programación C++.

Como resultado del presente trabajo se plantean acciones o iniciativas que se proponen como un trabajo a futuro entre las cuales podemos mencionar.

Pese a que se ha mencionado en este trabajo es necesario investigar sobre las estrategias y rendimiento del sistema distribuido cuando los servidores se encuentren en diferentes dominios de broadcast.

Se ha mencionado formas de que los clientes tengan acceso anónimo a la granja de servidores, sin embargo queda pendiente la implementación y análisis de cual es la mejor con respecto al modelo de sistema distribuido planteado.

Este cambio de paradigma en la comunicación abre una nueva perspectiva para el diseño de enrutadores y equipos intermedios que permitan el intercambio de comunicación anónima.

Esta etapa de la investigación cumplió con el objetivo de generar un modelo para sistemas distribuidos usando difusión fiable con anonimato por lo que en esta etapa no se consideró hacer pruebas de rendimiento del servicio.

AGRADECIMIENTO

Esta investigación es parte del proyecto que se desarrolla en tres Universidades ecuatorianas, la Escuela Politécnica Nacional, la Universidad Técnica de Ambato y la Universidad Técnica Particular de Loja, además cuenta con la colaboración de la Universidad Politécnica de Madrid. Este proyecto ha sido cofinanciado por CEDIA en la séptima convocatoria denominada CEPRA.

REFERENCIAS

Arevalo, S., Anta, A. F., Imbs, D., Jimenez, E., y Raynal, M. (2012). Failure detectors in homonymous distributed systems (with an application to consensus). En *Proceedings of the 2012 ieee 32nd international conference on distributed computing systems* (pp. 275–284). Washington, DC, USA: IEEE Computer Society. http://dx.doi.org/ 10.1109/ICDCS.2012.13

Bonnet, F., y Raynal, M. (2010). Consensus in anonymous distributed systems: Is there a weakest failure detector? *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, 206–213. doi: 10.1109/AINA.2010.19

Chandra, T. D., y Toueg, S. (1996, marzo). Unreliable failure detectors for reliable distributed systems. *J. ACM*, *43*(2), 225–267.http://doi.acm.org/10.1145/226643.226647

Chaudhuri, S. (1992). More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, *105*, 132–158.

Défago, X., Schiper, A., y Urbán, P. (2004). Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR)*. http://dl.acm.org/citation.cfm?id=1041682

Fischer, M. J., Lynch, N. A., y Paterson, M. S. (1985, abril). Impossibility of distributed consensus with one faulty process. *J. ACM*, *32*(2), 374–382. http://doi.acm.org/10.1145/3149.214121

Hanks, S., Li, T., Farinacci, D., y Traina, P. (1994, octubre). *Generic Routing Encapsulation (GRE)* (n.o 1701). RFC 1701 (Informational). IETF. Obtenido de http:// www.ietf.org/rfc/rfc1701.txt (Noviembre, 2015)

Haraty, R., y Zantout, B. (2014). The TOR data communication system. *Communications and Networks, Journal . . . , 16*(4), 415–420. Obtenido de la base de datos IEEE.

Lin, D., Sherr, M., y Loo, B. T. (2016). Scalable and anonymous group communication with mtor. *Proceedings on Privacy Enhancing Technologies*, *2016*(3), 1–18.

Mansfield-Devine, S. (2011, enero). Anonymous: serious threat or mere annoyance? *Network Security*, *2011*(1), 4–10. doi: 10.1016/S1353-4858(11)70004-6

Ozalp Babaoglu, K. M. (1993). Consistent Global States of Distributed Systems : Fundamental Concepts and Mechanisms. (January)

Pfitzmann, A., y Hansen, M. (2008). *Pseudonymity, and Identity Management. A Consolidated Proposal for Terminology, February 2008* (Inf. Téc.).

Ren, J., y Wu, J. (2010, marzo). Survey on anonymous communications in computer networks. *Computer Communications*, *33*(4), 420–431. doi: 10.1016/j.comcom.2009.11.009

Zhong, S., y Zhang, M. (2015). Research on the personal privacy information protection problem based on computer security technology. En *2015 international conference on automation, mechanical control and computational engineering.*