UNIVERSIDAD POLITÉCNICA DE PUEBLA

PROGRAMA ACADÉMICO DE
INGENIERÍA EN INFORMÁTICA

Recuperación y Almacenamiento de Datos en Ambientes Distribuidos

Javier Ramírez Muñoz

Reporte Técnico PI1-32-12-09

COMITÉ EVALUADOR

Dr. Luis Alberto Morales Rosales (Asesor)
cDr. Javier Velázquez Sandoval (Sinodal)
cDr. Eduardo López Domínguez (Sinodal)

PROFESOR(A) DE PROYECTO DE INVESTIGACIÓN II

Dra. María Auxilio Medina Nieto

Juan C. Bonilla, Puebla
Diciembre 2009

Í N D I C E

Capítulo 1. Planteamiento del problema

1.1 Introducción 4
1.2 Objetivo general 5
1.3 Objetivos específicos 5
1.4 Justificación 5
1.5 Alcances y limitaciones 6
1.6 Cronograma de actividades 7
1.7 Recursos de hardware y software 9

Capítulo 2. Marco teórico

2.1 Redundancia y monitorización 10
2.2 Control de congestión de la red 11
2.3 Identificación de desconexiones en la red 12
2.4 Protocolos UDP y Tcp. 13
2.5 Protocolo Tcp friendly 16
2.5.1 Tcp-friendly para Unicast y Multicast 18
2.5.2 Windows-based 19
2.5.3 Rate-based 19
2.6 Modelo Tfrcp 19
2.7 Protocolo Tfrc 20

Capítulo 3. Metodología

3.1 Entendimiento de los conceptos disponibilidad y alta disponibilidad en sistemas distribuidos 21
3.2 Explicación de las características que conforman la arquitectura propuesta 22
3.3 Explicación de las herramientas utilizadas en la arquitectura 22
3.4 Selección, estudio y explicación del protocolo TCP frienfly rate control 22
3.5 Estudio, explicación general e implementación de las clases con las que cuenta el protocolo Tfrc. 23
3.6 Diseño de la arquitectura. 24
3.6.1. Capa de identificación de desconexiones “IDR”. 26
3.6.2. Capa de control de congestión “CCR”. 28

Capítulo 4. Implementación

4.1 Diseño y explicación del diagrama de clases 30
4.2 Especificación de parámetros a considerar de la arquitectura 31
4.3 Especificación de los sistemas operativos utilizados 31
4.4 Obtención de la dirección IP del equipo servidor. 32
4.5 Modificación de clases AudioAppClient.java y AudioAppServer.java 32
4.6 Compilación de clases utilizadas. 33
4.7 Ejecución de la aplicación 34
4.8 Aplicación de tráfico dentro del grupo de trabajo 35

Capítulo 5. Resultados
5.1 Creación e interpretación de las gráficas obtenidas 36

Capítulo 6. Conclusiones
6.1 Características de la arquitectura propuesta 38
6.2 Ventajas que contiene la arquitectura distribuida 39
6.3 Desventajas encontradas en la arquitectura distribuida 39

Referencias 40

ÍNDICE DE FIGURAS Y TABAS

Tabla 1.6.A Cronograma de actividades de Proyecto de Investigación 1. 7
Tabla 1.6.B Cronograma de actividades de Proyecto de Investigación 2. 8
Tabla 2.4 Segmento de cabecera TCP 15
Figura 2.5.A Flujos de información TCP-Friendly 16
Figura 2.5.B Segmentos de cabecera TCP. 17
Figura 2.5.1.A Flujos de información TCP-Friendly versión extendida. 18
Figura 3.6.A Esquema general arquitectura distribuida RADAD. 24
Figura 3.6.B Esquema general arquitectura propuesta RADAD 25
Figura 3.6.1 Mecanismo de trabajo capa IDR 26
Figura 3.6.2.A Ecuación para calculo de tasa de envió 28
Figura 3.6.2.B Cálculo tiempo total RTT. 29
Figura 3.6.2.C Determinación del tRTO. 29
Figura 3.6.2.D Determinación de la velocidad de transmisión 29
Figura 4.1 Diagrama de clases arquitectura RADAD 30
Figura 4.4.A Consola de Windows 33
Figura 4.4.B Consola de Windows 33
Figura 4.5.A Modificación dirección IP en la clase AudioAppClient.java 33
Figura 4.5.B Ciclo While en la clase AudioAppServer.java 34
Figura 4.6 Compilación de clases de la arquitectura 34
Figura 4.7.A Ejecución de la clase AudioAppServer.java 35
Figura 4.7.B Ejecución de la clase AudioAppClient.java 35
Figura 4.8.A Software Wireshark para inyección de tráfico dentro de la red. 36
Figura 4.8.B Inyección de tráfico dentro de la red. 36

Capítulo 1. Planteamiento del problema

1.1 Introducción

Tras la evolución de los sistemas informáticos, muchas de las empresas han optado por la utilización de sistemas distribuidos. Los sistemas distribuidos se pueden definir como aquellos componentes de hardware y software que se localizan en distintas computadoras, unidos mediante una red computacional que pueden coordinar y comunicar sus acciones sólo mediante el paso de mensajes. Algunos ejemplos de los sistemas distribuidos son: internet, intranets, computación móvil y redes ad hoc [Coulouris 2001].
Las principales características de los sistemas distribuidos son [Coulouris 2001]:

- Elementos independientes no compartidos
- Carecen de un reloj general
- Mantienen la tolerancia a fallas
- Concurrencia

El propósito de utilización de los sistemas distribuidos es para poder compartir recursos, donde los recursos pueden ser bases de datos, programas de computadora, periféricos o medios de almacenamiento [Kshemkalyani y Singhal 2008].

Dentro de un sistema distribuido se busca garantizar la disponibilidad de la información, la cual puede ser definida como el porcentaje de tiempo en que un sistema se encuentra en funcionamiento para proporcionar un servicio [Gray 1985]. El problema de no tener una disponibilidad de la información de manera inmediata produce en las empresas pérdidas económicas, pérdidas en tiempo de procesamiento de datos, disminución en toma de decisiones, desperdicio de los recursos de tecnología de la información y aumento de costos para los proceso de la organización. Estos problemas pueden llevar al cierre de las operaciones [Stalin 2008].

Sin embargo, actualmente se requieren para entornos distribuidos una alta disponibilidad de la información. En este caso, la alta disponibilidad consiste en asegurar que los servicios continúen operando a pesar que el hardware o software fallen parcialmente [Farley 1997]. Entre los principales problemas que se presentan para garantizar la alta disponibilidad de la información en un entorno distribuido se encuentran:

- Control de congestión dentro de la red
- Identificación de desconexiones dentro de los enlaces de red
- Retardos del envío de paquetes dentro de una red
- Ruteo de paquetes

En este trabajo se propone el diseño de una arquitectura distribuida para garantizar la alta disponibilidad contemplando los problemas:

- Control de congestión dentro de la red
- Identificación de desconexiones dentro de los enlaces de red

1.2 Objetivo general

Diseñar una arquitectura distribuida identificando los servicios necesarios para almacenar, recuperar la información garantizando su alta disponibilidad.

1.3 Objetivos específicos

- Identificar los servicios de red que garanticen la alta disponibilidad de la información
- Seleccionar un método de distribución y replicación de información para un ambiente distribuido
- Creación de un escenario donde se verifique la viabilidad de la arquitectura diseñada

1.4 Justificación

En esta investigación, con el fin de reducir los principales problemas a los que se enfrentan las empresas cuando quieren garantizar alta disponibilidad de la información al momento de acceder a un servicio, se identifican los siguientes:

- *Identificación en desconexiones dentro de los enlaces*.- En el instante que se interrumpe una conexión entre el cliente y el servidor, dando como consecuencia el impedimento de envío de paquetes entre ellos. Para poder atacar este problema, se analizará la funcionalidad del protocolo "Stop and Wait ARQ".

- *Retardos de los paquetes*.- Durante el envío de un paquete enviado por el emisor se presenta un retardo hasta llegar a su destino. El principal problema es determinar si este retardo fue provocado por la congestión dentro de la red, pérdida del paquete enviado o un número excesivo de procesos atendidos por el servidor y determinar si los paquetes llegaron o no a su destino.

- *Control de congestión dentro de la red*.- Cuando existen demasiados paquetes que intentan establecer una comunicación, se degrada el rendimiento de la red. Para tratar de evitar un estado de congestión en la red, los ruteadores pueden descartar aleatoriamente datagramas cuando sobrepasan un cierto nivel de ocupación. Un método que se usa es "Random Early Detection", donde la pérdida de paquetes obliga a las aplicaciones a disminuir su tasa de transmisión.

Después de haber identificado y analizado los principales problemas previamente mencionados, se propone diseñar una arquitectura distribuida, en la cual se identifican los servicios necesarios para generar una alta disponibilidad de la información.

1.5 Alcances y limitaciones

Para el alcance de este proyecto, en esta investigación se llevará a cabo el diseño de una arquitectura distribuida que se delimita sólo al análisis de los principales servicios de red que garanticen la disponibilidad de la aplicación. Para garantizar la calidad de estos servicios se realizará lo siguiente:

- Se estudiará, analizará y expondrán principales problemas que se presentan para garantizar la alta disponibilidad de la información en un entorno distribuido

- Se exhibirá un protocolo de solución existente para los problemas:
    - Control de congestión de la red
    - Identificación de desconexiones dentro de la red

Posterior al análisis de los principales problemas que se presentan para garantizar la alta disponibilidad de la información en un entorno distribuido, se diseñará una arquitectura distribuida que estará compuesta por dos capas: "IDR y CCR", donde en cada una de ellas se implementan servicios necesarios que satisfagan la tarea de comunicación y envío de mensajes, haciendo uso del protocolo "Tcp-friendly-tfrc", con la finalidad de combatir los dos problemas que se abarcan en este documento que se presentan al momento de garantizar la alta disponibilidad: control de congestión dentro de la red e identificación de desconexiones dentro de los enlaces de red, así como uso de un script previamente codificado, el cual se enfocará a la tarea exclusivamente de mostrar la funcionalidad del protocolo propuesto.

En el proceso de llevar a cabo la implementación, no se analizarán o propondrán soluciones a la identificación de los retardos de paquetes ni al ruteo de paquetes.

1.6 Cronograma de actividades

Las tablas 1.6.A y 1.6.B muestran las actividades principales de este proyecto.

Tabla 1.6.A Cronograma de actividades de Proyecto de Investigación 1.

Actividad	Mayo		Junio				Julio				Agosto			
	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4
Capítulo 1. Planteamiento del problema														
1.1 Introducción														
1.2 Objetivo general														
1.3 Objetivos específicos														
1.5 Alcances y limitaciones														
1.6 Cronograma de actividades														
1.7 Recursos de hardware y software														
Capítulo 2. Marco Teórico														
2.1 Redundancia y monitorización														
2.2 Control de congestión de la red														
2.3 Identificación de desconexiones en la red														
2.4 Protocolo y mecanismo Tcp														
2.5 Protocolo Tcp friendly														
2.5.1 Tct-friendly para Unicast y Multicast														
2.5.2 Windows-based														
2.5.3 Rate-based														
2.6 Modelo Tfrcp														
2.7 Protocolo Tfrc														
Capítulo 3. Metodología														
3.1 Entendimiento de los conceptos disponibilidad y alta disponibilidad en sistemas														
3.2 Explicación de las características que conforman la arquitectura propuesta.														
3.3 Explicación de las herramientas utilizadas en la arquitectura.														
3.4 Selección, estudio y explicación del protocolo TCP Friendly Rate Control.														
3.5 Estudio y adaptación de las clases con las que cuenta el protocolo Tfrc programadas en Java.														
3.6 Diseño de la arquitectura distribuida.														
3.6.1 Capa de identificación de desconexiones "IDR".														
3.6.2. Capa de control de congestión "CCR".														

Tabla 1.6.A Cronograma de actividades de Proyecto de Investigación 2.

Actividad	Septiembre				Octubre				Noviembre				Diciembre	
	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2	Semana 3	Semana 4	Semana 1	Semana 2
Capítulo 4. Implementación														
4.1 Diseño y explicación del diagrama de clases														
4.2 Especificación de parámetros a considerar de la arquitectura														
4.3 Especificacion de los sistemas operativos utilizados														
4.4 Obtención de la dirección IP del equipo Servidor.														
4.5 Modificación de clases AudioAppClient.java y AudioAppServer.java														
4.6 Compilación de todas las clases.														
4.7 Ejecución de la aplicación.														
4.8 Aplicación de tráfico dentro del grupo de trabajo.														
Capítulo 5. Resultados														
5.1 Creación e interpretación de gráficas obtenidas.														
Capítulo 6. Concluciones														
6.1 Características de la arquitectura propuesta														
6.1 Ventajas que contiene la arquitectura distribuida														
6.1 Desventajas encontradas en la arquitectura distribuida														

1.7 Recursos de hardware y software

Para poder lograr el diseño de la arquitectura distribuida se necesitarán los siguientes recursos:

Cantidad	Hardware
6	Equipos de cómputo
6	Cables Utp Categoría 5
12	Conectores Rj45
1	Switchs marca 3com con 8 puertos

Cantidad	Software
1	Sistema Operativo Windows XP profesional y Windows 7 Ultimate.
1	IDE para la codificación de las clases del protocolo TCP-friendly-TFRC en el lenguaje Java sobre Netbeans.

Características de los equipos de cómputo

Dispositivo	Característica
PC Escritorio	
Disco Duro	250 Gb
Memoria	2 Gb
Procesador	Amd Atlon 1.66 Ghz
Sistema Operativo	Windows XP Professional Service pack 1

Capítulo 2. Marco teórico

Este capítulo menciona los elementos principales que requiere una aplicación para garantizar la alta disponibilidad de la información, éstos son: redundancia y monitoreo. Dentro del elemento de monitorización se encuentran cuatro problemas fundamentales en la red: congestión, retardos del envío de paquetes y ruteo, así como número de desconexiones en los enlaces.

Este proyecto se centra en los problemas de congestión de la red e identificación en desconexiones dentro de los enlaces, mostrando en qué consisten y sus características principales. Para atacar estos problemas, se presenta una solución basada en el uso de un protocolo de comunicación llamado Tcp, se explican sus características principales, funcionamiento y desventajas. Luego se propone el uso de un protocolo de comunicación llamado Tcp-friendly, mencionando sus características y ventajas en comparación al protocolo Tcp.

2.1 Redundancia y monitoreo

Para los principales problemas que se presentan cuando se quiere garantizar la alta disponibilidad de la información en un entorno distribuido, se ha optado por implementar diversas soluciones que tienen como base fundamental dos conceptos principales: *redundancia* y *monitoreo* [Neira 2007].

La *redundancia* es establecer una o múltiples réplicas de información almacenadas en dispositivos de almacenamiento que se encuentran listas para iniciar un mismo servicio. En caso de presentar un fallo en el instante de llevar a cabo la réplica, entra en funcionamiento un algoritmo de selección de réplica segura, considerando una réplica segura cuando su estado de funcionamiento se encuentra en ejecución, esto se conoce como "*agreement protocol"*. Este protocolo selecciona un reemplazo que pueda garantizar la continuidad del funcionamiento de forma correcta. Algunos ejemplos de réplicas de información son los sistemas "Raid", que consisten en utilizar un arreglo lógico de disco, en donde un grupo de discos almacena la información redundante que permite que en caso de que algún disco falle, la información pueda regenerarse a su último estado de forma automática. Existen diferentes niveles de arreglos, algunos de ellos son Raid 0, Raid 1, Raid 2, Raid 3 y Raid 0+1 [Tanenbaum 2003].

El *monitoreo* se puede definir como una colección de mecanismos de monitoreo de estados del sistema de réplicas y de la aplicación para garantizar el adecuado funcionamiento de la solución. Así mismo, existe un software que se encarga de realizar una función de monitorización de las réplicas que comprueba el estado adecuado del proceso. Al llevar a cabo el sistema de monitoreo, se encuentran los siguientes problemas: el monitoreo del estado del sistema y monitoreo del estado de las réplicas.

Cuando se lleva a cabo el monitoreo de estado del sistema, existen cuatro problemas [Chandra 1996]:

- **Congestión dentro de la red**
- **Desconexiones dentro de los enlaces de red**
- Retardos del envío de paquetes dentro de una red
- Ruteo de los paquetes

2.2 Control de congestión de la red

Cuando la aplicación se encuentra en funcionamiento, existen demasiados paquetes que están dentro de una parte de la subred donde el rendimiento se reduce. Esta situación se le llama *congestión de la red* [Tanenbaum 2003].

La congestión puede ser causada por varios factores. Un ejemplo es cuando una serie de paquetes llegan y requieren una línea de entrada, todos necesitan la misma línea de salida, se crea entonces una cola de espera. Si no hay suficiente memoria, los paquetes se pierden. Añadiendo más memoria puede ayudar hasta cierto punto, pero [Nagle 1987] reporta que si los ruteadores tienen una cantidad de memoria alta, la congestión empeora. La razón es que al momento de llegar los paquetes a la parte frontal de la cola, han agotado el tiempo de espera en repetidas ocasiones duplicando y enviando paquetes, los cuales serán transmitidos al siguiente ruteador. Esto ocasiona aumento de la carga hasta su destino.

Un problema para los protocolos de transporte es el control de la congestión. Anteriormente la congestión sólo se detectaba cuando un paquete no era reconocido y tenía que ser retransmitido. Tratar de identificar los paquetes perdidos a consecuencia de la congestión, resultaba difícil, ya que eran confundidos con los paquetes destruidos. La única solución a una red congestionada era disminuir los paquetes transmitidos [Tanenbaum 2003].

Hay dos razones por la cual una red se congestiona: "la capacidad de red y la capacidad del receptor para la recepción de paquetes". Si el emisor está transmitiendo paquetes mientras que la red se encuentra congestionada, esta situación sólo incrementa el retraso y la congestión de la red [Tanenbaum 2003].

La capacidad del receptor está determinada por la rapidez con la que lee los datos que ha recibido. No importa qué tan congestionada esta la red, si la capacidad del receptor es baja, el paquete se demorará en ser leído y reconocido. Por lo tanto, hay dos tamaños de ventana que se mantienen: el receptor y la congestión. El mínimo de los dos tamaños de ventana es el número de bytes que el remitente puede transmitir [Tanenbaum 2003].

TCP necesita controlar el tamaño de la ventana cuando se detecta la congestión dentro de un enlace de la red. TCP utiliza un algoritmo conocido como comienzo lento (“slow start”). Este algoritmo comienza con una pequeña ventana, si se reconoce la congestión antes de que el tiempo, se duplica el tamaño de la ventana. Este proceso se repite hasta el tamaño de la ventana del receptor ha alcanzado el máximo tamaño posible de la ventana de congestión. "Cuando la ventana de congestión está en *n* segmentos y si todos los *n* son reconocidos a tiempo, la ventana de congestión incrementa el número de bytes correspondiente a los *n* segmentos, dando como consecuencia la duplica la ventana de congestión [Tanenbaum 2003]”.

- Estrategia de TCP
  - Controlar congestión una vez que ésta ocurre
  - Repetidamente incrementar la carga en un intento de encontrar el punto al cual la congestión ocurre y luego retroceder
- Estrategia alternativa
  - Predecir cuando la congestión está a punto de ocurrir
  - Reducir la tasa antes de que empiecen a descartarse paquetes

En este proyecto, los protocolos implementados para llevar a cabo el control de congestión del ancho de banda dentro de la red son*: Tcp* y *Tcp-friendly*. Estos protocolos se analizarán y explicarán en las siguientes secciones.

2.3 Identificación de desconexiones en la red

Cuando se lleva a cabo una detección de desconexiones dentro de una red, se pueden utilizar los protocolos Tcp o Tcp-friendly. Para esta investigación se implementa el uso del protocolo Tcp-friendly. Este protocolo utiliza seis paquetes para su implementación: Req, Rep, Ack, Aya, Iaa, Ta y Au [Tanenbaum 1996]. Los principales paquetes que monitorean el estado del servicio son: Aya y Iaa. La razón por la cual se seleccionaron estos dos paquetes es con base a su función, ya que determina el reconocimiento del estado del sistema, es decir, si el estado del servicio está activo o caído.

Cuando se utilizan los códigos Aya y Iaa, se especifica que cada uno contiene un paquete de configuración individual que realizará una tarea diferente. El mecanismo de trabajo para los códigos de reconocimiento para las desconexiones dentro de la red se basa en dos fases:

Primera fase.- El código Aya se encuentra en el cliente, éste pregunta al servidor el estado del sistema con la interrogante ¿estás vivo? , y verifica si el servicio sigue disponible.
Segunda fase.- Al recibir la pregunta, el servidor responde con el código Iaa indicando que el estado del servicio se ha caído.

2.4 Protocolos UDP y Tcp.

A continuación se presentan los protocolos UDP y Tcp, mostrando su definición, principales características y cabeceras.

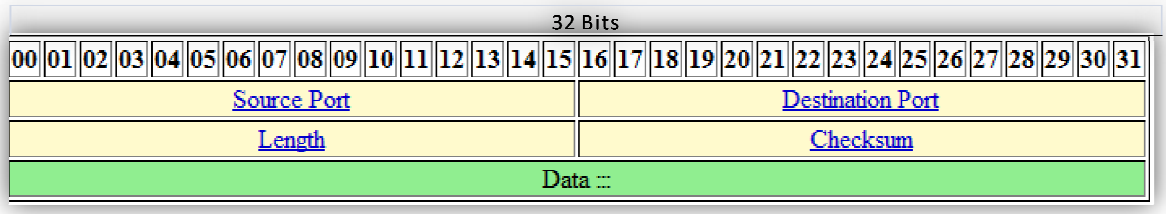
UDP

El protocolo UDP (User Datagram Protocol) protocolo de datagramas para el usuario, es un protocolo que se basa en el intercambio de paquetes en forma de datagramas a través de la red [Postel 1980].

Características:

- Sin conexión
- Flujo de información Unicast
- Poco fiable
- Utiliza mensajes de transmisión (datagramas de usuario)
- Tolerancia a fallas
- Envío de datos con una tasa constante
- No reduce la ventana de transmisión de datos
- No proporciona software para verificar la entrega de los mensajes
- No re ensambla los mensajes entrantes
- No utiliza acuses de recibo
- No proporciona control de flujo
- No ofrece garantías de retardo para los datos
- No brinda un respaldo en el ancho de banda
- No contiene capas dedicadas para garantizar la alta disponibilidad

Segmentos de cabecera TCP.



- **Source Port:** Número del puerto de envío, 16 bits
- **Destination Port:** Número del puerto donde es enviado el paquete, 16 bits
- **Length:** 16 bits
- **Checksum:** 16 bits

Un protocolo irresponsable como UDP puede dañar el funcionamiento de internet al no colaborar con el resto de tráfico para evitar y reducir las congestiones. Y no sólo esto, sino que fácilmente puede inhibir otras conexiones TCP, ya que al luchar por el ancho de banda disponible, TCP disminuirá su carga de trabajo en cuanto note alguna pérdida de paquete, mientras que UDP seguirá intentando enviar al máximo sin importar garantizar la fidelidad de la entrega de paquetes.

TCP

El protocolo TCP (Transmission Control Protocol) es uno de los más utilizados en internet hoy en día. TCP es un protocolo de conexión confiable. Este protocolo fue creado por el Departamento de Defensa para enlazar diferentes computadoras y redes teniendo como objetivo base garantizar la entrega de mensajes [Shanwei et al. 2009].

Así mismo, TCP es utilizado por los tres servicios más comunes: la transferencia de archivos, correo electrónico y conexión para accesos remotos [Tanenbaum 2003]. Este protocolo proporciona información sobre la entrega de paquetes al remitente mediante el envío de números de secuencia y acknowledgments (aceptaciones de paquetes), si los datos se pierden o llegan tarde, se retransmiten hasta que se reciban [Allman y Falk 2009].

Igualmente, TCP es un protocolo de control de la transmisión unicast. No implementa la multidifusión o difusión, utiliza conexiones de punto a punto, donde sólo hay un servidor y un cliente con una sola conexión entre ellos. TCP también es Full Duplex, esto implica que los datos pueden ser enviados desde el servidor al cliente y viceversa utilizando la misma conexión simultáneamente [Tanenbaum 2003].

Recientemente, el uso de audio y vídeo al utilizar internet ha aumentado, pero la fiabilidad, el control de la congestión y las retransmisiones son algunas de las razones por las que TCP no sea apropiado para la transmisión de aplicaciones multimedia.

Características:

- Orientado a la conexión
- Fiable
- Flujo de información Unicast
- Divide los mensajes salientes en segmentos
- Re-ensambla los mensajes en la estación destino a partir de los segmentos entrantes
- Re transmisión de todos los paquetes que no se recibieron
- Control de flujo
- Establecimiento de conexión
- No es adecuado para envío de información en vivo
- Usa control de congestión extremo a extremo

Tabla 2.4 Segmento de cabecera TCP

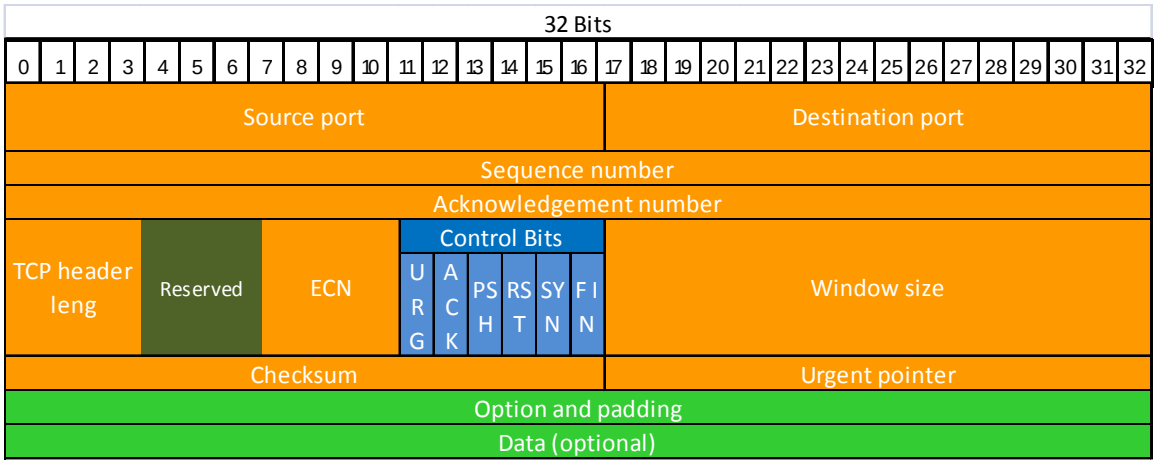


Tabla recolectada de [Tanenbaum 2003].

Donde:

- Source Port (Puerto de origen): 16 bits
- Destination Port (Puerto destino): 16 bits
- Sequence Number (Numero de secuencia): 32 bits
- Acknowledgment Number (Numero de reconocimiento): 32 bits
- Data Offset (Datos perdidos): 4 bits.
- Reserved (Reservado): 3 bits.
- E.C.N.: Notificación de congestión, 3 bits.
    a. N, NS: Suma non 1 bit.
    b. C, CWR: 1 bit.
    c. E, ECE: ECN-Echo 1 bit.
- Control Bits: U,A,P,R,S,F
    a. U, URG: 1 bit, Urgent pointer valid flag.
    b. A, ACK. 1 bit, Acknowledgment number valid flag.
    c. P, PSH. 1 bit, Push flag.
    d. R, RST. 1 bit, Reset connection flag.
    e. S, SYN. 1 bit, Synchronize sequence numbers flag.
    f. F, FIN. 1 bit, End of data flag.
- Window size (Tamaño de ventana): 16 bits, no asignado.
- Checksum: 16 bits.
- Data (Datos): variable de longitud.

2.5 Protocolo Tcp friendly

El protocolo Tcp-friendly es una variante mejorada del protocolo Tcp, presenta una mejora en cuestión de altos flujos de información, tiene mecanismos de control de congestión dentro de la red. Este protocolo comparte el ancho de banda disponible con aplicaciones construidas sobre el protocolo TCP tales como navegadores web, FTP o clientes de correo electrónico.

Así mismo, el protocolo Tcp-friendly mejora la calidad de los protocolos al momento de trabajar con flujos de información, otorgando una mejor forma en comparación con el protocolo TCP para la detección de pérdida de paquetes. Esta mejora se logra porque el receptor supervisa continuamente la disponibilidad, recepción y tasa de pérdida de datos, ajustando el tamaño de la ventana de acuerdo al nivel de congestión.

Este protocolo funciona con flujos de información unicast y multicast, tiene dos esquemas principales de acuerdo a cada tipo de flujo anteriormente mencionado:

- *Single-rate (tasa de envío simple)*: Desempeño de ráfagas simples de datos transferidos dentro de la red. Dentro de este esquema, existe la sub clasificación llamada Unicast (unidifusión).

- *Multi-rate (tasa de envío multiple)*: Desempeño de ráfagas múltiples de datos transferidos dentro de la red. En este esquema se encuentra una sub clasificación llamada Multicast (multidifusión).

De manera gráfica, se puede consultar la **Figura 2.5.A** para tener un panorama más claro acerca de los esquemas en los que se divide el protocolo Tcp friendly.

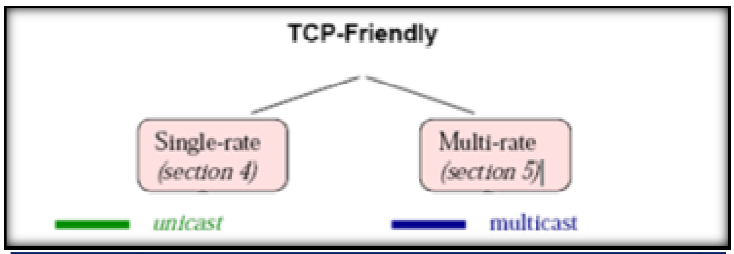


Figura 2.5.A Flujos de información TCP-Friendly

Características **Figura 2.5.A** Flujos de información TCP-Friendly:

- Orientado a la conexión y desconexión.
- Fiable.
- Flujo de información Multicast y Unicast.
- Divide los mensajes salientes en segmentos.
- Re-ensambla los mensajes en la estación destino a partir de los segmentos entrantes.
- Re transmisión de todos los paquetes que no se recibieron.
- Control de flujo.
- Establecimiento de conexión.
- Adecuado para envío de información en vivo.
- Usa control de congestión.
- Menos sensible a pérdida de paquetes.
- Gestión de tasa de envío.
- Reacciona de forma sutil ante pérdida de paquetes.

Figura 2.5.B Segmentos de cabecera TCP.

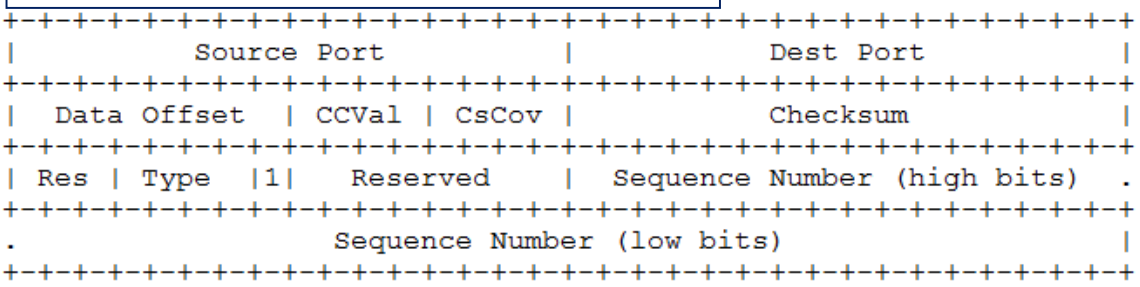


Imagen tomada de [Floy 2006].

Donde:

- **CCval:** Espacio donde se almacenan las rondas de tiempo.
- **CsCov:** Ayuda a la suma de los Checksum.
- **Numero de secuencia altos bits (high bits):** Secuencia de paquetes recibidos.
- **Numero de secuencia bajos bits (low bits):** Secuencia de paquetes perdidos.

A continuación se explican las categorías por las que se encuentra conformada los flujos de información para el protocolo Tcp-friendly.

2.5.1 TCP-friendly para Unicast y Multicast

El flujo de información Unicast (unidifusión) a largo plazo, no reduce el rendimiento de procesamiento de cualquier flujo de información coexistente con el protocolo Tcp. El flujo de información es hacia una misma trayectoria dentro de la red.

El flujo de información Multicast (multidifusión): en esta categoría el flujo de la información es punto-multipunto, la información que está compuesta por un remitente y un receptor, donde el emisor envía la información a un grupo de host distintos.

Este proyecto se enfoca en el análisis y explicación de la forma de trabajo del esquema de flujos de información de Single-rate, tomando un enfoque basado en el flujo de información Unicast, *ver la* ***Figura 2.5.1.A***.

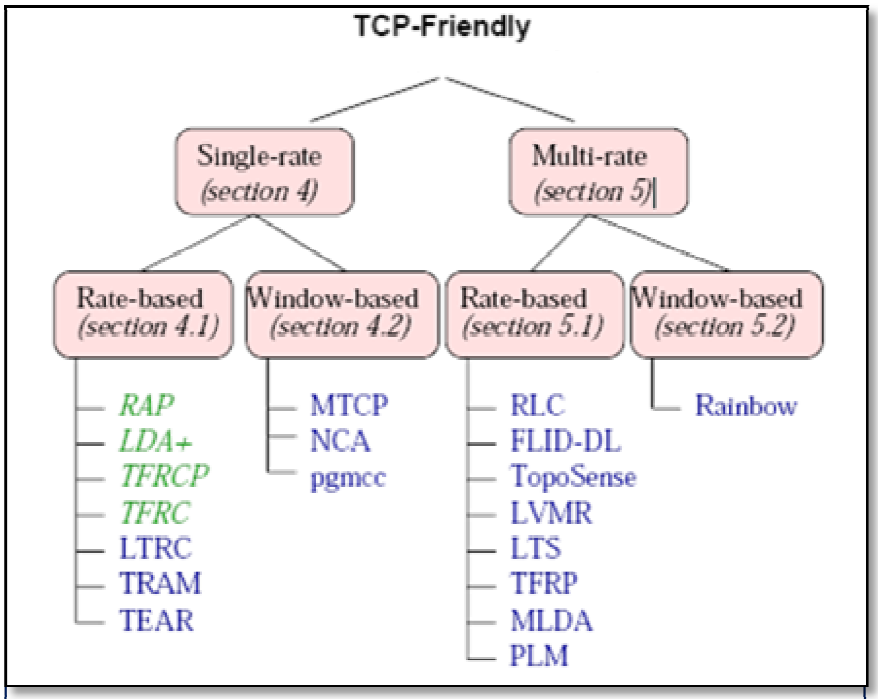


Figura 2.5.1.A Flujos de información TCP-Friendly versión extendida.

Dentro de la clasificación de Single-rate, existen dos formas de trabajar para el protocolo Tcp-friendly: Rate-based (basado en tasa de envío) y Windows-based (basado en ventana); dependiendo de si adaptan su carga de trabajo de red ofrecida basada en una ventana de congestión (Windows-based) o en una tasa de transmisión (Rate-based) [Widmer et al. 2001]. Estas formas de trabajo se describen en las secciones siguientes.

2.5.2 Windows-based

En esta forma de trabajo se utiliza una ventana de congestión en el remitente o en los receptores para asegurar el correcto funcionamiento del protocolo Tcp-friendly. Cada paquete transmitido consume una ranura de la ventana de congestión, mientras que cada paquete recibido o el reconocimiento de un paquete recibido liberan una ranura. El remitente transmite los paquetes solamente cuando una ranura libre está disponible. El tamaño de la ventana de congestión se incrementa al verificar ausencia de congestión.

2.5.3 Rate-based

La forma de trabajo "Rate-based" es dinámica, contiene un acuerdo de la tasa de envío de transmisión a un cierto mecanismo de renovación de la red que indica congestión. Este control puede producir cambios mucho más rápidos de la tasa de envío para su mejor desempeño, adaptándose al tipo de tráfico presentado en la congestión.

Los principales modelos para esta forma de trabajo son:

1. TFRCP (protocolo de control de tasa de envío basado en Tcp-friendly).
2. TFRC (protocolo de control de tasa de envío).
3. RAP (protocolo de adaptación de la tasa de envío).
4. LAD+ (algoritmo de pérdida-retardo).
5. LTRC (control de la tasa de envío tolerante a pérdidas).
6. TRAM (árbol basado en protocolo de fiabilidad multidifusión).
7. TEAR (emulación de receptores basado en protocolo Tcp).

En las secciones siguientes se explican únicamente los algoritmos TFRCP y TFRC.

2.6 Modelo Tfrcp

El modelo Tcp-friendly está basado en el modelo descrito en [Padhye et al. 1999], el cual ajusta su tasa de envío usando el modo complejo de TCP. Es decir, su modo de trabajo se basa en dividir el tiempo en rondas de trabajo con una duración fija y el modelo de parámetros son calculados para cada ronda de envío. La tasa de envío de paquetes se duplica en la siguiente ronda al no existir pérdida de paquetes.

En Tfrc, cada paquete de datos es reconocido por el receptor. El protocolo confía fuertemente en la duración del intervalo de re-computarizado de los paquetes, puesto que este parámetro es estático. El modelo no se adapta bien a las condiciones de la heterogeneidad de la red, dando como consecuencia el duplicado de la tasa de envío. En comparación con el protocolo Tcp, éste modelo contiene mecanismos de control de congestión diseñado para la operación de flujos de información de tipo Unicast en un entorno de internet que compite con el tráfico del protocolo Tcp.

2.7 Protocolo Tfrc

El protocolo del control de la tasa de envío de Tcp-friendly (Tfrc) fue desarrollado a partir del modelo Tfrcp. Este protocolo se especifica para el flujo de información unicast, aunque con algunas modificaciones puede ser adaptado al flujo de información de tipo multicast. Similar a Tcp-friendly, la forma de trabajo de Tfrcp ajusta su tasa de envío utilizando métodos más sofisticados para recolectar los parámetros necesarios para su correcto funcionamiento. Para medir la tasa de envío en cuestión de pérdida de paquetes, se lleva a cabo con el uso de intervalos del número total de pérdidas de paquetes, creando intervalos de la pérdida de los paquetes.

Capítulo 3. Metodología

La metodología para ofrecer una alta disponibilidad dentro de un sistema distribuido está compuesta por varios puntos que contemplan la definición de disponibilidad y alta disponibilidad, éstos son:

3.1.- Entendimiento de los conceptos disponibilidad y alta disponibilidad en sistemas distribuidos.

a) Explicación de los problemas de la alta disponibilidad.
    a. *Desconexiones de la red*: Es el instante en el que se interrumpe una conexión entre el cliente y el servidor, dando como consecuencia el impedimento de envío de paquetes entre ellos.

    b. *Congestión de tráfico de la red*: Cuando la aplicación se encuentra en funcionamiento, existen demasiados paquetes que están dentro de una parte de la subred donde el rendimiento se reduce. [Tanenbaum 2003].

b) Análisis de las soluciones para garantizar la alta disponibilidad en sistemas distribuidos.
    a. *Redundancia de información*: Establecer una o múltiples réplicas de información almacenadas en dispositivos de almacenamiento que se encuentran listas para iniciar un mismo servicio.

    b. *Monitoreo de la red*: Es una colección de mecanismos de monitoreo de estados del sistema de réplicas y de la aplicación para garantizar el adecuado funcionamiento de la solución.

c) Comparación de los protocolos UDP, TCP y Tcp-friendly:

d) Selección del protocolo Tcp-friendly para implementarlo en la arquitectura.

e) Explicación de los diferentes esquemas trabajo para Tcp-friendly: Single-rate y Multi-rate, seleccionando en este trabajo el esquema Single-Rate en la sección 2.5.1.

    Single-rate (tasa de envío simple): Desempeño de ráfagas simples de datos transferidos dentro de la red.

    Multi-rate (tasa de envío múltiple): Desempeño de ráfagas múltiples de datos transferidos dentro de la red.

f) Selección del mecanismo del flujo de la información Multicast (multidifusión) y comparación de sus características con el mecanismo Unicast.

3.2.-Explicacion de las características que conforman la arquitectura propuesta.

- *Grupos cerrados:* Esto quiere decir que dentro de cada grupo existen miembros que se encuentran conectados, siguiendo una forma distribuida, teniendo en común los datos que sólo los miembros del grupo pueden compartir, quienes tienen la posibilidad de modificar su contenido.

- *Canales independientes para envío y recepción de información:* La forma de comunicación es por medio de canales independientes para conectar a los miembros de cada grupo cerrado.

- *No existe un coordinador central:* Dentro de cada grupo cerrado, todos los miembros son iguales con respecto a su nivel jerárquico. Cada miembro tiene la capacidad de almacenar y distribuir la información contenida de forma local o remotamente.

- *Capa IDR:* Tiene la finalidad de la identificación de desconexión dentro de un enlace de la red.

- *Capa CCR:* Al presentarse una congestión dentro del enlace, esta capa tiene como objetivo llevar a cabo un control de congestión.

3.3.-Explicación de las herramientas utilizadas en la arquitectura.

- *Windows Xp Professional y Windows 7:* Sistemas operativos utilizados como plataforma gráfica para llevar a cabo el desarrollo de la arquitectura.

- *Java:* Lenguaje de programación de alto nivel orientado a objetos.

- *NetBeans 6.7.1:* Plataforma IDE que permite el desarrollo de aplicaciones creadas bajo el lenguaje Java.

- *Wireshark:* Analizador de protocolos utilizado para capturar tramas de la red.

3.4.- Selección, estudio y explicación del protocolo TCP Friendly Rate Control.

3.5.- Estudio, explicación general e implementación de las clases con las que cuenta el protocolo Tfrc.

- *AudioAppClient.java:* Interfaz del cliente
- *AudioAppServer.java:* Interfaz del servidor
- *AckWindowList.java:* Listado de paquetes recibidos
- *ControlStream.java:* Control de flujos de información
- *DataStream.java:* Gestiona el orden de envío para los paquetes
- *TFRCCongestion.java:* Determinador de tasa de congestión
- *TFRCInputStream.java:* Servicio de lectura de los datos de entrada
- *TFRCOutputStream.java:* Servicio de lectura de los datos de salida
- *TFRCPacket.java:* Gestiona los paquetes y envío a TFRCPacketHeader
- *TFRCPacketHeader.java:* Almacena los datos dentro de un arreglo
- *TFRCServerSocket.java:* Conecta las máquinas dentro del grupo de trabajo
- *TFRCSession.java:* Gestión del número de secuencias a enviar por ACK
- *TFRCSocket.java:* Consulta estado de los puestos, direcciones, tamaño del paquete

Estas clases fueron recolectadas del trabajo [Silva et al. 2009]

3.6.- Diseño de la arquitectura distribuida

A continuación se presenta un esquema general de la arquitectura distribuida propuesta en este proyecto, contemplando un grupo de trabajo que se encuentra constituido por tres equipos de cómputo conectados, que se encuentran funcionando de forma distribuida, tal y como se muestra en la **Figura 3.6.A**.

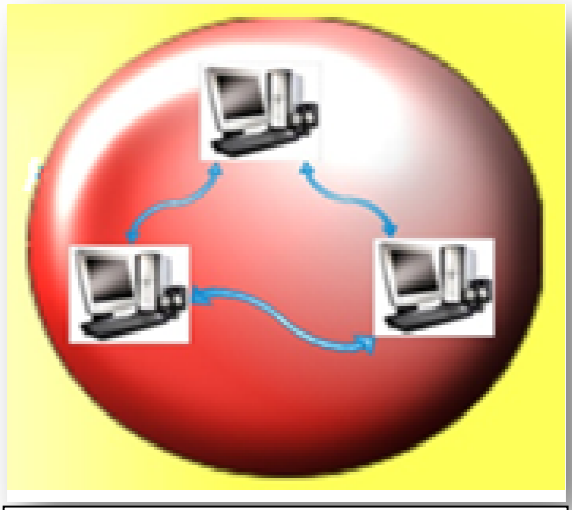


Figura 3.6.A Esquema general arquitectura distribuida RADAD.

Posteriormente la arquitectura se divide dejando únicamente a dos equipos de cómputo como se muestra en la **Figura 3.6.B** Esquema general arquitectura propuesta, donde se representa un enlace de comunicación entre los equipos A y B, que fueron tomados del grupo de trabajo de la Figura X. Ambos equipos contienen las dos capas "IDR y CCR" para afrontar y disminuir los problemas al momento de buscar la alta disponibilidad: desconexiones y congestión dentro de la red, ambas capas son soportas por el uso del protocolo "Tcp-friendly-Rate Control", en el cual se inicializa con la primera capa "IDR" y en paralelo con la segunda "CCR", teniendo en todo momento comunicación entre ellas.

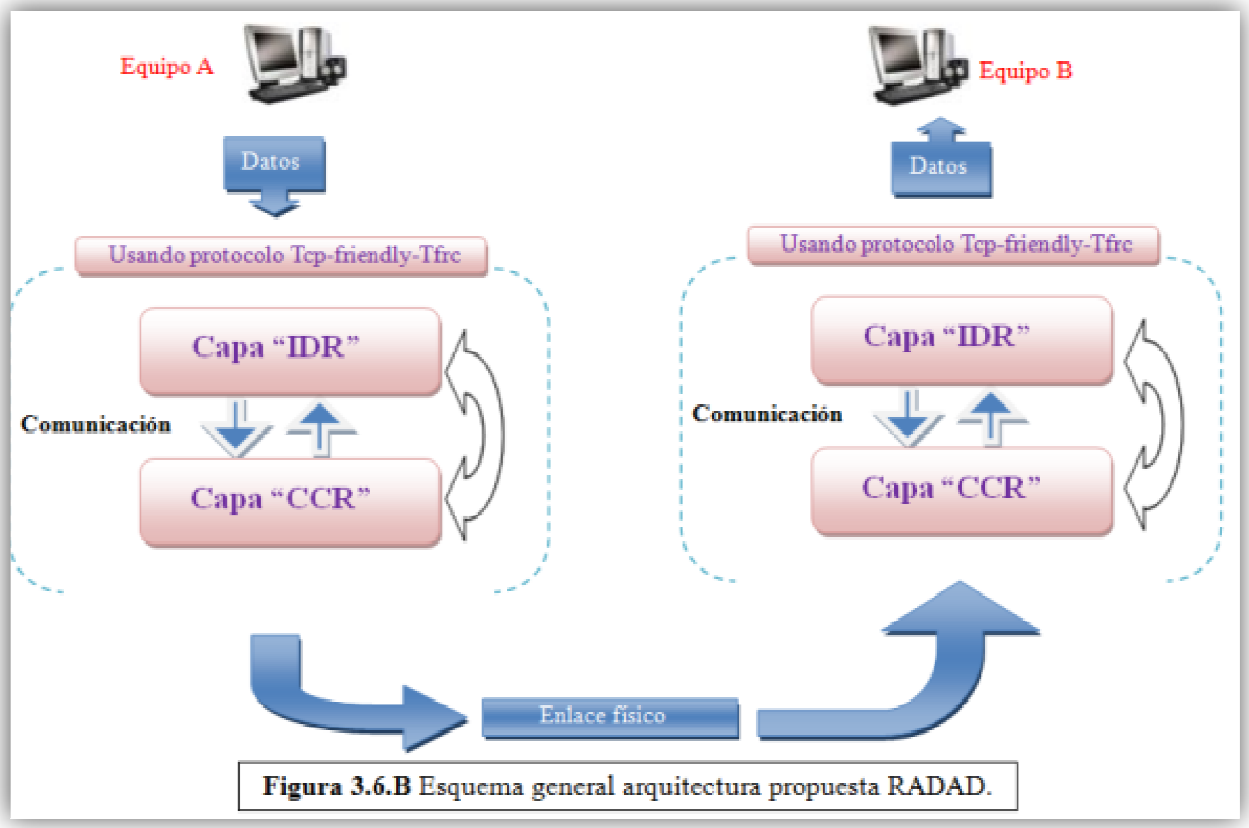


Figura 3.6.B Esquema general arquitectura propuesta RADAD.

A continuación se procede a la explicación de cada una de las capas que conforman la arquitectura distribuida.

3.6.1.- Capa de identificación de desconexiones dentro de la red "IDR"

Para llevar a cabo una Identificación de desconexiones dentro de la red, la arquitectura propuesta utiliza la capa numero uno: IDR, utilizando el protocolo Tcp-friendly-Rate Control; Este protocolo tiene un mecanismo de trabajo que consiste en: El envío de la información desde el emisor hasta el receptor e inversamente desde el receptor hasta emisor, *Ver la* ***Figura 3.6.1.***

Figura 3.6.1 Mecanismo de trabajo capa IDR

Para que pueda existir una comunicación entre un emisor y receptor, es necesario inicializar con los estados de conexión llamados *"Conectado" y "Desconectado"*, siempre comenzando con la inicialización del estado Conectado, una vez iniciado el estado de conexión *conectado* entran en funcionamiento las capas numero uno y dos: *IDR y CCR.*

Para la capa número uno, cada paquete de información que viaja desde el *emisor* al receptor contiene tres campos:

- *Campo 1 EMI.*- Indica el número de secuencia del paquete, permite al receptor advertir sobre el instante que lleguen a surgir alteraciones en el orden de llegada de los paquetes y además de advertir el momento de una desconexión.

- *Campo 2 EMI.*- Contiene una marca de tiempo mostrando el instante de tiempo de cuando fue enviado el paquete.

- *Campo 3 EMI.*- Muestra la estimación por parte del emisor del tiempo de "ida y vuelta", llamado RTT (Round-Trip Time) y tanto este campo como el numero 2, tienen la finalidad de notificar al receptor sobre los paquetes perdidos que hayan presentado dentro del enlace de la red.

Mientras que los paquetes que se envían desde el *receptor* hacia el emisor contiene cuatro paquetes:

- *Campo 1 RECEPTOR*.- Almacena una marca temporal del último paquete de datos recibido.

- *Campo 2 RECEPTOR*.- Demuestra el tiempo transcurrido entre la recepción del último paquete de datos y la generación de un mensaje: feedback.

- *Campo 3 RECEPTOR*.- Muestra la tasa de transferencia que contuvo el receptor desde la llegada de los paquetes hasta el envío del último mensaje: feedback.

- *Campo 4 RECEPTOR*.- Indica la estimación de la tasa de pérdidas de paquetes.

Además de utilizar el campo 1EMI del emisor para detectar las desconexiones dentro de los enlaces de red, la capa uno *IDR* utiliza: la Latencia, que es el tiempo que toman los paquetes para viajar desde el emisor hasta el receptor, esta genera a su vez un Jitter, que es la variación de la latencia, causada por: las variaciones en las longitudes de trayectoria de los paquetes de información, la variación de la velocidad y tiempo de procesamiento de paquetes que llegan fuera de orden y la congestión dentro de la red [Montaner 2007],cuando el numero de expiraciones de tiempo de los No Feedback es consecutiva entre cada Feedback (paquetes que confirman la llegada exitosa de los mensajes) se presentan valores pico por parte del Jitter, dando como consecuencia una desconexión dentro de un enlace de la red, cambiando a un estado de conexión: *"Desconectado"* [Floyd y Handley 2008].

3.6.2 Capa de control de congestión "CCR"

Al momento de ser inicializada, la capa "*CCR*" tiene como objetivo: el control de congestión dentro de un enlace de la red. Para lograrlo, se calcula tasa de envío, pérdidas de paquetes por parte del receptor y velocidad de transmisión; estos cálculos se realizan tanto en la parte del emisor como el transmisor.

Para establecer una tasa de envío por parte del emisor se utiliza la fórmula que se presenta en la **Figura 3.6.2.A**, esta tasa de envío se regula de dos formas:

- Cuando termina el temporizador de feedback sin recibir paquetes por parte del receptor, la tasa de envío del emisor se reduce a la mitad

- Al momento de captar mensajes del feedback, las medidas que vienen insertados dentro de los paquetes sirven para resolver una ecuación que indica la nueva tasa de envío.

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}$$

Figura 3.6.2.A Ecuación para calculo de tasa de envió.

Donde:

T: la tasa de envío.
S: tamaño del paquete.
R: tiempo de ida y vuelta promediado (RTT)
P: tasa de pérdidas que experimenta la conexión (según el receptor).
tRTO: tiempo que tarda en vencer el temporizador de retransmisión.

La misión de establecer una tasa de envío dentro de la capa *"CCR"* es:

- No almacenar de forma agresiva el ancho de banda disponible en la red como lo hacen otros protocolos de comunicación, si no aumentar lentamente la tasa de envío al notar que se ha presentado una congestión dentro del enlace de la red

- No reducir bruscamente la tasa de envío al presentarse un evento de pérdida, sino apoyarse en un historial de pérdidas que afine la variación en la productividad de TFRC.

A continuación se muestran los cálculos que se realizan dentro de la capa número dos *"CCR"* por parte del emisor y receptor:

Por parte del emisor:

- El cálculo del tamaño de los paquetes se lleva a cabo con ayuda de la clase: "TFRCPacket", donde se establece la tasa de envío a 1 paquete por segundo, cuando llegue un mensaje de feedback, donde primeramente se calcula el RTT de ese paquete, como se muestra en la **Figura 3.6.2.B.**

$$RTT = t\_llegada - t\_rdatos - t\_retraso$$

Figura 3.6.2.B Cálculo tiempo total RTT.

- Determinación de tRTO con una aproximación, como se muestra en la **Figura 3.6.2.C**:

$$t_{RTO} = 4\ R$$

Figura 3.6.2.C Determinación del tRTO.

En el caso del receptor el cálculo que se realiza es:

- Determinación de tRTO con una aproximación, como se muestra en la **Figura 3.6.2.B.**

Una vez determinada la tasa de pérdidas, se procede a la modificación del rendimiento de la velocidad de transmisión con la ecuación mostrada en la **Figura 3.6.2.D**:

$$X = \frac{s}{R*\sqrt{2*b*p/3} + (t\_RTO * (3*\sqrt{3*b*p/8}) * p * (1+32*p^2))}$$

Figura 3.6.2.D Determinación de la velocidad de transmisión.

Posteriormente, el emisor y el receptor ajustan su velocidad de transmisión para resolver la congestión dentro del enlace de la red.

A continuación se presenta el capítulo 4, donde se explica la implementación de la arquitectura propuesta.

Capítulo 4. Implementación

A continuación se presentan el diagrama de clases contenidas dentro de la arquitectura distribuida propuesta.

4.1 Diseño y explicación del diagrama de clases.

Para la creación del diagrama de clases, se utilizó la aplicación “Rational Rose”, este diagrama consta de diez clases como se pueden ver en la **Figura 4.1**.

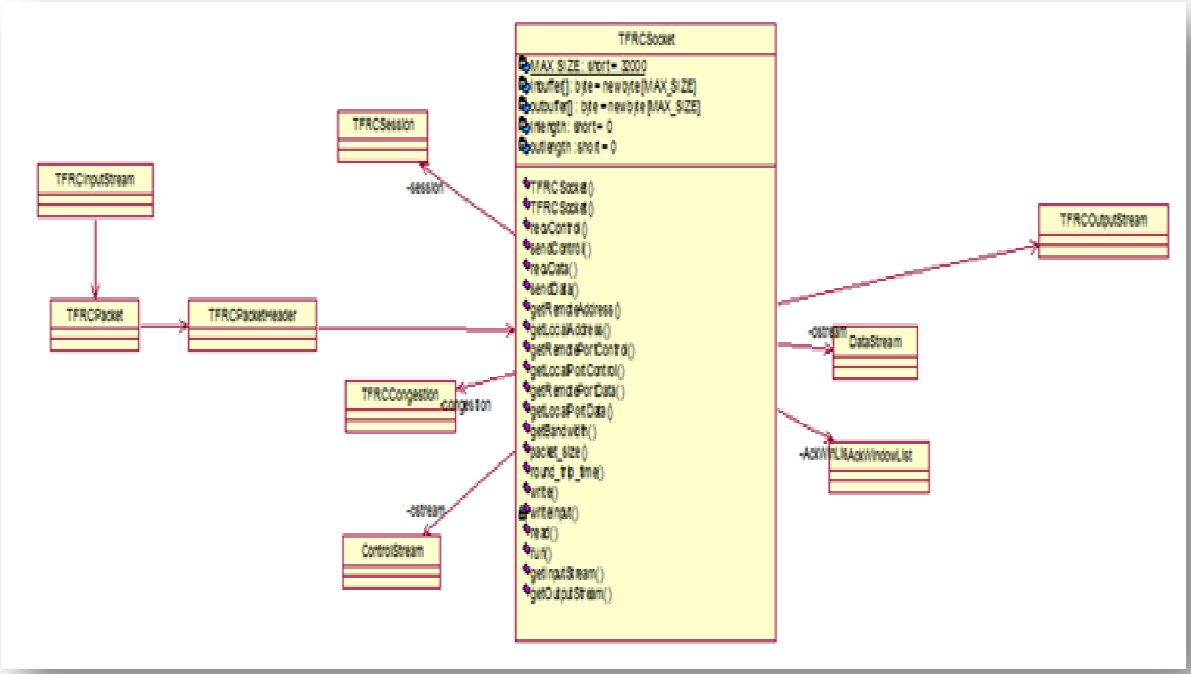


Figura 4.1 Diagrama de clases arquitectura RADAD.

El diagrama de clases **Figura 4.1** comienza con la ejecución de un TrfcSocket, esta clase tiene como objetivo el establecer los parámetros siguientes: dirección IP del emisor y receptor, puertos establecidos, agregación al grupo y equipos conectados.

La comunicación comienza y el emisor envía un paquete de información. Para esta acción se requiere de la clase TfrcInputSream, la cual tiene la función de captar los paquetes de envío, indicando a la clase TfrcPacket el tamaño del paquete que se enviará. Posteriormente, la clase TfrcPacket notifica a la clase TfrcPacketHeader para que ésta última le anexe un encabezado donde se ingresan los tres campos de identificación (Campo 1 Emisor, Campo 2 Emisor y Campo 3 Emisor). Al instante que se anexan los campos mencionados, las clases TfrcSession, TfrcCongestion, ControlStream comienzan a funcionar. La clase TfrcSession indica que se ha establecido una nueva sesión de comunicación, la clase TfrcCongestion tiene la tarea de controlar la congestión dentro del enlace de red y ControlStream maneja el control de los flujos de información hasta que el receptor reciba el paquete de información con la ayuda de la clase Receptor ControlStream. A su vez, el receptor inicializa la clase AcknowList que determina el número de paquetes aceptados. Por último, la clase TfrcOutputStream, contabiliza el número de paquetes recibidos.

4.2 Especificación de parámetros a considerar de la arquitectura.

Al momento de llevar a cabo la implementación de la arquitectura distribuida se usara las clases mostradas en la sección 3.5.

Las direcciones IP se que utilizaron para llevar a cabo la arquitectura fueron:

Servidor: 172.16.33.68
Cliente 1: 172.16.33.78
Cliente 2: 172.16.33.88
Cliente 3: 172.16.33.98

4.3 Especificación de los sistemas operativos utilizados.

Los sistemas operativos que se utilizaron como plataforma para la ejecución de las clases fueron:

Servidor: Windows 7 Ultimate
Cliente 1: Windows Xp Professional
Cliente 2: Windows Xp Professional
Cliente 3: Windows Xp Professional

4.4 Obtención de la dirección IP del equipo Servidor:

Se abre una consola dentro del sistema operativo Windows XP (combinación de teclas: Banderita de Windows + R) y se escribe el comando *cmd.exe* y presiona *ejecutar*, posteriormente en la consola se escribe el comando *ipconfig*, se presiona la tecla de enter y posteriormente la consola muestra la dirección IP del equipo servidor, tal y como se muestran en las figuras **Figura 4.4.A** y **Figura 4.4.B**.

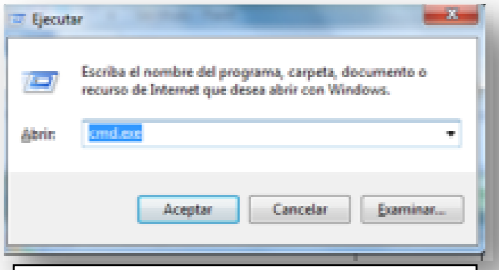


Figura 4.4.A Consola de Windows.

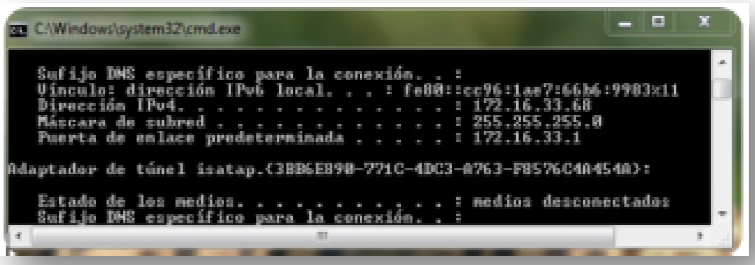


Figura 4.4.B Dirección IP en consola de Windows.

4.5 Modificación de clases AudioAppClient.java y AudioAppServer.java

AudioAppClient.java:

Se modifica la dirección IP que se encuentra dentro de esta clase por la dirección IP que tiene el servidor de la arquitectura distribuida, como se muestra en la **Figura 4.5.A.**

```
try
  {
     writer = new PrintWriter
     (new BufferedWriter
     (new FileWriter(filename)));
  }
catch (java.io.IOException e)
{
     System.err.println("Client: opening file");
     e.printStackTrace();
     System.exit(0);
}

try
  {
     sock = new TFRCSocket("172.16.33.78", 2849, 1550, 5500);
  }
catch (java.io.IOException e)
{
     System.err.println("Cliente: Conexion aceptada");
     e.printStackTrace();
     System.exit(0);
}
inStream = sock.getInputStream();
```

Figura 4.5.A Modificación dirección IP en la clase AudioAppClient.java

AudioAppServer.java:

Se agrega una condición *while* para asegurar que el tamaño del archivo enviado es mayor a 0 como se muestra en la **Figura 4.5.B**.

```
while (index < FILESIZE)
{
        byte[] tempbuf = new byte[BUFSIZE];
        byteStr = new ByteArrayOutputStream(BUFSIZE);
        dataStr = new DataOutputStream(byteStr);
        f = sock.getBandwidth() / 10;

        if (f > 8820)
          freq = 8820;
          System.out.println("Paquetes recibidos!!" );
         else
            freq = (short)f;

}
```

Figura 4.5.B Ciclo While en la clase AudioAppServer.java

4.6 Compilación de clases utilizadas:

En una consola de Windows se escribe la instrucción *javac –d . *.java* que se muestra en la **Figura 4.6** para la compilación de todas las clases de la arquitectura. Esto se debe de realizar en cada equipo de cómputo que se encuentre conectado dentro del grupo de trabajo.

Figura 4.6 Compilación de clases de la arquitectura.

4.7 Ejecución de la aplicación:

- Por parte del servidor:

  En la dirección IP *172.16.33.68* se ejecuta la clase *AudioAppServer.java*. En caso de ser exitosa la ejecución, despliega un mensaje: "Ready to accept clients", diciendo que "se encuentra listo para la aceptación de clientes", tal y como se muestra en la **Figura 4.7.A**.

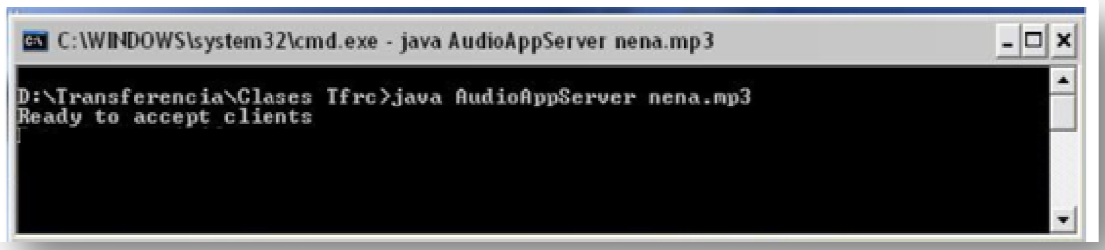


Figura 4.7.A Ejecución de la clase AudioAppServer.java

- Por parte del cliente:

  Se ejecuta la clase *AudioAppClient.java*. Al ejecutarse correctamente, se despliegan en consola los mensajes de "paquetes enviados", tal y como se muestra en la **Figura 4.7.B**.

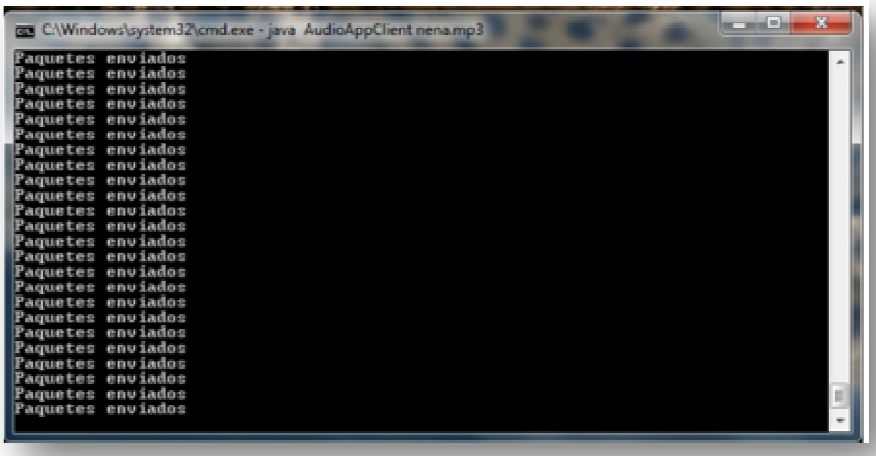


Figura 4.7.B Ejecución de la clase AudioAppClient.java

En caso de no establecer una comunicación, la aplicación mostrará un mensaje de error: "Conexión no establecida".

4.8 Aplicación de tráfico dentro del grupo de trabajo.

Para inyectar tráfico a la red cuando se ejecuta la aplicación distribuida, se ejecutó la aplicación *Wireshark.* Esta aplicación previamente se instaló y configuró dentro del equipo personal de trabajo. La ejecución de la aplicación e inyección del tráfico se representan en las Figuras **4.8.A y 4.8.B.**

Al inyectar tráfico a la red, se utilizaron los siguientes parámetros y configuraciones en cuestión de la dirección IP:

Servidor: 172.16.33.68
Cliente 1: 172.16.33.78

El tiempo total de prueba que se llevó a cabo la inyección de tráfico fue de 5 minutos.
El tamaño de los paquetes enviados por la red fue de 5Mb.

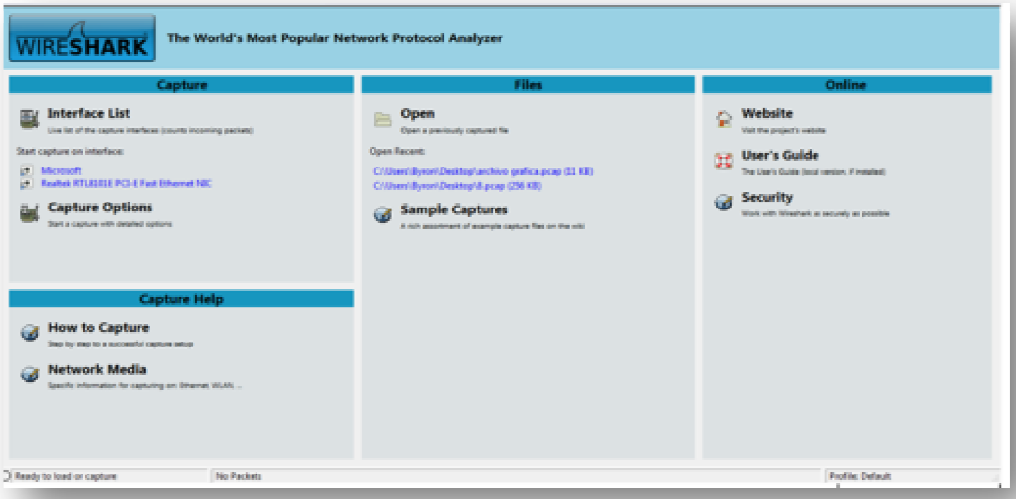


Figura 4.8.A Software Wireshark para inyección de tráfico dentro de la red.

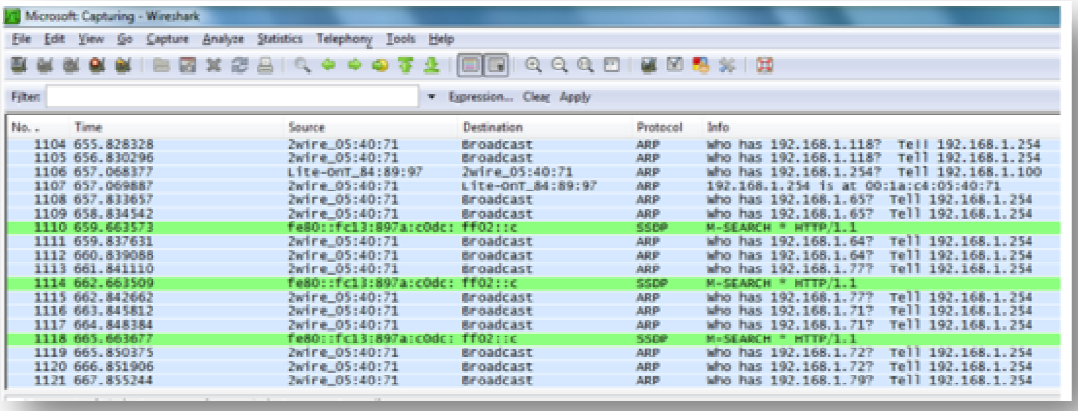


Figura 4.8.B Inyección de tráfico dentro de la red.

Capítulo 5. Resultados

5.1 Creación e interpretación de gráficas obtenidas.

Al momento de la aplicación de tráfico de la red y utilizar la arquitectura distribuida se utilizaron los protocolos TCP y TCP-friendly-Rate Control, en los cuales la interpretación es la siguiente:

Usando el protocolo TCP, se presenta en las figuras **5.1.A y 5.1.B**:

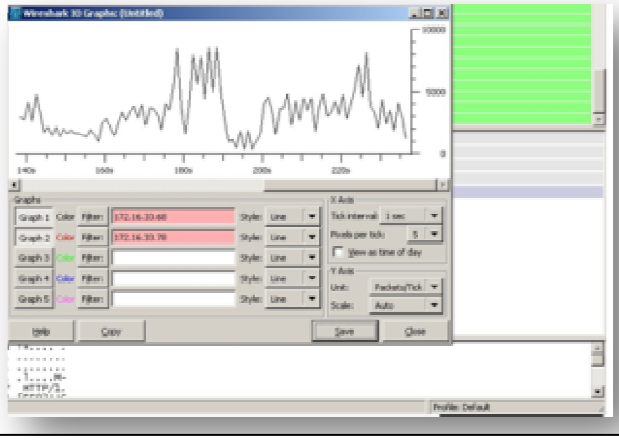


Figura 5.1.A Intervalos consecutivos que denotan una congestión.

En la **Figura 5.1.A** se observa un número de intervalos consecutivos de picos, esta figura muestra una congestión dentro del enlace de la red.

En la **Figura 5.1.B** se aprecian dos indicadores de caída por parte de la señal, mostrando que existieron desconexiones dentro del enlace de la red.

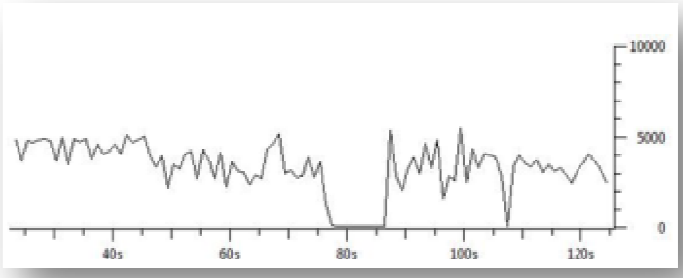


Figura 5.1.B Desconexiones dentro de la red.

.

Usando el protocolo TCP-friendly-Rate-Control, se presenta en las figuras **5.1.C y 5.1.D**:

La **Figura 5.1.C** muestra una comunicación constante y estable sin la presencia de picos que determinen una congestión dentro del enlace de la red.

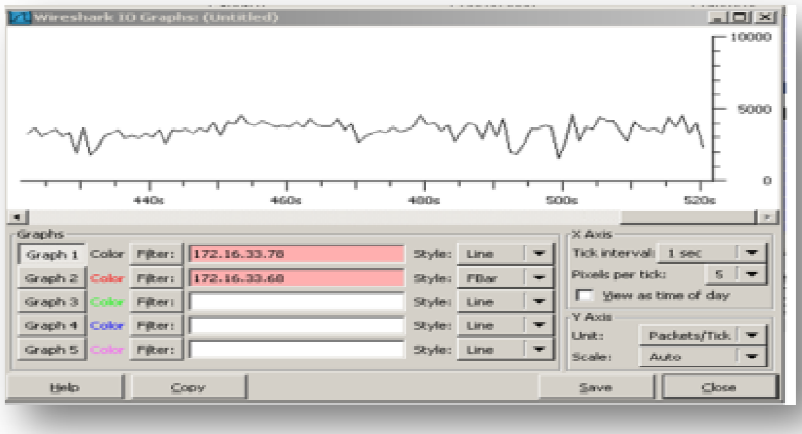


Figura 5.1.C Comunicación con TCP-friendly-Rate-Control sin congestión en la red.

La **Figura 5.1.D** muestra que no existen indicadores de caída por parte de la señal, estableciendo que no hay desconexiones dentro del enlace de la red.

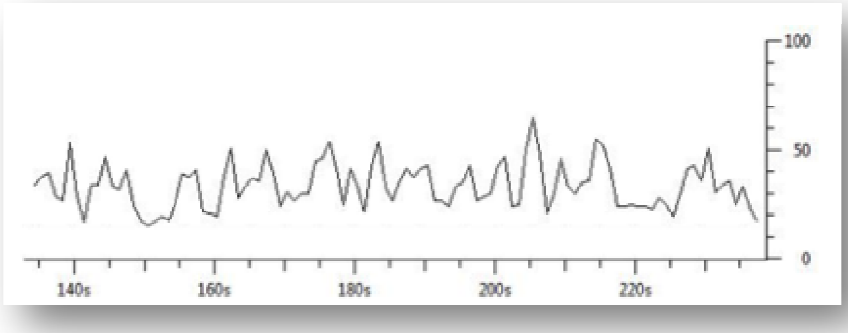


Figura 5.1.D Comunicación con TCP-friendly-Rate-Control sin desconexiones dentro de la red.

Capítulo 6. Conclusiones

Al momento de la creación de una comunicación entre las computadoras conectadas dentro de un grupo de trabajo, se presentan principalmente dos problemas "desconexiones y congestiones dentro del enlace de la red", dando como consecuencia que no se garantice alta disponibilidad. Al implementar la arquitectura propuesta en este proyecto, se pretende garantizar la alta disponibilidad de la información, teniendo como estructura el uso del protocolo TCP-friendly, que lleva a cabo la implementación de un mecanismo de trabajo llamado TFRC, este mecanismo resuelve los dos problemas al momento de garantizar la alta disponibilidad de la siguiente forma:

- Para el problema de desconexiones dentro del enlace de la red, la arquitectura utiliza la capa de *"IDR"*, donde a cada paquete de información que es enviado por el emisor se le agregan 3 campos para detectar las desconexiones y pérdidas de información que se presenten dentro del enlace de red.

- Para el problema de la congestión dentro del enlace de red, la arquitectura usa la capa *"CCR"* en donde se determina:

    - *La tasa de transferencia de información*, dando como ventaja la liberación del ancho de banda dentro del enlace de la red en comparación con otros protocolos de comunicación que acaparan todo el ancho de banda

    - *Tamaño de la ventana:* se ajusta la ventana de transferencia de forma gradual al presentarse pérdidas de datos dentro del enlace de la red, mientras que los demás protocolos de comunicación ajustan el tamaño de forma radical, limitando el desempeño de la comunicación

6.1 Características de la arquitectura propuesta

Al implementar la arquitectura propuesta dentro de un ambiente distribuido, utilizando el protocolo "Tcp-friendly-Rate Control", se obtienen las siguientes características:

- Menos sensible a pérdidas de paquetes
- Identificación de desconexiones
- Control de la congestión
- Tolerancia a fallos
- Orientado a conexión
- Trabaja con flujos Multicast & Unicast
- Los cálculos de tasa de pérdida de paquetes recaen sobre el receptor
- Limita la tasa de envío según ciertos parámetros de la red
- No reduce drásticamente la tasa de envío al encontrar un evento de pérdida
- No acapara de forma agresiva el ancho de banda disponible
- Controla la tasa de envío de un flujo de información según la congestión de la red

6.2 Ventajas que contiene la arquitectura distribuida:

- Se disminuyen los problemas al intentar garantizar la alta disponibilidad
- Se obtiene un mejor uso del ancho de banda
- Se reduce el tráfico generado por el tipo de comunicación
- Se disminuyen los problemas de tráfico

6.3 Desventajas encontradas en la arquitectura distribuida:

- No se garantiza la alta disponibilidad ante fallos físicos
- No se cuenta con mecanismos de recuperación

La arquitectura distribuida propuesta en este documento tendrá continuidad de trabajo, algunas modificaciones a futuro serán: migración y configuración a una plataforma Linux Debian , implementando software de monitoreo de red, inyección de tráfico de alto nivel llamado Ns-2 , además del comienzo de replicación de información bajo el funcionamiento de un arreglo de discos duros tipo RAID 1.

Referencias

[Allman y Falk 1999]
Allman M. y Falk A. .1999. ”On the effective evaluation of TCP”,ACM SIGCOMM Computer Communication Review archive, Volumen 29. ISSN:0146-4833

[Chandra 1996]
Chandra T.,Toueg S.. *“Unreliable failure detectors for reliable distributed systems.”* In *ACM journal*, pages 225–267, mar 1996.

[Coulouris 2001]
Coulouris, G. 2001.*”Distributed Systems:Concepts and Design”* Pearson. 3ra Edición.

[Farley 1997]
Farley M., Hsu J., Stearns T.1997. *“Guía LAN Times de seguridad e integridad de datos”*.1ra edición. McGraw-Hill. España.

[Floyd y Handley 2008]
Floyd S. y Handley M. 2008. ”TCP Friendly Rate Control (TFRC): Protocol Specification”, University College London.

[Floy y Jacobson 2008]
Floyd S., Jacobson V. 1993.*”Random Early Detection Gateways for Congestion Avoidance”*. *IEEE/ACM Transactions on Networking (TON)* 1, Agosto, 397-413

[Floy 2006]
Floyd S. 2006. ”TCP-Friendly Specification”.Network Working Group. ICIR. Fecha de acceso: 4 Noviembre 2009. Disponible en: http://tools.ietf.org/html/rfc4342

[Gray 1985]
Gray J. 1985. *“Why Do Computers Stop and What Can We Do About It”. LABShp. Reporte Técnico 85.7.* Fecha de acceso: 24 Junio 2009
Disponible en: http://www.hpl.hp.com/techreports/tandem/TR-85.7.pdf

[Kshemkalyani y Singhal 2008]
Kshemkalyani A. D. y Singhal M..2008, *“Distributed computing: principles, algorithms, and systems”.* Cambridge University Press.

[Montaner 2007]
Montaner H. 2007. “Estudio de Algoritmos de Control de Congestión TCP-Friendly”. Máster de Ingeniería de Computadores. Universodad Politécnica de Valencia.

[Nagle 1987]
Nagle, J.: *"On Packet Switches with Infinite Storage,"* IEEE Trans. on Commun., Vol. COM-35, 435-438, Abril 1987.

[Neira 2007]
Neira P. *"Tolerancia a fallos para la alta disponibilidad de software de red de alto rendimiento con estados"2007.* Departamento de lenguajes y istemas informáticos. Fecha de acceso: 17 Junio 2009. Disponible en: http://www.lsi.us.es/docs/doctorado/memorias/Neira,%20Pablo.pdf

[Padhye et al. 1999]
Padhye J., Kurose D. y Towsley R. 1999. *"A model based TCPfriendly rate control protocol"*. Proc. International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV).

[Postel 1980]
Postel J. 1980. "User Datagram Protocol". ISI. Fecha de acceso: 1 Noviembre 2009. Disponible en: http://www.ietf.org/rfc/rfc0768.txt

[Shanwei et al. 2009]
Shanwei C., Calton P., Jonathan W. "Flow and Congestion Control for Internet Media Streaming Applications". Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology. Disponible en: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.951&rep=rep1&type=pdf

[Stalin 2008]
Stalin U. 2008. *"Disponibilidad de la Información con BackUp exec"*. Redpartner S.A., presentación. Fecha de acceso: 16 Junio 2009. Disponible en: http://www.moratalaz.jazztel.es/pdfs/disponibilidad.pdf

[Silva et al. 2009]
Silva A. , Stevens P. y Thorle K." Java-Based TFRC Protocol for Streaming Multimedia".WPI Worcester Polytechnic Institute, Computer Science Departament, Project MQP-MLC-MP00. Fecha de acceso: 16 Octubre 2009. Disponible en: http://web.cs.wpi.edu/~claypool/mqp/tfrc/appendix.pdf

[Tanenbaum 2003]
Tanenbaum A.T.2003.*"Computer Networks"*.Prentice hall 4ta edicion.Marzo

[Tanenbaum 1996]
Tanenbaum A. 1996.*"Sistemas distribuidos"*.Prentice hall.Primera ediccion,

[Widmer et al. 2001]
Widmer, J. Denda, R. Mauve, M.2001. *"A survey on TCP-friendly congestion control"*. Network, IEEE 15. Mayo. 28-37