Títol: Géminis Server

Volum: 1
Alumne: Sergi Monteagudo Martín

Director/Ponent: José A. González Alastrue
Departament: EIO
Data: 07-07-2004

Índice

Agradecimientos

A todos aquellos que pacientemente han esperado a que finalizase esta etapa que hace tiempo empecé: mis padres Pedro y Marta, mis hermanos. Y muchos otros, que sin necesidad de nombrarlos, saben que sin su apoyo y paciencia en los momentos difíciles no habría llegado a esta meta.

Barcelona, Julio de 2004

1. Introducción

1.1 Origen y Motivaciones

Mis primeros contactos con el director del presente proyecto, Jose Antonio Gonzalez, fueron en Marzo del 2003. La idea inicial que me propuso, fue la de realizar un ***simulador*** que pudiera usar se en las sesiones de laboratorio de la asignatura, remarcando que debería ser una aplicación ***multimedia***, posiblemente basada en Macromedia ® Flash. A partir de ese momento, empecé a estudiar las posibilidades que ofrecía Flash, claramente no me preocupaban las posibilidades de la plataforma para ofrecer una visualización multimedia de lo que tratábamos de hacer, pero si me preocupaba la potencia tanto del lenguaje de programación, como la potencia de cálculo, ya que se trata de un lenguaje interpretado. El lenguaje de programación que incorpora Flash, es 'ActionScript', por este motivo, empecé a estudiar las posibilidades del lenguaje implementar las funcionalidades requeridas en el proyecto.

A medida que iba madurando la definición del proyecto, se apuntó la posibilidad de que la simulación fuera como una especie de ***juego en red***, es decir, permitir a todos los alumnos realizar una serie de eventos sobre una misma simulación, viendo el mismo estado en cada uno de sus clientes al mismo tiempo. Eso planteaba la existencia de un ***servidor*** que mantuviera la coherencia de la información que ven todos los clientes. Por eso se decidió ***separar el proyecto*** en dos partes: cliente y servidor. El primero se encargaría de mostrar el estado de la simulación y enviar eventos a esta, mientras que el segundo mantendría el estado de la

simulación según los eventos que va recibiendo de los distintos clientes y comunicaría a cada uno de ellos su estado.

Surgió también la necesidad de una pequeña ***gestión del sistema***, tanto de usuarios y profesores, como de grupos a los que pertenecen, ya que en una sesión no podrían mezclarse alumnos de distintos grupos y podría darse la posibilidad de coincidir 2 o más sesiones de laboratorio en el mismo tiempo. También se planteo la posibilidad de poder almacenar las definiciones de red en una base de datos, a la que accedería el profesor para seleccionar una de ellas al iniciar la sesión, para no tener que perder tiempo al inicio de cada sesión para definir la topología de una red.

La necesidad de implementar un servidor, iba descartando poco a poco el uso de Flash, ya que el lenguaje 'ActionScript', se trata de una incorporación nueva, que no nos da la potencia necesaria para desarrollar un servidor con las características deseadas. En el capítulo 2.2.1, realizamos un breve análisis de las características de Flash.

Se planteó entonces la posibilidad de usar Java para implementar tanto la parte del cliente (que se trataría de un proyecto distinto), como la del servidor (que es el objetivo del actual proyecto), la ventaja de usar Java era la posibilidad de reusar componentes entre los dos proyectos, y de disponer un software que puede ejecutarse en cualquier plataforma. Pero después de algunas charlas con otras personas que están trabajando actualmente en el campo de la simulación, surgieron dudas sobre si la potencia de cálculo requerida, podría ser ofrecida por Java, ya que se trata de un lenguaje interpretado. Es por eso, que se decidió implementar el servidor en C++, ya que el resultado es un ejecutable en código máquina,

el cual es claramente interpretado con mayor rapidez que cualquier otro código interpretado (Java, Flash, etc.).

Era necesario pues, definir un ***protocolo de comunicación*** entre el cliente y el servidor para poder comunicarse entre ellos, independientemente de la plataforma. Es por eso se definió el 'Protocolo Géminis', basado sobre 'sockets' TCP/IP. Este protocolo desarrollado conjuntamente con el proyectista de 'Géminis Client', será la interficie que permitirá a los clientes acceder a funcionalidades que ofrece el servidor.

A partir de estos requerimientos iniciales y de otros que han ido evolucionando durante el desarrollo del proyecto, se ha obtenido como resultado '***Géminis Server.*** En los próximos capítulos definiremos con detalle la metodología empleada para el desarrollo del proyecto y mostraremos los resultados obtenidos.

El nombre de '***Géminis***' surgió por dos motivos. Géminis es una constelación y como toda constelación su nombre viene dado por la similitud de unir sus estrellas con una serie de líneas imaginarias, resultando el conjunto de ambas un dibujo que nos recuerda un animal, un dios de la mitología griega, etc.

Este conjunto de estrellas (nodos) y líneas imaginarias (vértices) también nos recuerdan a una red, que es lo que trataremos de modelizar en este proyecto. Pero en el firmamento tenemos decenas de constelaciones, en concreto el nombre de Géminis, lo usamos, por que esta en concreto, se trata se trata de un grupo de estrellas, entre las que destacan principalmente dos de ellas: Castor y Pólux, dando lugar a una forma

similar a dos gemelos; lo que tiene una cierta similitud con nuestro proyecto, ya que se trata de dos subproyectos (cliente y servidor) que tienen entre ellos algo en común, al igual que dos gemelos tienen en común una serie de rasgos y tal vez comportamiento.

Para más detalles sobre la constelación, puede consultarse el Anexo 5.

1.2 Objetivos

El proyecto '**Géminis'**, pretende desarrollar una herramienta visual para utilizar en los laboratorios de la asignatura de estadística. Queremos poner a disposición de los alumnos una herramienta que les proporcione una serie de eventos sobre los que realizar mediciones, para posteriormente aplicar, según las indicaciones del profesor, los conceptos y metodologías aprendidos en las sesiones de teoría. Estos eventos de los que deben tomar nota, podrían ser observaciones realizadas en la naturaleza, pero para facilitar esta tarea, se ha modelizado un sistema real, para poder ser estudiado en un laboratorio con la ayuda de un ordenador. Este modelo, no es idéntico al modelo real en que nos hemos fijado, de este tomar las características que más nos importan y descarta otras que no nos aportan información relevante para lo que pretendemos hacer.

En concreto '***Géminis Server***' se encarga de modelizar un sistema y realizar una simulación sobre el y permite a los clientes que conozcan el protocolo que se ha desarrollado poder interactuar con el servidor, para que de este modo puedan también interactuar con la simulación y recibir la información de su estado.

Los principales objetivos a cubrir en este proyecto, son:

- **Modelizar una red de ordenadores que transmiten mensajes entre ellos**.

  La principal función de la aplicación es la de proporcionar a los alumnos de un mecanismo que les ayude en la recopilación de

información de estado de una serie de variables para poder hacer un estudio posterior. Hemos modelado con este propósito, un escenario real con el que los alumnos están familiarizados: una red de nodos conectados entre ellos por nodos, que se intercambian mensajes.

- **Definir un protocolo de comunicación entre cliente y servidores.**

  Al separar por una parte la interficie del sistema, del estado de la simulación, y querer ofrecer la posibilidad de actuar a un grupo de personas sobre un mismo sistema (como si de un juego en red se tratase) se hace necesario definir un protocolo que asegure la correcta interrelación entre ambas partes, y que ofrezca a los clientes una interficie con la que interactuar con el servidor. Por eso en la definición de '***Géminis Protocol***' se ha tenido en cuenta ofrecer las siguientes características: anunciar un servicio en red, autentificar a los usuarios y haceros partícipes de la simulación. Siendo posible ampliarlo dicho protocolo en el futuro según las necesidades que vayan surgiendo.

- **Poder observar el estado de la simulación y actuar sobre esta.**

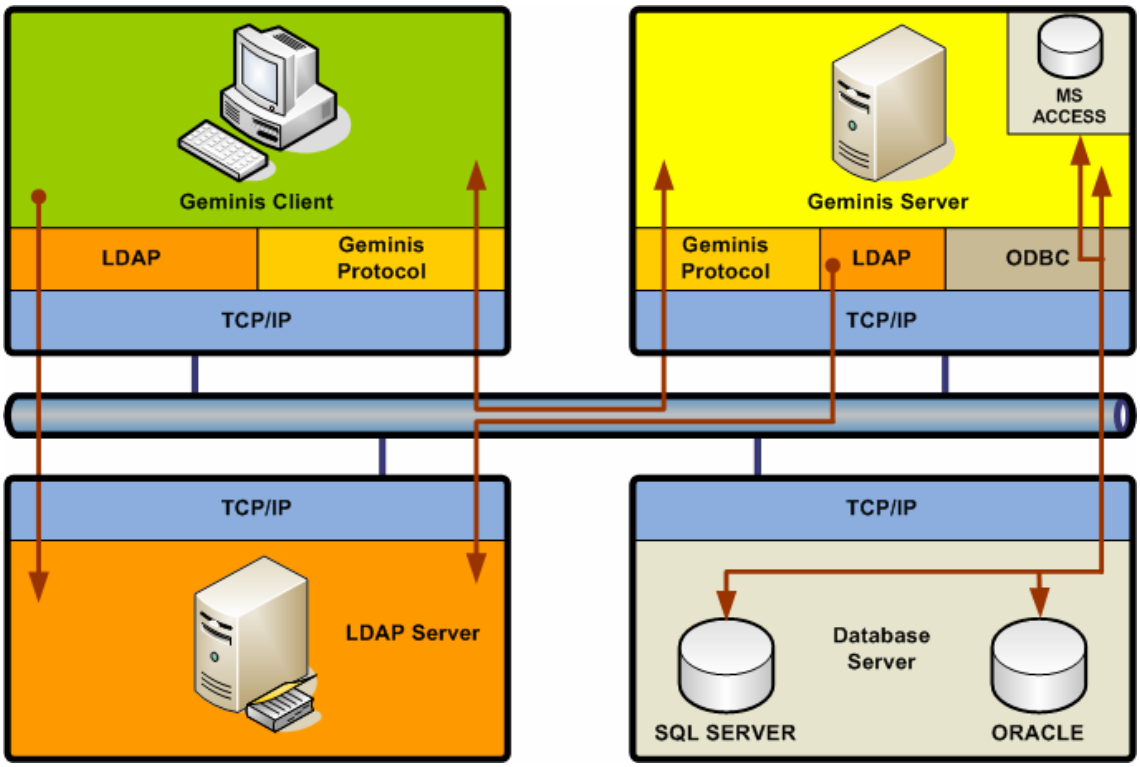
  Otro de los objetivos a cubrir en este proyecto es proporcionar un mecanismo para que los clientes que se conectan al servidor, tengan la posibilidad de ver el estado de la simulación que en él se ejecuta y poder actuar sobre ella. Para ello se ha definido un modulo de comunicaciones que envía a todos los clientes que previamente se han registrado en la sesión el estado de lo que ocurre en cada momento.

- **Registrar la actividad de usuarios.**

  Cada vez que un usuario inicia una sesión en una simulación guardaremos su información en la base de datos. Del mismo modo, una vez finalice su sesión esta información se actualizará. De este modo pretendemos ofrecer al profesor información sobre la asistencia de los alumnos a las sesiones de laboratorio, así como el uso que hacen de la sesión. En la actualizad solo registramos el tiempo en que un usuario participa de una sesión, pero se podrían introducir nueva información a registrar, según el criterio de los profesores, por ejemplo, el número de mensajes enviados por un alumno a lo largo de una sesión, etc.

1.3 Planteamiento

1.3.1 Estructuración



El proyecto '***Géminis***' se divide en dos partes:

- ***Géminis Server***: Es la parte encargada de simular una red y coordinar a los distintos clientes que se conecta a ella para visualizarla y participar en ella añadiendo nuevos eventos.
- ***Géminis Client***: Es la parte encarga de representar el estado de la simulación y de enviar al servidor los eventos que los alumnos vayan generando.

1.3.2 Descripción del proyecto

Modelización

'**Géminis**' *modeliza* una red de ordenadores (que también llamaremos nodos), los cuales se envían mensajes entre ellos. Estos mensajes son cadenas de texto, las cuales para ir de un nodo origen a su destino, tienen que atravesar los canales de comunicación (o vértices) que unen estos nodos. A partir de aquí se ira generando una información que es la que los alumnos deberán ir capturando para analizarla posteriormente.

Autenticación del profesor

Al iniciar la aplicación '***Géminis Server***' el profesor introducirá las credenciales que habitualmente usa para autenticarse en cualquier ordenador de la universidad, estas serán validadas en un servidor LDAP, para a continuación dar la opción al usuario de elegir una definición de red predefinidas que se encuentra almacenada en una base de datos, que será la que usaremos para crear los objetos que formen parte de la simulación. El profesor, también deberá elegir el grupo al que estará destinada la sesión.

Anuncio de servicios en la red

Una vez iniciada la sesión, el servidor *anunciará sus servicios* en la red (*broadcast*). Se ha escogido este mecanismo, ya que de este modo, los servidores pueden iniciarse cada sesión en un ordenador con una dirección de red distinta, evitando así la necesidad de tener que configurar los

clientes con una dirección de red fija. Serán pues los clientes los que tendrán que descubrir los servicios de 'Géminis Server' que hay en la red, siendo posible la existencia de más de uno al mismo tiempo ya que pueden coincidir varias sesiones en el mismo horario, por eso, en la información de descubrimiento que emite el servidor, se incluyen el grupo al que se dará servicio y el profesor que inicia la sesión, de este modo los alumnos tendrán información suficiente para escoger la sesión adecuada.

Parametrización de los clientes

Para evitar configurar los clientes con ficheros de parámetros, información en el registro o bien en el mismo código de la aplicación, se les proporciona un mecanismo para poder preguntar al servidor información sobre parámetros que pueden cambiar con el tiempo. Por eso, una vez los clientes han elegido una sesión activa de '***Géminis Server***', preguntarán al servidor aquellos parámetros necesarios para seguir adelante con su ejecución, como por ejemplo la dirección del servidor LDAP en el que debe el cliente autenticar a los usuarios. La información que envía el servidor será la que encuentre en la base de datos de la aplicación.

Validación clientes

Una vez los clientes han autenticado a los alumnos en el servidor LDAP de la universidad, pasarán su identificador al servidor, que se encargará de verificar que pertenecen al grupo que ha iniciado la sesión de simulación.

Asociación clientes y simulación

Una vez la credencial de un cliente ha sido validada, el servidor asigna un nodo de la red simulada a la dirección IP donde se encuentra el cliente, de modo que el alumno toma posesión de dicho nodo. A partir de este momento y desde este nodo asignado, el alumno podrá enviar mensajes a otros nodos de la red. En caso de que en el cliente haya más de un usuario, todos ellos estarán asignados al mismo nodo.

En este momento el cliente empezará también a recibir del servidor, el estado de la simulación, esto incluye la topología de la red, la creación de nuevos mensajes y la localización en la que se encuentran los que están viajando en este momento.

Envió de mensajes

De forma automática, el servidor va generando mensajes entre nodos, para que de este modo siempre haya actividad en la red, el texto de estos mensaje los recupera de un base de datos, por lo que puede ser modificado regularmente si se desea. El tiempo medio entre mensajes puede ser variado por el profesor en tiempo de ejecución, para que de este modo, poder provocar saturaciones en la red o bien intentar descongestionarla.

Los clientes que se han conectado correctamente a la simulación, también tienen la posibilidad de enviar mensajes desde su nodo a cualquier otro de la red, siendo el cliente, el encargado de visualizar el estado de

dicho mensaje (o cualquier otro) en cada momento, según la información que va recibiendo del servidor.

Velocidad de simulación

Una vez iniciada la simulación, esta va actualizando el estado de los elementos según la frecuencia del reloj de la simulación, al cual se le puede aumentar o disminuir la frecuencia en tiempo de ejecución, de este modo podemos descongestionar la red o bien causar congestión según el interés del profesor.

Desconexión cliente

Cuando el cliente se desconecte de la simulación, enviará un mensaje al servidor indicando tal situación. De este modo se liberará el nodo que estaba usando de la simulación y se registrará la información sobre el uso de la simulación.

También el servidor es capaz de detectar la desconexión no controlada de un cliente, comprobando si recibe correctamente la información de estado que envía periódicamente. En caso de que el cliente no reciba esta información en un plazo de 3 segundos, considerará que este se ha desconectado y registrará igualmente los datos sobre el uso de la simulación y liberará el nodo que tenía asignado.

Finalización del servidor

Una vez termina la sesión del laboratorio el profesor puede terminar la simulación, lo provocará que se envíe a todos los clientes una notificación de finalización de la sesión en curso.

2. Contexto del proyecto

2.1 Tecnologías

2.1.1 Arquitectura cliente / Servidor

En este capítulo hablaremos acerca de la arquitectura Cliente-Servidor, ya que para nuestra aplicación utilizamos ésta arquitectura, siendo el servidor el elemento que simula un modelo, recibe eventos de distintos clientes y les comunica el estado de la simulación y los clientes visualizan el estado de la simulación, enviando al servidor eventos generados por los clientes.

En el mundo de TCP/IP las comunicaciones entre computadoras se rigen básicamente por lo que se llama modelo Cliente-Servidor, éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones.

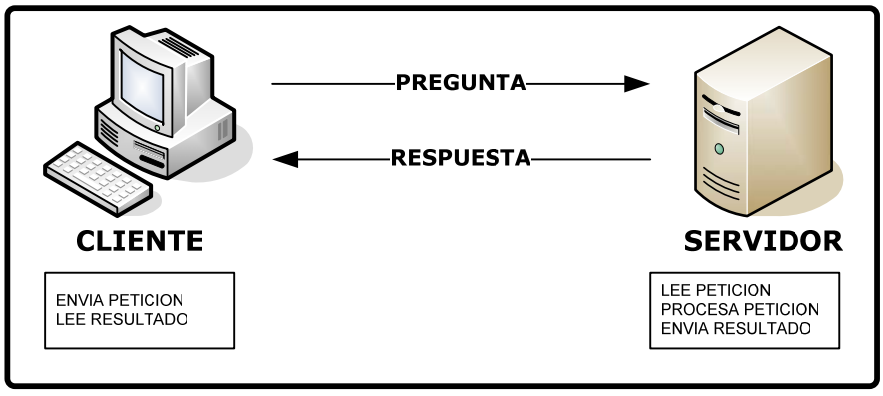


Ilustración 1 . Modelo cliente / servidor

El término Cliente/Servidor fue usado por primera vez en 1980 para referirse a PC's en red. Este modelo Cliente/Servidor empezó a ser aceptado a finales de los 80's. Su funcionamiento es sencillo [Ilustración 1]: se tiene una máquina cliente, que requiere un servicio de una máquina servidor, y éste realiza la función para la que está programado (nótese que no tienen que tratarse de máquinas diferentes; es decir, una computadora por sí sola puede ser ambos cliente y servidor dependiendo del software de configuración).

Descripción y características relevantes

La arquitectura cliente-servidor permite al usuario en una máquina, llamada el cliente, requerir algún tipo de servicio de una máquina a la que está unido, llamado el servidor, mediante una red como una LAN (Red de Área Local) o una WAN (Red de Área Mundial). Estos servicios pueden ser peticiones de datos de una base de datos, de información contenida en archivos o los archivos en sí mismos, o peticiones de imprimir datos en una impresora asociada.

Aunque clientes y servidores suelen verse como máquinas separadas, pueden, de hecho, ser dos áreas separadas en la misma máquina. Por tanto, una única máquina puede ser al mismo tiempo cliente y servidor. Además una máquina cliente unida a un servidor puede ser a su vez servidor de otro cliente y el servidor puede ser un cliente de otro servidor en la red. También es posible tener el cliente corriendo en un sistema operativo y el servidor en otro distinto.

Los clientes en una red cliente-servidor son las máquinas o procesos que piden información, recursos y servicios a un servidor unido. Estas peticiones pueden ser, por ejemplo, datos para alimentar una base de datos, aplicaciones, partes de archivos o archivos completos a la máquina cliente. Los datos, aplicaciones o archivos pueden residir en un servidor y ser simplemente accedidos por el cliente o pueden ser copiados o movidos físicamente a la máquina cliente.

Esta disposición permite a la máquina cliente ser relativamente pequeña. Para cada tipo de entorno de cliente, hay habitualmente software específico (y a veces hardware) en el cliente, con algún software y hardware análogo en el servidor. Los servidores pueden ser sistemas operativos diferentes como Windows, OS/2, Unix, Linux.

Los servidores en una red cliente-servidor son los procesos que proporcionan información, recursos y servicios a los clientes de la red. Cuando un cliente pide un recurso como, por ejemplo, un archivo, datos de una base de datos, acceso a aplicaciones remotas o impresión centralizada, el servidor proporciona estos recursos al cliente.

Las características más importantes que se distinguen en la arquitectura cliente/servidor son:

- Orientado a servicios. El servidor los ofrece y el cliente los consume.

- Compartición de recursos. Servicios ofrecidos a muchos clientes. Un servidor puede atender muchos clientes que solicitan esos servicios.

- Transparencia de ubicación. El servidor es un proceso que puede residir en el mismo aparato que el cliente o en un aparato distinto a lo largo de una red. Un programa puede ser un servidor en un momento y convertirse en un cliente posteriormente.

- Mezcla e igualdad. Tal vez de las más importantes ventajas de este paradigma. Una aplicación cliente/servidor, idealmente es independiente del hardware y de sistemas operativos; mezclando e igualando estas plataformas.

- Interacción a través de mensajes, para envío y respuesta de servicios.

- Servicios encapsulados, exponiendo los servicios a través de interfaces, lo que facilita la sustitución de servidores son afectar los clientes; permitiendo a la vez una fácil escalabilidad.

2.1.2 Simulación

La ***simulación*** es una técnica que permite imitar mediante un ordenador el comportamiento de sistemas físicos o teóricos, según unas ciertas condiciones de operación, con el objetivo de analizar, estudiar y mejorar el comportamiento de dicho sistema. Para ello, desarrollaremos en primer lugar un *modelo conceptual* que describa las dinámicas de interés, para posteriormente *implementarlo* en un simulador para poder *analizar* los resultados.

Un ***sistema***, puede definirse como una colección de objetos o entidades que interactúan entre si para alcanzar un cierto objetivo. Esta colección de objetos, puede ser un subconjunto de aquellos objetos que forman el sistema real.

El ***estado*** de un sistema puede definirse como el conjunto mínimo de variables necesarias para caracterizarlo en un cierto instante de tiempo. A estas variables las llamaremos *variables de estado*. Su número y tipo dependerán de los objetivos de nuestro estudio.

Clasificación de sistemas. Teniendo en cuenta la relación entre la evolución de las distintas variables de estado y la variable independiente 'tiempo', los sistemas los podemos clasificar como:

Sistemas continuos: las variables de estado evolucionan de forma continua a lo largo del tiempo, por ejemplo el nivel del depósito de gasolina de un vehículo.

Sistemas discretos: las variables de estado cambian únicamente en un cierto instante o secuencia de instantes, permaneciendo constantes el resto del tiempo. La secuencia de instantes en que puede producirse un cambio en esas variables suele seguir un patrón periódico.

Sistemas orientados a eventos discretos: Al igual que en los sistemas discretos, sus variables de estado cambiarán únicamente en un cierto instante o secuencia de instantes, manteniéndose constantes el resto del tiempo. La diferencia con los sistemas anteriores es que la secuencia de instantes en los que en el sistema puede producirse un cambio es aleatoria.

Sistemas combinados: Son aquellos sistemas que por sus características combinan subsistemas continuos o discretos.

La ***metodología*** *usada* ***para modelizar un sistema***, será la habitual en la resolución de problemas, teniendo en cuenta que para una representación eficiente del sistema real:

- El modelo se desarrollará a partir de una serie de aproximaciones e hipótesis, por lo que representará parcialmente la realidad

- Dicho modelo se construye para una finalidad específica, debiendo ser formulado para ser útil a dicho fin

- El modelo será equilibrará simplicidad con necesidad de recoger aspectos esenciales del sistema en estudio

Son válidas como ***definición de modelo***, cualquiera de las siguientes:

- Un modelo es un objeto o concepto que utilizamos para representar cualquier otra entidad (un sistema). Así pues, mediante un proceso de abstracción se muestran en un formato adecuado las características de interés de un objeto (sistema) real o hipotético.

- Un modelo es una representación simplificada de un sistema que nos facilitará explicar, comprender, cambiar, preservar, prever y posiblemente controlar el comportamiento de un sistema.

- Un modelo es el sustituto de un sistema físico concreto.

- Un modelo debe representar el conocimiento que se tiene de un sistema de modo que facilite su interpretación, formalizando tan sólo los factores que son importantes para los objetivos de modelado.

- Un modelo debe ser tan sencillo como sea posible, porque el desarrollo de modelos universales es impracticable y poco económico, siempre y cuando represente adecuadamente los aspectos de interés.

Clasificación de modelos. Según las características de un modelo y los objetivos del estudio de simulación, estas son algunas de las posibles clasificaciones:

Modelos estáticos respecto a modelos dinámicos

Los Modelos Estáticos suelen utilizarse para representar el sistema en un cierto instante de tiempo y en su formulación no se considera el avance del tiempo

Los modelos dinámicos permiten deducir cómo las variables de interés del sistema en estudio evolucionan respecto al tiempo

Modelos deterministas respecto a modelos estocásticos

Un *modelo determinista* es aquel en que su nuevo estado está definido a partir del estado previo y de sus entradas.

Un *modelo estocástico* requiere de una o más variables aleatorias para formalizar las dinámicas de interés. Como consecuencia, el modelo no genera un único conjunto de salida cuando es utilizado para realizar un experimento, sino que los resultados generados son utilizados para estimar el comportamiento real del sistema.

Modelos continuos respecto a modelos discreto

Los modelos continuos se caracterizan por representar la evolución de las variables de interés de forma continua. En general suelen utilizarse ecuaciones diferenciales ordinarias si se considera simplemente la evolución de una propiedad respecto al tiempo, o bien ecuaciones en derivadas parciales si se considera también la evolución respecto al espacio.

Los modelos discretos se caracterizan por representar la evolución de las variables de interés de forma discreta

Según los objetivos particulares de cada estudio, es posible describir un sistema continuo mediante un modelo discreto y, al revés, también es posible describir un sistema discreto mediante un modelo continuo.

En nuestro caso concreto, la ***Red Géminis*** se trata de un sistema formado por varios subsistemas distintos:

- Tratamiento de mensajes en el sistema (sistema discreto): el estado de las variables de estado varían a cada llamada del reloj de la simulación, el cual es posible acelerarlo o retardarlo, pero que siempre tiene un valor predecible.

- Generación de mensajes automáticos entre nodos (sistema orientado a eventos discretos), se modifica una serie de variables de estado cada cierto tiempo, que es variable aleatoria continua con una distribución exponencial.

Podemos considerar por este motivo nuestro sistema es combinado, por tener un subsistema discreto y otro orientado a sistemas discretos.

La modelización usada para definir nuestra ***Red Géminis*** tiene las siguientes características:

- Se trata de un *modelo dinámico*, ya que el estado de la red va cambiando a medida que transcurre el tiempo. Con el paso del tiempo, los mensajes van moviéndose entre los nodos de esta, siendo el estado de cada uno de los elementos distinto en cada instante.

- Es también un *modelo determinista*, ya que el estado de los elementos depende únicamente del estado anterior y de las entradas, no existiendo ninguna entrada cuyo valor sea aleatorio.

- Podemos considerarlo modelo discreto, ya que el tratamos la evolución de las variables discretamente a lo largo del tiempo.

En el Apartado [3.4], se detalla la modelización de la Red Géminis, describiendo los elementos que la forman, sus estados, las simplificaciones que se han realizado y sus propiedades.

2.1.3 Comunicación TCP/IP

El Protocolo de control de transmisión/Protocolo Internet (TCP/IP) es un conjunto de Protocolos aceptados por la industria que permiten la comunicación en un entorno heterogéneo (formado por elementos diferentes). Además, TCP/IP proporciona un protocolo de red encaminable y permite acceder a Internet y a sus recursos. Debido a su popularidad, TCP/IP se ha convertido en el estándar de hecho en lo que se conoce como *interconexión de redes*, la intercomunicación en una red que está formada por redes más pequeñas.

TCP/IP se ha convertido en el protocolo estándar para la interoperabilidad entre distintos tipos de equipos. La interoperabilidad es la principal ventaja de TCP/IP. La mayoría de las redes permiten TCP/IP como protocolo. TCP/IP también permite el encaminamiento y se suele utilizar como un protocolo de interconexión de redes.

Entre otros protocolos escritos específicamente para el conjunto TCP/IP se incluyen:

SMTP (Protocolo básico de transferencia de correo). Correo electrónico.

FTP (Protocolo de transferencia de archivos). Para la interconexión de archivos entre equipos que ejecutan TCP/IP.

SNMP (Protocolo básico de gestión de red). Para la gestión de redes.

Diseñado para ser encaminable, robusto y funcionalmente eficiente, TCP/IP fue desarrollado por el Departamento de Defensa de Estados Unidos como un conjunto de protocolos para redes de área extensa (WAN). Su propósito era el de mantener enlaces de comunicación entre sitios en el caso de una guerra nuclear. Actualmente, la responsabilidad del desarrollo de TCP/IP reside en la propia comunidad de Internet. La utilización de TCP/IP ofrece varias ventajas:

- **Es un estándar en la industria.** Como un estándar de la industria, es un protocolo abierto. Esto quiere decir que no está controlado por una única compañía, y está menos sujeto a cuestiones de compatibilidad. Es el protocolo, de hecho, de Internet.

- **Contiene un conjunto de utilidades para la conexión de sistemas operativos diferentes.** La conectividad entre un equipo y otro no depende del sistema operativo de red que esté utilizando cada equipo.

- **Utiliza una arquitectura escalable, cliente/servidor.** TCP/IP puede ampliarse (o reducirse) para ajustarse a las necesidades y circunstancias futuras. Utiliza sockets para hacer que el sistema operativo sea algo transparente.

Un *socket* es un identificador para un servicio concreto en un nodo concreto de la red. El *socket* consta de una dirección de nodo y de un número de puerto que identifica al servicio.

Históricamente, TCP/IP ha tenido dos grandes inconvenientes: su tamaño y su velocidad. TCP/IP es una jerarquía de protocolos relativamente grandes que puede causar problemas en clientes basados en MS-DOS. En cambio, debido a los requerimientos del sistema (velocidad de procesador y memoria) que imponen los sistemas operativos con interfaz gráfica de usuario (GUI), el tamaño no es un problema.

2.1.4 Bases de datos relacionales

Una ***Base de datos*** es la representación de un conjunto de entidades instancia y sus interrelaciones, la cual puede ser accedida simultáneamente por usuarios de distinto tipo. La representación de las entidades es única, pero permite usos diversos y simultáneos. Su implementación se realiza mediante un conjunto de ficheros interrelacionados, con estructuras complejas i compartidos entre diversos procesos que acceden simultáneamente.

El acceso a estos ficheros se realiza mediante ***Sistemas Gestores de Bases de Datos (SGBD)*** *o Data Base Management Systems (DBMS).* Entre los SGDB más conocidos en el mercado figuran: Oracle, Informix, DB2, SQLServer, etc, los cuales proporcionan la plataforma necesaria para poder realizar:

- consultas no predefinidas y complejas
- flexibilidad a los cambios e independencia física de los datos
- Independencia lógica de los datos
- Eliminación de redundancia
- Integridad de los datos
- Concurrencia entre usuarios
- Seguridad

La ***tabla***, es la unidad básica de almacenamiento en una BD relacional, la cual a su vez está formada por una o más unidades de

información (registro, 'row', fila), cada una de las cuales puede contener diferentes tipos de valores (campo, 'column', columna)

Una fila, se refiere a la visión horizontal de la tabla, conjunto de valores, uno por cada columna de la tabla. Una columna es la versión vertical, subdivisión de la tabla caracterizada por su nombre único dentro de la tabla y definida por un tipo específico de datos.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Tabla 1 - Tabla de empleados

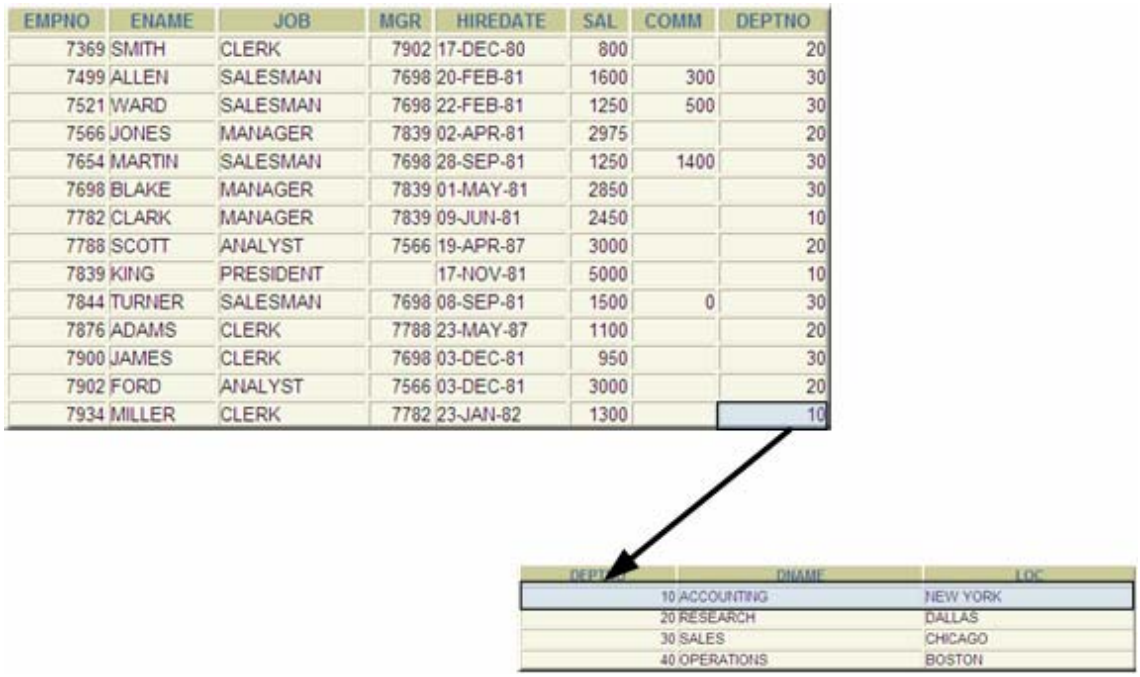
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Tabla 2 - Tabla de departamentos

El término de ***Base de Datos Relacional*** se basa en la existencia de una relación entre campos de distintas tablas, es decir, que en una tabla existe una columna (clave foránea) que hace referencia a una columna de otra tabla (clave primaria)

La ***clave primaria*** es un identificador único en una tabla (por ejemplo el DNI) mientras que una clave foránea hace referencia a una clave primaria que no tiene que ser única dentro de su tabla.

En los ejemplos anteriores, existe una relación entre la tabla de empleados y la tabla de departamentos a través del número de departamento:



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Tabla 3 - Relación entre tablas

Para manipular una base de datos usaremos el lenguaje SQL (Structured Query Language). Se trata de un lenguaje de alto nivel, muy próximo al lenguaje natural, con el que podemos realizar una serie de instrucciones sobre distintos SGDB. Este lenguaje se estructura en tres tipos de sentencia:

- ***DDL***, 'Data Definition Language' o lenguaje de definición de datos, son aquellas que permiten definir estructuras en la Base de Datos: *crear* (CREATE TABLE, CREATE INDEX), *eliminar* (DROP TABLE), *modificar* (ALTER TABLE), etc.

- ***DML***, 'Data Manipulation Language' o lenguaje de manipulación de datos, son aquellas instrucciones que nos permiten manipular datos: *leer* (SELECT), *escribir* (INSERT), *borrar* (DELETE) y *actualizar* (UPDATE)

- ***DCL***, 'Data Control Language' o sentencias de control, permiten definir permisos a los distintos usuarios sobre los objetos de la base de datos: dar permisos (GRANT), quitar permiso (REVOKE), etc.

En nuestro proyecto podemos usar cualquier SGDB que soporte consultas con el lenguaje SQL (la mayoría de ellos) y accederemos a través de la capa que nos proporciona el sistema operativo ***ODBC*** (*Open Database Connectivity*), la cual nos permite modificar fácilmente la localización de los datos sin tener que modificar nuestro programa.

2.1.5 LDAP

En el contexto de las redes de ordenadores, se denomina *directorio* a una base de datos especializada que almacena información sobre los recursos, u "objetos", presentes en la red (tales como usuarios, ordenadores, impresoras, etc.) y que pone dicha información a disposición de los usuarios de la red. Por este motivo, esta base de datos suele estar optimizada para operaciones de búsqueda, filtrado y lectura más que para operaciones de inserción o transacciones complejas. Existen diferentes estándares que especifican servicios de directorio, siendo el denominado *X.500* tal vez el más conocido.

El estándar X.500 define de forma nativa un protocolo de acceso denominado DAP (*Directory Access Protocol*) que resulta muy complejo (y computacionalmente pesado) porque está definido sobre la pila completa de niveles OSI. Como alternativa a DAP para acceder a directorios de tipo X.500, ***LDAP*** (***Lightweight Directory Access Protocol***) ofrece un protocolo "ligero" casi equivalente, pero mucho más sencillo y eficiente, diseñado para operar directamente sobre TCP/IP. Actualmente, la mayoría de servidores de directorio X.500 incorporan LDAP como uno de sus protocolos de acceso.

LDAP permite el acceso a la información del directorio mediante un esquema cliente-servidor, donde uno o varios servidores mantienen la misma información de directorio (actualizada mediante réplicas) y los clientes realizan consultas a cualquiera de ellos. Ante una consulta concreta de un cliente, el servidor contesta con la información solicitada y/o con un

"puntero" donde conseguir dicha información o datos adicionales (normalmente, el "puntero" es otro servidor de directorio).

Internamente, el modelo de datos de LDAP (derivado de X.500, pero algo restringido) define una estructura jerárquica de objetos o *entradas* en forma de árbol, donde cada objeto o entrada posee un conjunto de atributos. Cada atributo viene identificado mediante un *nombre* o acrónimo significativo, pertenece a un cierto *tipo* y puede tener uno o varios *valores* asociados. Toda entrada viene identificada unívocamente en la base de datos del directorio mediante un atributo especial denominado *nombre distinguido* o ***dn*** (*distinguished name*). El resto de atributos de la entrada depende de qué objeto esté describiendo dicha entrada. Por ejemplo, las entradas que describen personas suelen tener, entre otros, atributos como ***cn*** (*common name*) para describir su nombre común, ***sn*** (*surname*) para su apellido, ***mail*** para su dirección de correo electrónico, etc. La definición de los posibles tipos de objetos, así como de sus atributos (incluyendo su nombre, tipo, valor(es) admitido(s) y restricciones), que pueden ser utilizados por el directorio de un servidor de LDAP la realiza el propio servidor mediante el denominado *esquema* del directorio. Es decir, el esquema contiene las definiciones de los objetos que pueden darse de alta en el directorio.

Utilizaremos los servicios del servidor LDAP de la Facultad de Informática, para autentificar a los usuarios que van a utilizar '***Géminis***', tanto los alumnos, como los profesores, para ellos añadiremos a las aplicaciones tanto de cliente, como del servidor de las capacidades necesarias para dialogar con el protocolo LDAP.

2.2 Herramientas de desarrollo

Desde los planteamientos originales del proyecto hasta su aspecto final, se ha pensado en distintas alternativas para implementar la solución, descartando algunas de ellas en el camino por los motivos que detallamos a continuación.

2.2.1 Macromedia ® Flash

Una de las principales características que se pidieron al entorno 'Géminis' era el de dotar de una ***herramienta visual*** a los alumnos para poder trabajar sobre un entorno que les proporcionase eventos sobre los que estudiar una serie de características previamente aprendidas en al sesiones teóricas de la asignatura de *estadística.* Es por eso que en primer lugar se estudió la posibilidad de trabajar en *Flash*, ya que nos ofrece unas posibilidades multimedia muy potentes que no es posible encontrar en otras plataformas. Además dispone en las últimas versiones del producto, del ***lenguaje de programación*** '*ActionScript*' el cual se bastante similar a JavaScript, lo que nos proporciona una herramienta de programación estructurada lo suficientemente potente para implementar las características deseadas en nuestro proyecto.

Flash y Géminis

El principal problema que encontramos haciendo pruebas preliminares con Flash y ActionScript, era que no se trata de un lenguaje de programación orientado a objetos, por lo que no se puede aprovechar de las características que brindan este tipo de lenguajes: encapsulación, reusabilidad, mantenibilidad, herencia, polimorfismo, etc. Otros puntos en contra, es de que se trata de un entorno de desarrollo bastante novedoso, por lo que las interficies de trabajo no son bastante amigables para trabajar con grandes proyectos. Además ActionScript solo implementa una parte del estándar ECMA-Script que es el Standard de la industria en el que se basan JavaScript. Creemos que estos problemas se solucionarán en un futuro, a medida que evolucionen el propio lenguaje y la herramienta visual para trabajar con el, pero en el momento actual de desarrollar el proyecto eran puntos importantes como para pensar en otra alternativa.

2.2.2 Sun ® Java

Java fue diseñado en 1990 por James Gosling, de Sun Microsystems, como software para dispositivos electrónicos de consumo. Curiosamente, todo este lenguaje fue diseñado antes de que diese comienzo la era World Wide Web, puesto que fue diseñado para dispositivos electrónicos como calculadoras, microondas y la televisión interactiva.

En los primeros años de la década de los noventa, Sun Microsystems decidió intentar introducirse en el mercado de la electrónica de consumo y desarrollar programas para pequeños dispositivos electrónicos. Tras unos

comienzos dudosos, Sun decidió crear una filial, denominada FirstPerson Inc., para dar margen de maniobra al equipo responsable del proyecto.

Inicialmente Java se llamó Oak (roble en inglés), aunque tuvo que cambiar de denominación, debido a que dicho nombre ya estaba registrado por otra empresa. Se dice este nombre se le puso debido a la existencia de tal árbol en los alrededores del lugar de trabajo de los promotores del lenguaje.

Tres de las principales razones que llevaron a crear Java son:

- Creciente necesidad de interfaces mucho más cómodas e intuitivas que los sistemas de ventanas que proliferaban hasta el momento.

- Fiabilidad del código y facilidad de desarrollo. Gosling observó que muchas de las características que ofrecían C o C++ aumentaban de forma alarmante el gran coste de pruebas y depuración. Por ello en los sus ratos libres creó un lenguaje de programación donde intentaba solucionar los fallos que encontraba en C++.

- Enorme diversidad de controladores electrónicos. Los dispositivos electrónicos se controlan mediante la utilización de microprocesadores de bajo precio y reducidas prestaciones, que varían cada poco tiempo y que utilizan diversos conjuntos de instrucciones. Java permite escribir un código común para todos los dispositivos.

Por todo ello, en lugar de tratar únicamente de optimizar las técnicas de desarrollo y dar por sentada la utilización de C o C++, el equipo de Gosling se planteó que tal vez los lenguajes existentes eran demasiado complicados como para conseguir reducir de forma apreciable la complejidad de desarrollo asociada a ese campo. Por este motivo, su primera propuesta fue idear un nuevo lenguaje de programación lo más sencillo posible, con el objeto de que se pudiese adaptar con facilidad a cualquier entorno de ejecución.

Basándose en el conocimiento y estudio de gran cantidad de lenguajes, este grupo decidió recoger las características esenciales que debía tener un lenguaje de programación moderno y potente, pero eliminando todas aquellas funciones que no eran absolutamente imprescindibles.

Existen muchas críticas a Java debido a su lenta velocidad de ejecución, aproximadamente unas 20 veces más lenta que un programa en lenguaje C. Sun está trabajando intensamente en crear versiones de Java con una velocidad mayor.

El problema fundamental de Java es que utiliza una representación intermedia denominada código de byte para solventar los problemas de portabilidad. Los códigos de byte posteriormente se tendrán que transformar en código máquina en cada máquina en que son utilizados, lo que ralentiza considerablemente el proceso de ejecución.

Java y Géminis

Una vez descartado el uso de Flash para la implementación de cliente y servidor de la '***Red Géminis***', se pensó en la alternativa de Java, ya que es una solución multiplataforma, orientada a objetos y con una gran cantidad de herramientas de desarrollo en el mercado para gestionar un proyecto de estas características. El inconveniente encontrado en Java, es que no es recomendado utilizarlo para operaciones de cálculo intensivo, y dado que una simulación requiere de una gran capacidad de cálculo para actualizar el estado de todos sus elementos, se desestimó el uso de esta tecnología en el servidor, aunque no en el cliente.

2.2.3 Microsoft ® Visual C++

El lenguaje C nació en los laboratorios *Bell* de *AT&T* y ha sido estrechamente asociado con el sistema operativo UNIX, ya que su desarrollo se realizó en este sistema y debido a que tanto UNIX como el propio compilador C y casi la totalidad de los programas y herramientas de UNIX, fueron escritos en C. Su eficiencia y claridad han hecho que el lenguaje ensamblador apenas haya sido utilizado en UNIX.

Este lenguaje está inspirado en el lenguaje B escrito por Ken Thompson en 1970 con intención de recodificar el UNIX, que en la fase de arranque estaba escrito en ensamblador, en vistas a su transportabilidad a otras máquinas. B era un lenguaje evolucionado e independiente de la máquina, inspirado en el lenguaje BCPL concebido por Martin Richard en 1967.

En 1972, Dennis Ritchie, toma el relevo y modifica el lenguaje B, creando el lenguaje C y reescribiendo el UNIX en dicho lenguaje. La novedad que proporcionó el lenguaje C sobre el B fue el diseño de tipos y estructuras de datos. Una de las peculiaridades de C es su riqueza de operadores. Puede decirse que prácticamente dispone de un operador para cada una de las posibles operaciones en código máquina.

Este lenguaje ha evolucionado paralelamente a UNIX. Así, en 1980 se añaden al lenguaje C, características como las clases (Tomado de Simula67), chequeo y conversión de los tipos de argumentos de función, etc.; el resultado fue el lenguaje denominado "C con Clases".

En 1983/84, "C con Clases" fue rediseñado, extendido y nuevamente implementado. El resultado se denomino "Lenguaje C++". Después de algún otro refinamiento más, C++ queda disponible en 1985. Este lenguaje fue inventado por Bjarne Stroustrup (AT&T Bell Laboratories) y documentado por varios libros suyos.

El nombre C++ se debe a Rick Mascitti, significando "el carácter evolutivo de las transformaciones de C" ('++' es el operador de incremento de C).

C++ y Géminis

Finalmente se optó desarrollar Géminis con el lenguaje C++, en concreto con la plataforma de desarrollo Microsoft ® Visual C++, ya que nos ofrece un entorno RAD (Rapad Application Developement) muy elaborado y que responde a las necesidades de gestión de cualquier proyecto. En esta decisión primó la necesidad de trabajar con una aplicación que trabaja en código máquina en lugar de ser interpretado (como en los casos de Flash y Java) ya que dará una mayor potencia al simulador. También se han usado las librerías MFC (Microsft Foundation Class), para simplificar la creación de ventanas y para reusar la implementación de algunas estructuras que necesitaremos en nuestro proyecto.

3. Especificación

3.1 Requerimientos funcionales

Géminis Server, debe dar respuesta a una serie de requerimientos que se han formulado a lo largo del proyecto según las reuniones mantenidas con el director del proyecto. Algunas de estas funcionalidades han ido cambiando, evolucionando o desapareciendo a medida que iba madurando el proyecto, siendo las que se presentan en el presente, las detalladas a continuación:

En primer lugar es necesario poder identificar que profesores inician la aplicación, así como tener la posibilidad de validar sus credenciales en el directorio LDAP de la universidad, para de este modo facilitar la gestión de los passwords y que el usuario, en este caso el profesor, no tenga la necesidad de recordar nuevas contraseñas para cada nueva aplicación de la que tiene que hacer uso.

El siguiente paso es poder configurar los parámetros del servicio que pondremos en marcha en la red, para ello seleccionaremos el puerto de escucha del servidor, el cual tiene un valor por defecto que podremos modificar en caso de que existan conflictos con otras aplicaciones de la red.

Cada sesión que iniciemos estará dirigida por un profesor y destinada a un grupo de alumnos, es por ello, que deberemos seleccionar estos dos parámetros antes de poner en marcha el servidor, ya que de este modo

solo permitiremos el acceso a este a aquellos alumnos que pertenezcan al grupo que hemos seleccionado.

Para asegurarnos de que hay un tráfico mínimo en la simulación, el sistema generará automáticamente una serie de mensajes entre nodos, la frecuencia de los cuales el profesor podrá especificar antes de iniciar la simulación o durante el transcurso de esta.

Será posible acelerar el ritmo de la simulación o retardarlo según nuestras necesidades, es decir, podemos aumentar o disminuir la frecuencia de impulsos de reloj de la simulación por segundo, por defecto esta será la de 1 impulso por segundo.

Otra requerimiento de la aplicación, es el de poder registrar la actividad de los usuarios, para que posteriormente el profesor puede conocer el uso que de ella han hecho los alumnos. Para ello, cada vez que un alumno se une a la simulación registraremos esta información en una base de datos. También registraremos la desconexión de un alumno, ya sea voluntariamente o involuntariamente (se ha perdido la conexión de red)

Por último, el requerimiento más importante y el que da sentido a este proyecto, es el poder simular el comportamiento de una red que modelizaremos según unos criterios especificados en el apartado 3.3. Para agilizar las sesiones se da la posibilidad de recuperar definiciones de redes almacenadas en la base de datos.

3.2 Casos de uso

3.2.1 Caso de uso: Profesor

A continuación detallamos aquellas operaciones que puede realizar el usuario 'Profesor' con la aplicación 'Géminis Server':

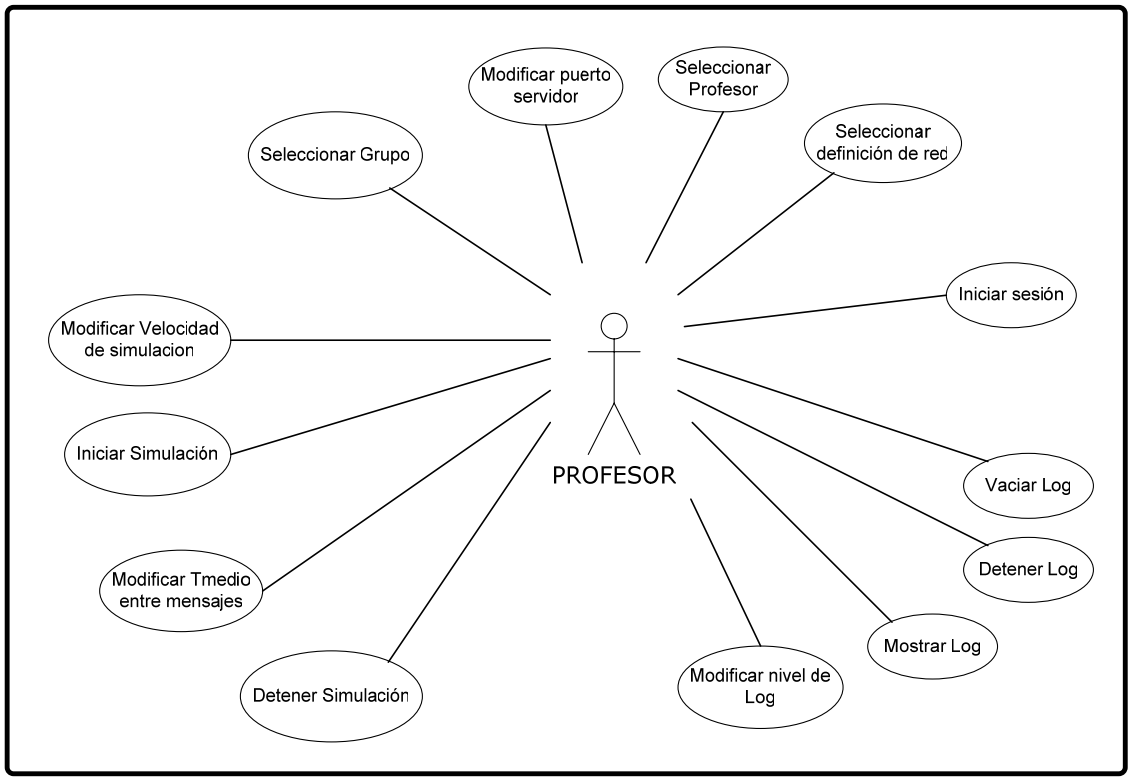


Ilustración 2 . Diagrama de casos de uso del usuario 'Profesor'

Operaciones de autenticación

	Operación	Descripción
1	Iniciar sesión	Validar las credenciales del profesor que iniciará el servidor y comprobar que tiene permisos para hacerlo

Operaciones de control del servidor

	Operación	Descripción
2	Modificar puerto servidor	Modificamos el puerto por el que los clientes se comunicarán con el servidor
3	Seleccionar grupo	Seleccionamos el grupo de laboratorio que participará en la sesión
4	Seleccionar profesor	Seleccionamos el profesor responsable de la sesión
5	Seleccionar definición de red	Seleccionamos la red que queremos simular

Operaciones de control de la simulación

	Operación	Descripción
6	Modificar velocidad de simulación	Modificamos el número de impulsos de la simulación por segundo
7	Modificar tiempo medio entre mensajes	Modificamos el tiempo medio entre mensajes automáticos
8	Iniciar Simulación	Iniciamos el reloj de la simulación, se empiezan a actualizar los estados de los elementos
9	Detener Simulación	Detenemos el reloj de la simulación

Operaciones de control de la información de Log

	Operación	Descripción
10	Modificar nivel de log	Modifica la cantidad de información que mostraremos en la ventana de Log
11	Vaciar Log	Vaciamos toda la información de Log generada hasta el momento
12	Detener Log	Detenemos la generación de Log
13	Mostrar Log	Reanudamos la generación de Log

<table>
<tr><th colspan="2">1. Iniciar sesión</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Validar las credenciales del profesor que iniciará el servidor y comprobar que tiene permisos para hacerlo</td></tr>
<tr><td>Resumen</td><td>El profesor proporciona sus credenciales de autenticación a la pantalla de login, la cual se encargará de validar esos datos en el directorio de servicios LDAP y en la base de datos de la aplicación</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Proporciona identificador de usuario y contraseña</td><td></td></tr>
<tr><td>2</td><td></td><td>Valida el identificador de usuario y contraseña en el directorio LDAP</td></tr>
<tr><td>3</td><td></td><td>Comprueba en la base de datos, si el identificador de usuario tiene privilegios para iniciar el servidor</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: si las credenciales no han podido ser validadas en el directorio LDAP
Error: si el profesor no está autorizado a iniciar el servidor</td></tr>
</table>

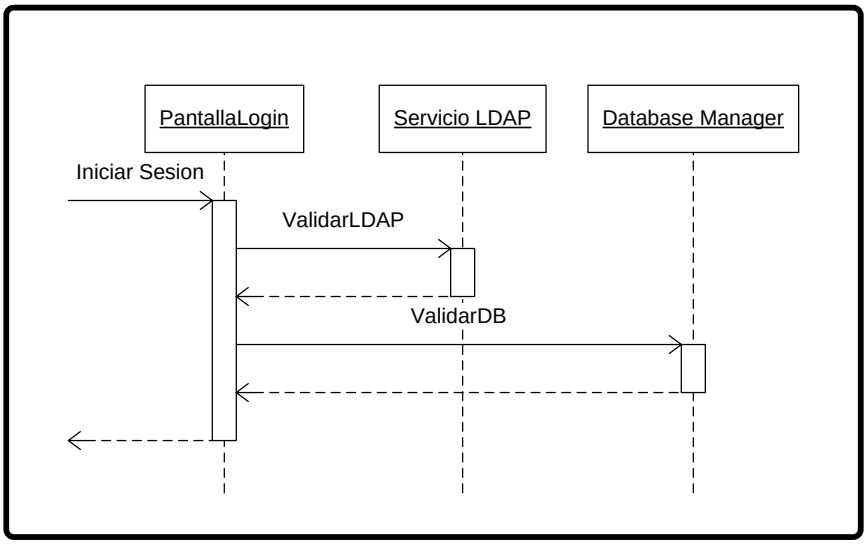


Ilustración 3 . Diagrama de secuencia 'Iniciar sesión'

<table>
<tr><th colspan="2">2. Modificar puerto del servidor</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Modificar el puerto de escucha del servidor</td></tr>
<tr><td>Resumen</td><td>El profesor modifica el puerto por el que el servidor atenderá las peticiones de los clientes</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Asignamos nuevo valor al puerto</td><td></td></tr>
<tr><td>2</td><td></td><td>Iniciamos la escucha de peticiones de clientes en el puerto</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: si el puerto no está dentro del rango válido 0.. 65536
Error: si hay otro proceso en la misma máquina escuchando por el mismo puerto que hemos seleccionado</td></tr>
</table>

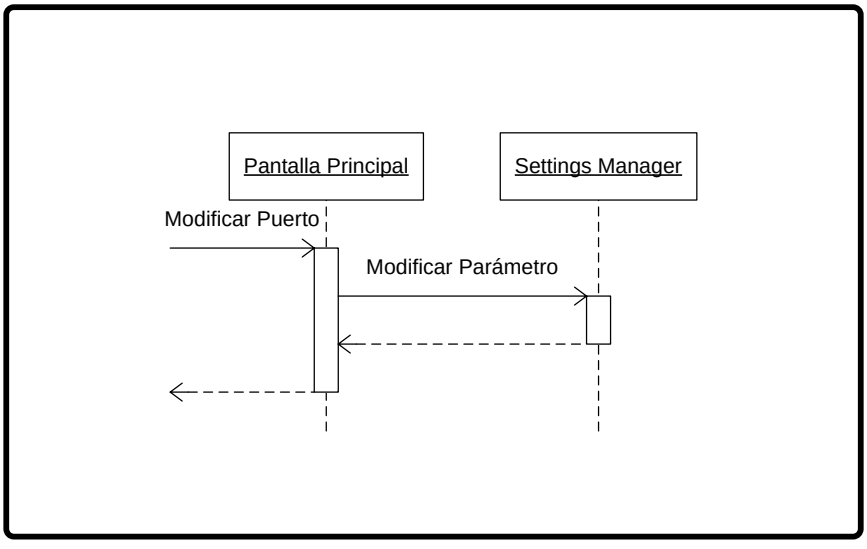


Ilustración 4 . Diagrama de secuencia 'Modificar puerto del servidor'

<table>
<tr><th colspan="2">3. Seleccionar grupo</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Seleccionamos grupo de laboratorio</td></tr>
<tr><td>Resumen</td><td>El profesor selecciona el grupo de laboratorio, dentro de una lista de grupo definidos en la base de datos. Todos los alumnos que participen en esta sesión deberán pertenecer al grupo seleccionado por el profesor</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Seleccionamos grupo</td><td></td></tr>
<tr><td>2</td><td></td><td>La información de anuncio del servicio incluye el grupo que ha seleccionado el profesor</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: si no hay grupos dados de alta en la base de datos</td></tr>
</table>

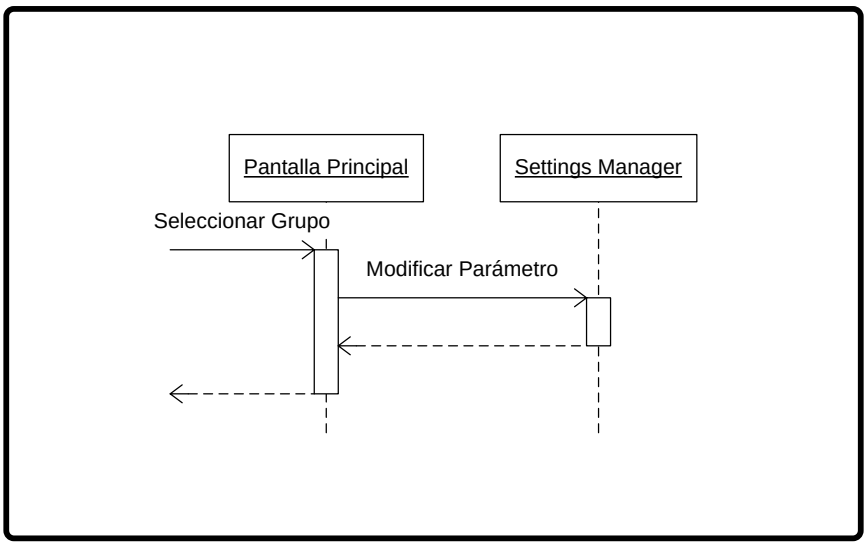


Ilustración 5 . Diagrama de secuencia 'Seleccionar grupo'

<table>
<tr><th colspan="2">4. Seleccionar profesor</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Seleccionamos profesor responsable de la sesión</td></tr>
<tr><td>Resumen</td><td>El profesor que ha iniciado la sesión, puede elegir como profesor responsable de la sesión otro que no sea el mismo (por ejemplo, por motivo de una substitución). La lista de profesores se cargará de la base de datos</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Seleccionamos profesor</td><td></td></tr>
<tr><td>2</td><td></td><td>La información de anuncio del servicio incluye el nombre del profesor responsable que se haya seleccionado</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: si no hay profesores dados de alta en la base de datos</td></tr>
</table>

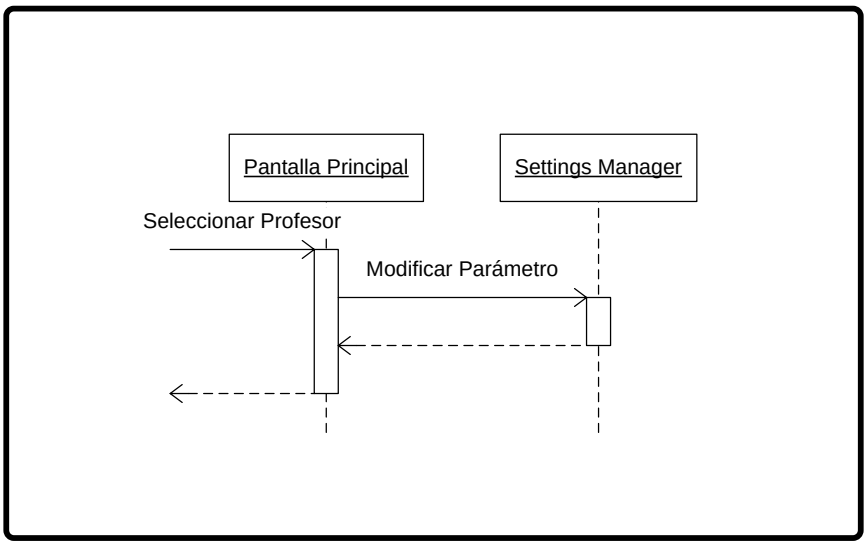


Ilustración 6 . Diagrama de secuencia 'Seleccionar profesor'

<table>
<tr><th colspan="2">5. Seleccionar definición de red</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Seleccionamos una simulación con la que trabajaremos</td></tr>
<tr><td>Resumen</td><td>El profesor seleccionará de la base de datos la definición de una red, la cual será modelizada por el simulador.</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Seleccionamos simulación</td><td></td></tr>
<tr><td>2</td><td></td><td>Se carga la definición de red y se crean los nodos y vértices según indica su definición</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: si no hay simulaciones dadas de alta en la base de datos</td></tr>
</table>

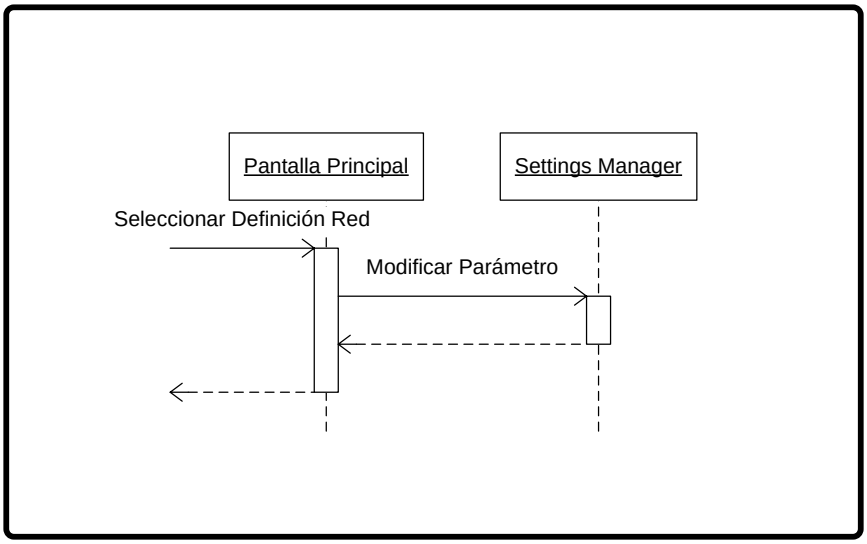


Ilustración 7 . Diagrama de secuencia 'Seleccionar definición de red'

<table>
<tr><th colspan="2">6. Modificar velocidad de simulación</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Modificar el numero de pulsos que el reloj de la simulación emite cada segundo</td></tr>
<tr><td>Resumen</td><td>El profesor seleccionará el número de pulsos que genera el reloj de la simulación, a cada pulso generado se actualizará el estado de todos los elementos de la simulación.</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Seleccionar número de pulsos por segundo</td><td></td></tr>
<tr><td>2</td><td></td><td>El reloj de la simulación generará tantos pulsos por segundo, como haya especificado el profesor</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: Si el número de pulsos no está en el rango 0..20, se descartará el valor especificado por el profesor y se usará el valor por defecto de la aplicación (1 pulso/segundo)</td></tr>
</table>

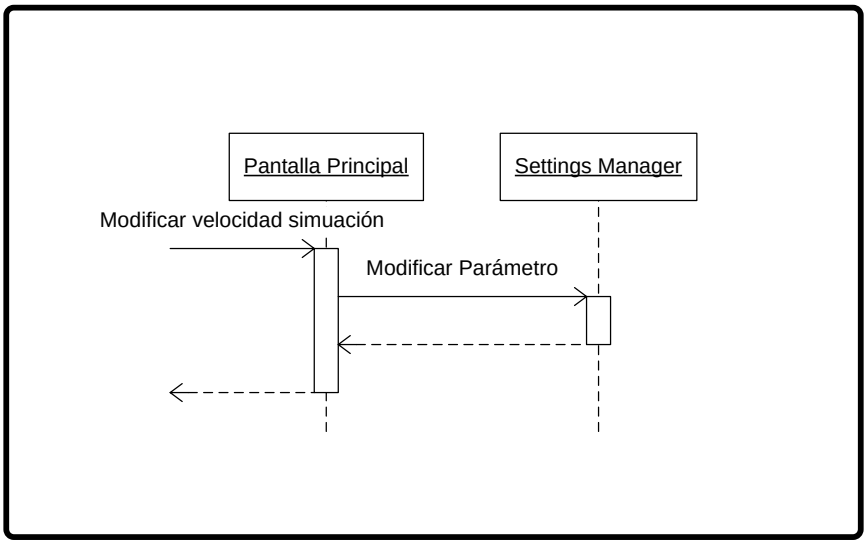


Ilustración 8 . Diagrama de secuencia 'Modificar velocidad de simulación'

<table>
<tr><th colspan="2">7. Modificar tiempo medio entre mensajes</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Modificar el tiempo medio entre mensajes automáticos</td></tr>
<tr><td>Resumen</td><td>El profesor seleccionará el tiempo medio, en segundos, entre mensajes que el sistema genera automáticamente</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Seleccionar número de mensajes por segundo</td><td></td></tr>
<tr><td>2</td><td></td><td>El generador de mensajes automáticos del sistema, ajusta el intervalo medio entre mensajes</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: Si el tiempo medio (en segundos) entre mensajes no está en el rango 0..10, se descartará el valor especificado por el profesor y se usará el valor por defecto de la aplicación (1 msj. cada 5 segundos)</td></tr>
</table>

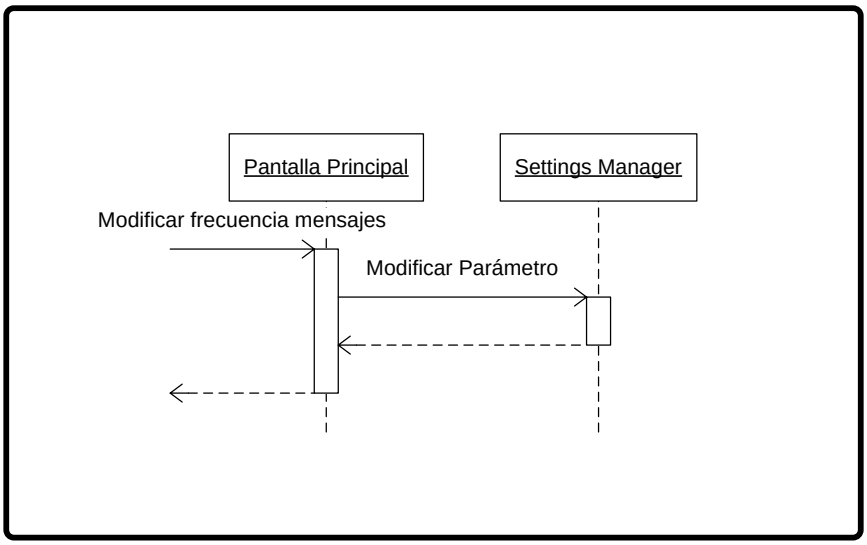


Ilustración 9 . Diagrama de secuencia 'Modificar tiempo medio entre mensajes'

<table>
<tr><th colspan="2">8. Iniciar simulación</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Iniciar la simulación</td></tr>
<tr><td>Resumen</td><td>El profesor inicia la simulación del modelo elegido de red.</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Iniciar simulación</td><td></td></tr>
<tr><td>2</td><td></td><td>Inicia el servicio encargado de enviar mensajes a los clientes, incluido el anuncio de Géminis Server</td></tr>
<tr><td>3</td><td></td><td>Se inician los procesos de escucha de clientes y analizador de mensajes</td></tr>
<tr><td>4</td><td></td><td>Se carga la definición del modelo a simular y se crean los objetos de la red</td></tr>
<tr><td>5</td><td></td><td>Se inicia el generador de pulsos de la simulación</td></tr>
<tr><td>6</td><td></td><td>Se inicia el generador automático de mensajes</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: Si no se ha seleccionado un grupo
Error: Si no se ha seleccionado un profesor responsable
Error: Si no se ha seleccionado una definición de red
Error: Si se ha seleccionado un puerto de escucha usado por otra aplicación
Error: Si no esta disponible la base de datos</td></tr>
</table>

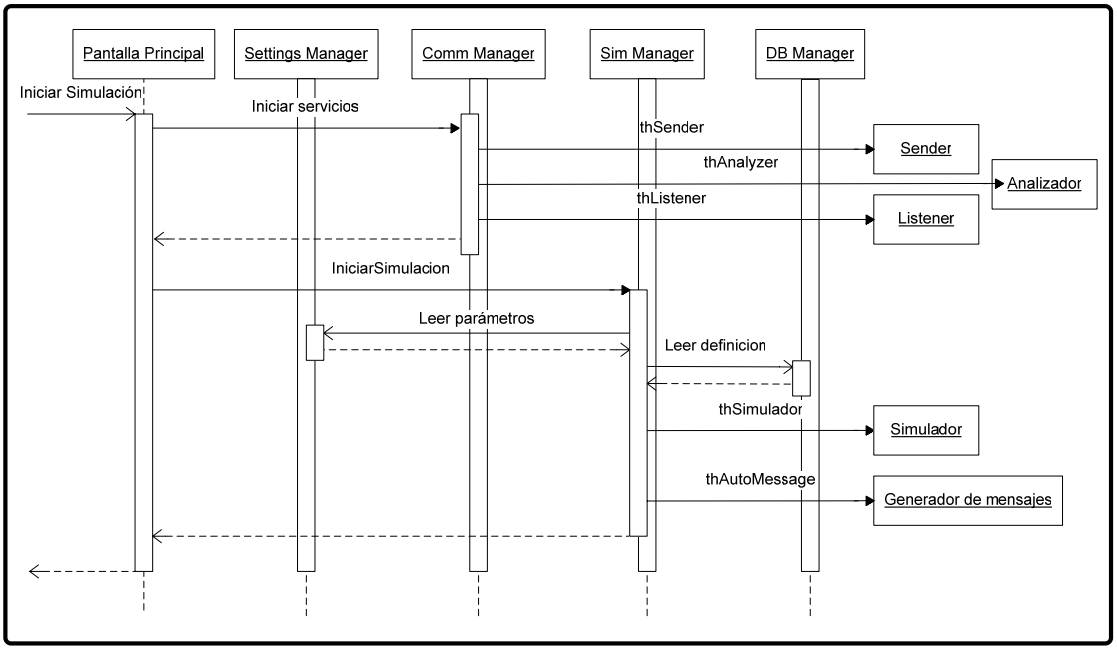


Ilustración 10 . Diagrama de secuencia 'Iniciar simulación'

<table>
<tr><th colspan="2">9. Detener simulación</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Detener la simulación</td></tr>
<tr><td>Resumen</td><td>El profesor detiene la simulación en curso</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Detener simulación</td><td></td></tr>
<tr><td>2</td><td></td><td>Se notifica a todos los clientes conectados del fin de la simulación</td></tr>
<tr><td>3</td><td></td><td>Se detienen el generador de mensajes automáticos</td></tr>
<tr><td>4</td><td></td><td>Se detienen el generador de pulsos del simulador</td></tr>
<tr><td>5</td><td></td><td>Se detiene el anuncio del servicio en la red</td></tr>
<tr><td>6</td><td></td><td>Se detiene el proceso de escucha de clientes</td></tr>
<tr><td></td><td></td><td>Se liberan los objetos generados por el simulador</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td></td></tr>
</table>

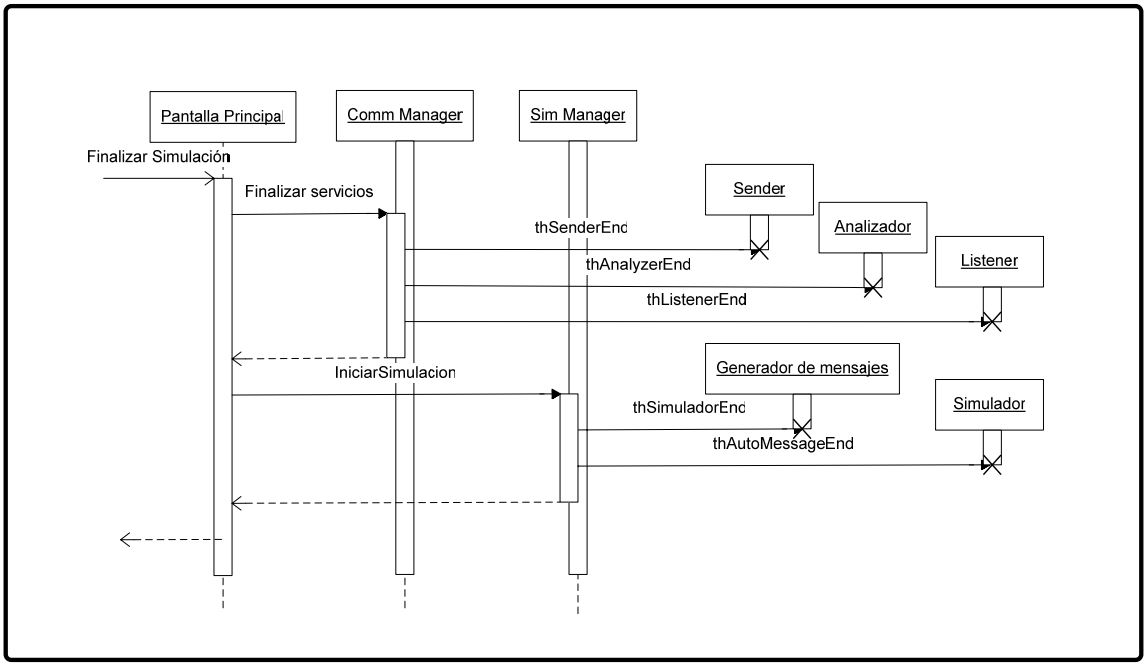


Ilustración 11 . Diagrama de secuencia 'Detener simulación'

<table>
<tr><th colspan="2">10. Modificar nivel de log</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Modificar la cantidad de información que sobre la situación de la simulación se muestra por pantalla</td></tr>
<tr><td>Resumen</td><td>El profesor modifica el nivel de log mostrado por pantalla, des de no mostrar nada a mostrar todo con detalle</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Seleccionar nivel de Log</td><td></td></tr>
<tr><td>2</td><td></td><td>El sistema muestra aquella información cuya importancia sea mayor o igual a la que ha seleccionado el profesor</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: Si el nivel de log seleccionado no está en el rango 0..4, se descartará el nuevo valor y se usará el anterior</td></tr>
</table>

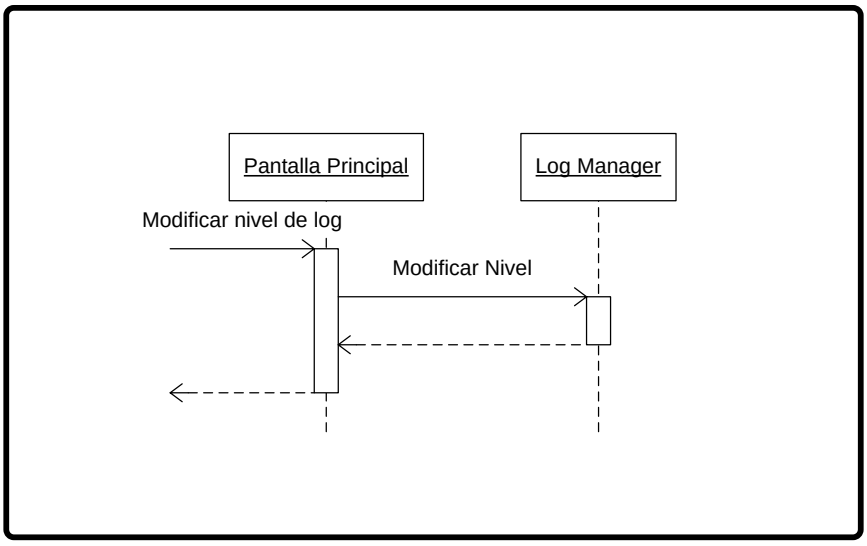


Ilustración 12 . Diagrama de secuencia 'Modificar nivel de log'

<table>
<tr><th colspan="2">11. Vaciar log</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Vaciar la información de log mostrada por pantalla</td></tr>
<tr><td>Resumen</td><td>El profesor vaciará la información que se muestra sobre el funcionamiento de la aplicación, dejando el espacio libre para que se añada nueva información</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Vaciar Log</td><td></td></tr>
<tr><td>2</td><td></td><td>Se limpiará la información de log escrita hasta el momento, liberando la memoria usada</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td></td></tr>
</table>

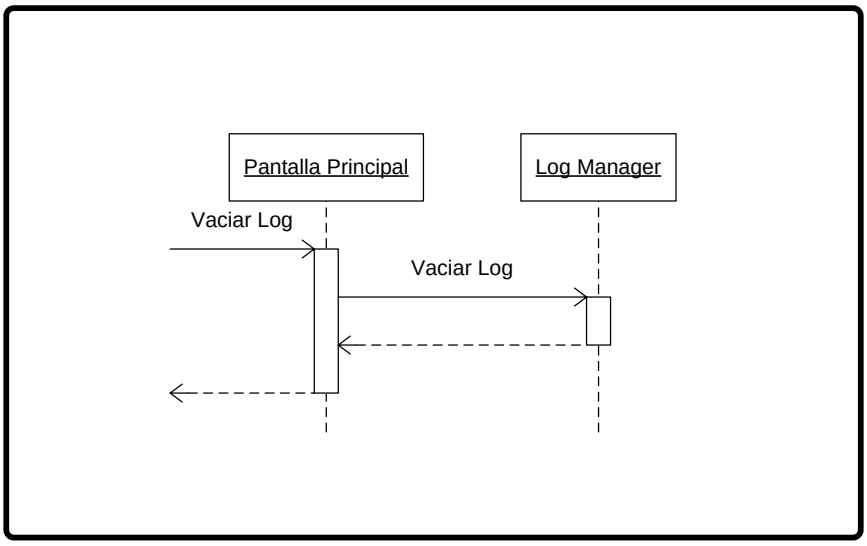


Ilustración 13 . Diagrama de secuencia 'Vaciar log'

<table>
<tr><th colspan="2">12. Detener log</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Detener la generación de log</td></tr>
<tr><td>Resumen</td><td>El profesor decide no mostrar la información de control que van generando los distintos módulos del sistema</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Detener Log</td><td></td></tr>
<tr><td>2</td><td></td><td>El sistema omite cualquier mensaje que algunos de los componentes del simulador quiera notificar al usuario</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td></td></tr>
</table>

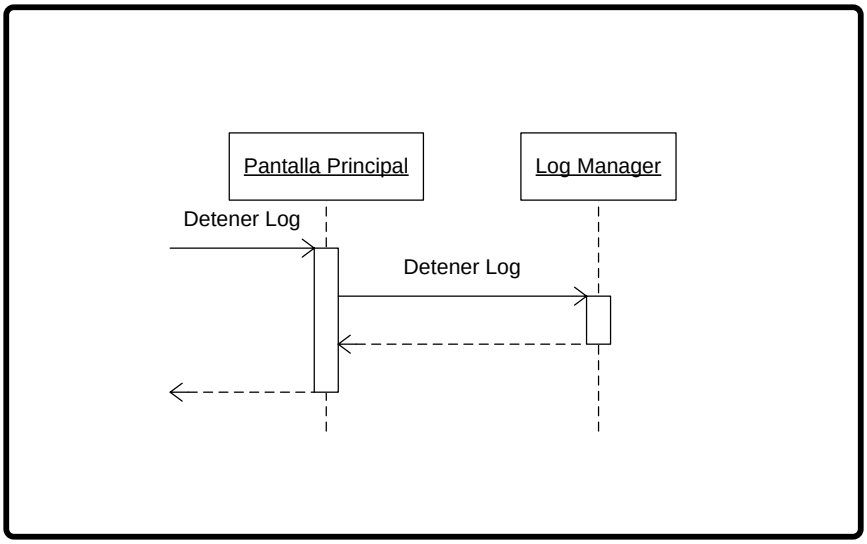


Ilustración 14 . Diagrama de secuencia 'Detener log'

<table>
<tr><th colspan="2">13. Mostrar log</th></tr>
<tr><td>Actores</td><td>Profesor</td></tr>
<tr><td>Propósito</td><td>Mostrar la generación de log</td></tr>
<tr><td>Resumen</td><td>El profesor decide mostrar la información de control que van generando los distintos módulos del sistema</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Profesor</th><th>Sistema</th></tr>
<tr><td>1</td><td>Mostrar Log</td><td></td></tr>
<tr><td>2</td><td></td><td>El sistema muestra los mensajes generados por los distintos componentes del sistemas, dependiendo del nivel de Log que haya especificado el profesor</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td></td></tr>
</table>

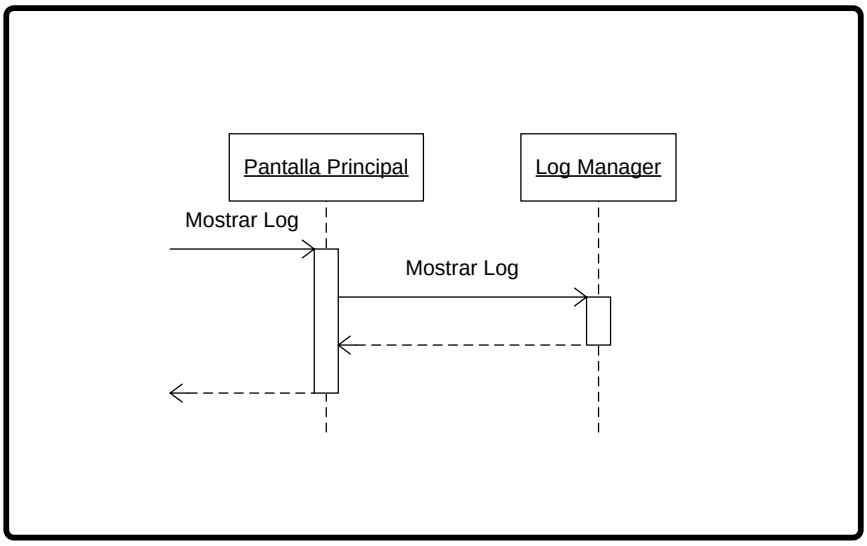


Ilustración 15 . Diagrama de secuencia 'Mostrar log'

3.2.2 Caso de uso: Géminis Client

Aunque no actuará directamente con la aplicación 'Géminis Server' todos los alumnos, mediante un cliente podrán acceder actuar con el servidor realizando una serie de funciones que son las ofrecidas por el protocolo 'Géminis Protocol' que actúa como interfície entre cliente y servidor.

El nombre de las operaciones que puede realizar un cliente está relacionado con el respectivo nombre del mensaje usado por el protocolo de comunicaciones para transmitir la llamada al servidor.

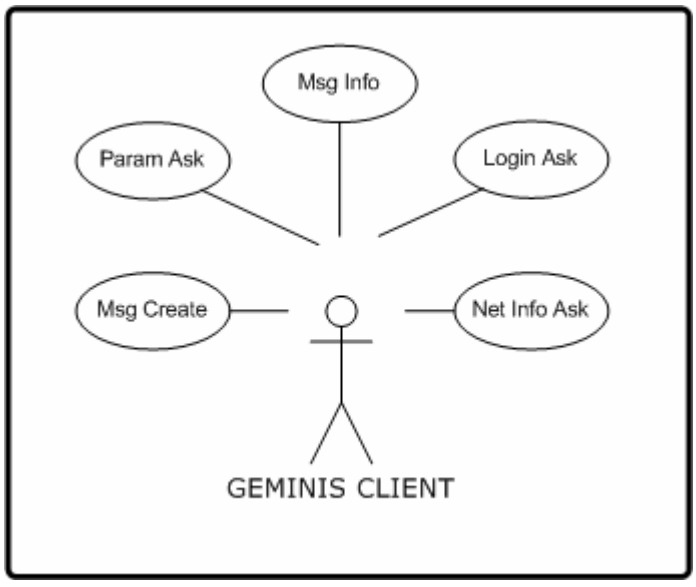


Ilustración 16 . Diagrama de casos de uso del usuario 'Géminis Client'

	Operación	Descripción
14	Param Ask	Petición de un parámetro de configuración de cliente que está almacenado en la base de datos
15	Login Ask	Paso de credenciales del usuario para verificar su pertenencia al grupo que ha iniciado la sesión
16	Msg Create	Creación de un mensaje que se incluirá en la simulación
17	Msg Info	Petición de información referente a un mensaje de la simulación
18	Net Info Ask	Petición de información referente al estado de la simulación

<table>
<tr><th colspan="2">14. Param Ask</th></tr>
<tr><td>Actores</td><td>Géminis Client</td></tr>
<tr><td>Propósito</td><td>Petición de un parámetro de configuración del cliente</td></tr>
<tr><td>Resumen</td><td>‘Géminis Client’ pedirá al servidor un parámetro necesario para configurar sus propiedades</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Géminis Client</th><th>Sistema</th></tr>
<tr><td>1</td><td>Mostrar Log</td><td></td></tr>
<tr><td>2</td><td></td><td>Buscar parámetro pedido en la base de datos</td></tr>
<tr><td></td><td></td><td>Enviar mensaje con el resultado</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: El parámetro no existe en la base de datos</td></tr>
</table>

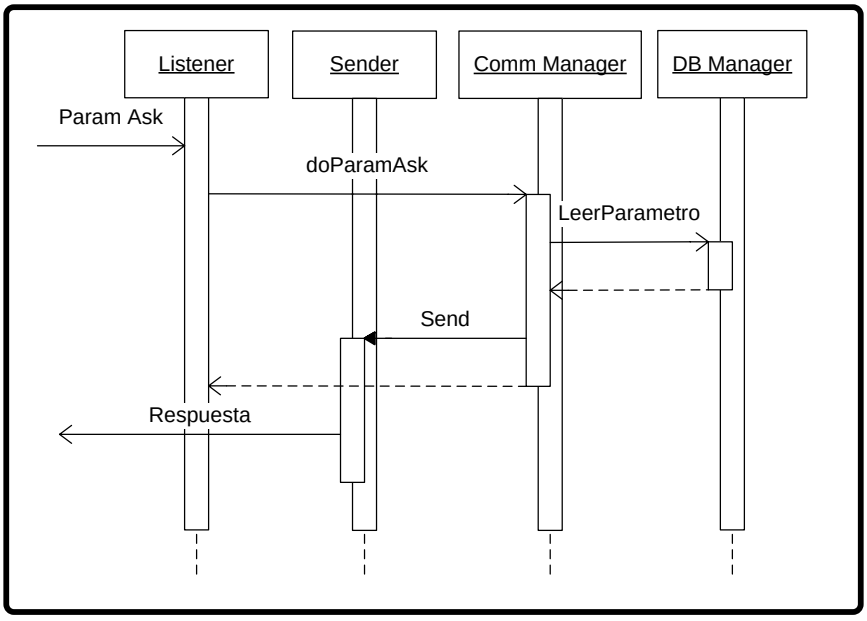


Ilustración 17 . Diagrama de secuencia 'Param Ask'

<table>
<tr><th colspan="2">15. Login Ask</th></tr>
<tr><td>Actores</td><td>Géminis Client</td></tr>
<tr><td>Propósito</td><td>Petición de formar parte de la sesión del servidor</td></tr>
<tr><td>Resumen</td><td>‘Géminis Client’ pide al servidor permiso para incorporarse a la simulación</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Géminis Client</th><th>Sistema</th></tr>
<tr><td>1</td><td>Petición de Login</td><td></td></tr>
<tr><td>2</td><td></td><td>Comprobar si el usuario pertenece al grupo que ha iniciado la sesión</td></tr>
<tr><td></td><td></td><td>Comprobar si el usuario ya ha sido autorizado en la sesión</td></tr>
<tr><td></td><td></td><td>Buscar un nodo libre para asignarlo al usuario</td></tr>
<tr><td></td><td></td><td>Enviar mensaje con el resultado</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: El usuario no pertenece al grupo que ha iniciado la simulación
Error: El usuario ya ha iniciado una sesión en la simulación actual
Error: No hay nodos libres disponibles</td></tr>
</table>

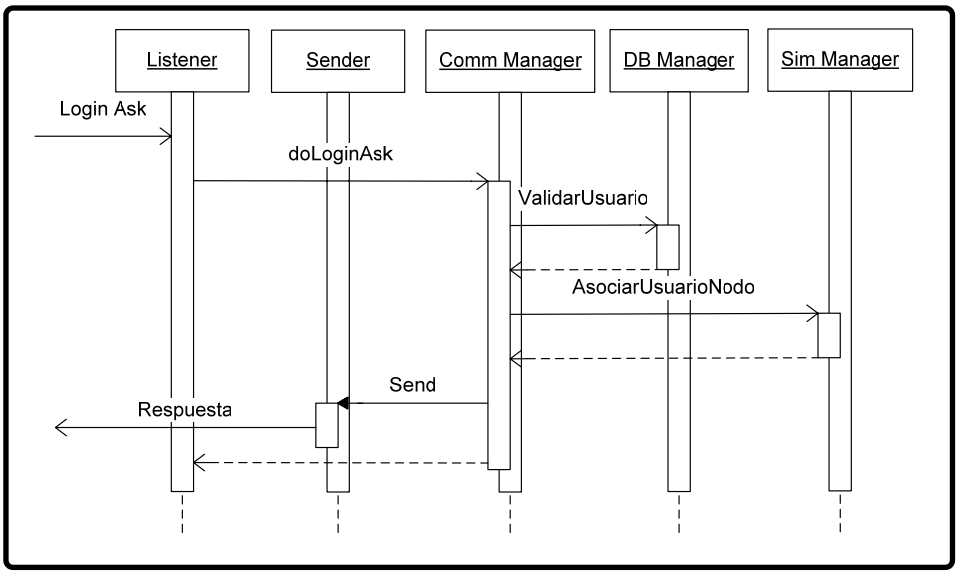


Ilustración 18 . Diagrama de secuencia 'Login Ask'

<table>
<tr><th colspan="2">16. Msg Create</th></tr>
<tr><td>Actores</td><td>Géminis Client</td></tr>
<tr><td>Propósito</td><td>Añadir un nuevo mensaje en la simulación</td></tr>
<tr><td>Resumen</td><td>‘Géminis Client’ creará y añadirá un nuevo mensaje a la simulación</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Géminis Client</th><th>Sistema</th></tr>
<tr><td>1</td><td>Crear mensaje</td><td></td></tr>
<tr><td>2</td><td></td><td>Comprobamos que origen y destino sean distintos</td></tr>
<tr><td>3</td><td></td><td>Comprobamos existencia de origen y destino</td></tr>
<tr><td>4</td><td></td><td>Comprobamos existencia de ruta entre origen y destino</td></tr>
<tr><td>5</td><td></td><td>Creamos mensaje</td></tr>
<tr><td>6</td><td></td><td>Añadimos mensaje al nodo origen</td></tr>
<tr><td>7</td><td></td><td>Enviamos mensaje con el resultado</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: No existe origen
Error: No existe destino
Error: Origen y destino son iguales
Error: No existe ruta entre origen y destino</td></tr>
</table>

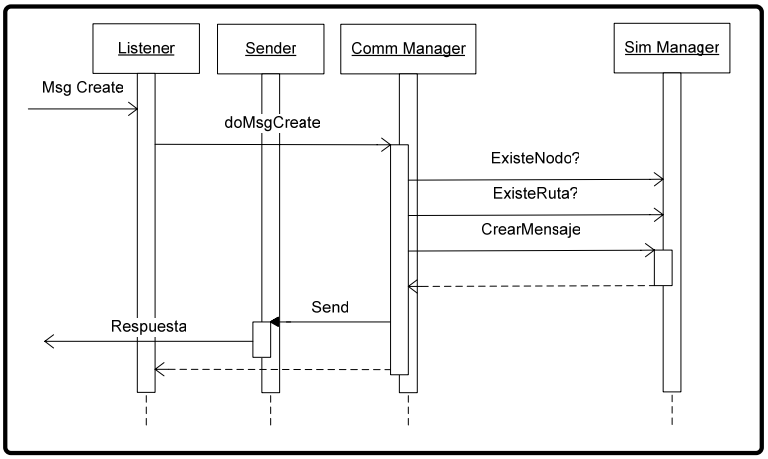


Ilustración 19 . Diagrama de secuencia 'Msg Create'

<table>
<tr><th colspan="2">17. Msg Info</th></tr>
<tr><td>Actores</td><td>Géminis Client</td></tr>
<tr><td>Propósito</td><td>Petición de información sobre un mensaje</td></tr>
<tr><td>Resumen</td><td>‘Géminis Client’ pide al sistema información sobre un determinado mensaje</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Géminis Client</th><th>Sistema</th></tr>
<tr><td>1</td><td>Petición de información</td><td></td></tr>
<tr><td>2</td><td></td><td>Comprobamos la existencia del mensaje</td></tr>
<tr><td>3</td><td></td><td>Recopilamos información sobre el mensaje</td></tr>
<tr><td>4</td><td></td><td>Codificamos información del mensaje</td></tr>
<tr><td>5</td><td></td><td>Enviamos información al cliente</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: No existe el mensaje
Error: La simulación está detenida</td></tr>
</table>

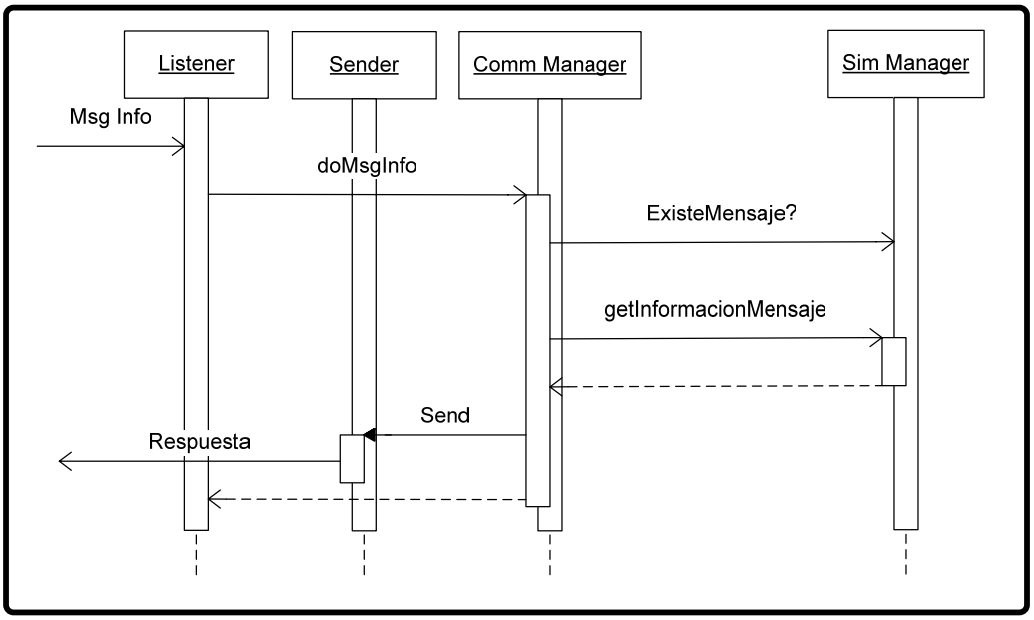


Ilustración 20 . Diagrama de secuencia 'Msg Info'

<table>
<tr><th colspan="2">18. Net Info Ask</th></tr>
<tr><td>Actores</td><td>Géminis Client</td></tr>
<tr><td>Propósito</td><td>Petición de información sobre el estado de la red</td></tr>
<tr><td>Resumen</td><td>‘Géminis Client’ pide al sistema información sobre el estado actual de la red</td></tr>
<tr><td>Flujo Normal</td><td>
<table>
<tr><th></th><th>Géminis Client</th><th>Sistema</th></tr>
<tr><td>1</td><td>Petición de información de estado</td><td></td></tr>
<tr><td>2</td><td></td><td>El sistema recopila información sobre nodos, vértices y mensajes</td></tr>
<tr><td>3</td><td></td><td>Codifica la información</td></tr>
<tr><td>4</td><td></td><td>Envía la información al cliente</td></tr>
</table>
</td></tr>
<tr><td>Flujo Alternativo</td><td>Error: La simulación está detenida</td></tr>
</table>

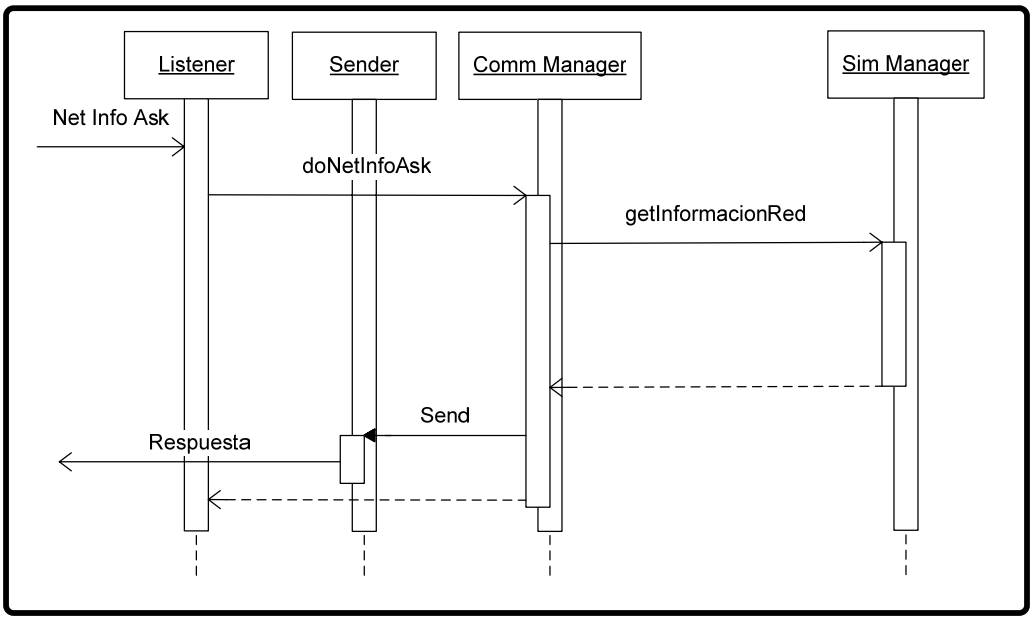


Ilustración 21 . Diagrama de secuencia 'Net Info Ask'

3.3 Modelización sistema

Para modelizar la '***Red Géminis***' se ha partido como base de las características y funcionalidades de una red de comunicaciones que intercambian mensajes entre ordenadores, tomando de ella aquellas características que resultan más interesantes en nuestra simulación.

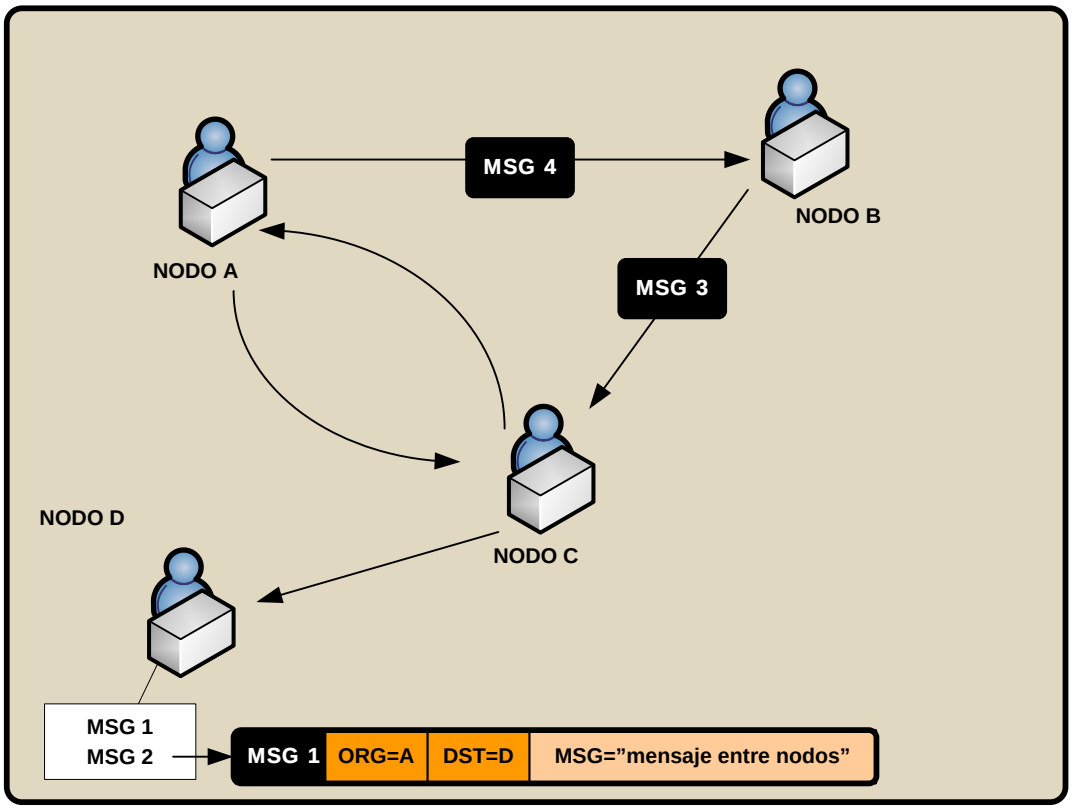


Ilustración 22 . Red Géminis

3.3.1 Elementos de la 'Red Géminis'

Los principales elementos que conforman la '***Red Géminis***' son *nodos, vértices y mensajes*. Nodos y vértices se definirán antes de iniciar la simulación, estas definiciones se cargaran de un repositorio en el que previamente, el profesor se ha encargado de almacenar redes con características y topologías de su interés (número de nodos, vértices entre nodos, dirección de vértices y su capacidad). En cuanto a los mensajes, estos se irán creando en el sistema una vez se haya iniciado la simulación, esta creación será como respuesta a eventos producidos por los alumnos, o bien por la propia simulación que se encargará de crear mensajes aleatoriamente, según una frecuencia definida por el profesor.

Existirán otros elementos en la red, que nos servirán de apoyo para la implementación de la simulación como los fragmentos de mensajes y las colas de fragmentos.

Mensajes y Fragmentos

Un ***mensaje*** es una información de texto que se envía des de un *nodo origen*, a un *nodo destino*. Para que un mensaje pueda alcanzar su destino, partiendo de su origen ira pasando por aquellos nodos a en los que exista un canal de comunicación hasta finalmente situarse en el *nodo destino*, que será a quien se le entregará la información transmitida por el *nodo origen*. A este recorrido, lo llamaremos *ruta del mensaje*. En nuestra simulación, la ruta de un mensaje se calcula en el momento en que se crea el mensaje, y se han implementado dos algoritmos para alcanzar el destino:

Ruta más corta sin repetir ningún nodo.

Ruta aleatoria sin repetir ningún nodo.

El *tamaño* de un mensaje es *ilimitado*, siendo el cliente el encargado de limitar este según su conveniencia, de este modo podemos desarrollar distintos clientes que usen el mismo simulador para distintos propósitos.

A pesar de que el tamaño de los mensajes es *ilimitado*, los canales por los que ha de pasar para alcanzar otro nodo son *limitados* por su *ancho de banda*, una propiedad que podemos definir para cada nodo, que podemos modificar durante la ejecución de la simulación y que nos indica la cantidad de bytes que pueden cruzar por un canal por cada 'clock' de la simulación.

Por este motivo, necesitamos dividir los mensajes en ***fragmentos*** de tamaño *limitado*. Serán estos fragmentos los que atravesaran los canales entre nodos. A medida que un mensaje esta siendo transmitiendo entre dos nodos a través de un canal, se van extrayendo del mensaje original fragmentos cuyo tamaño es igual a la capacidad del canal que ha de atravesar. Una vez en el nodo de destino, hasta que no llegan todos los fragmentos de un mismo mensaje no se vuelve a recomponer este para volver a pasar al siguiente nodo.

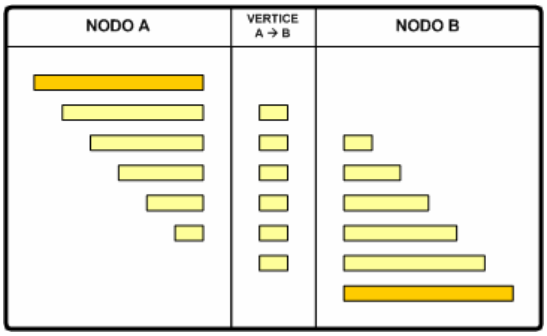


Ilustración 23 . Fragmentación de un mensaje

Nodos

Son el origen y destino de todos los mensajes que circulan por la '***Red Géminis***'. En un nodo se originará un mensaje destinado a otro nodo de la red (excepto a el mismo), siempre que exista una ruta que conecte ambos nodos.

Cada vez el nodo recibe un impulso del reloj de la red (clock de simulación), se realizarán una serie de actividades que se detallan más adelante, con el objetivo de mover los mensajes que están en la cola de entrada a las correspondientes colas de salida.

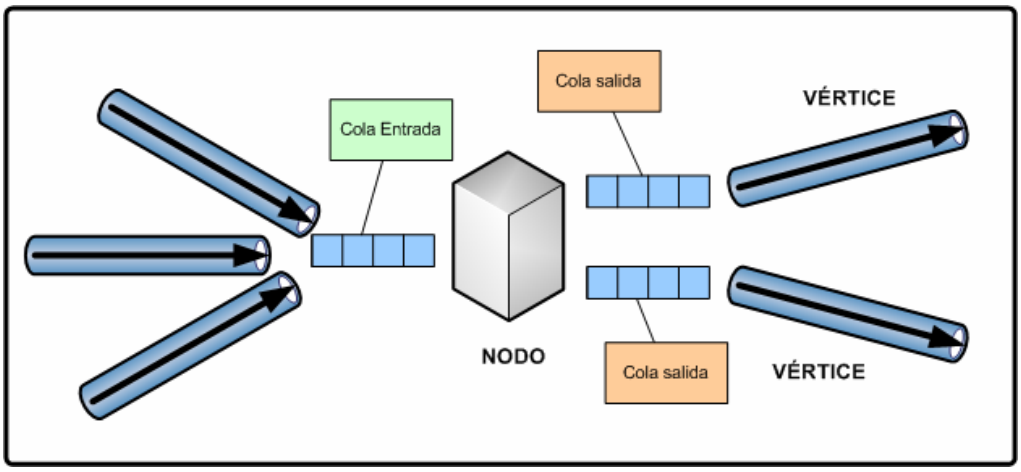


Ilustración 24 . Detalle de colas de un nodo

Para el tratar el comportamiento de los mensajes que atraviesan un nodo, éste dispondrá de las siguientes estructuras de datos:

Cola de mensajes de entrada (CE). Se trata de una cola de tipo FIFO[1] (First In First Out) de tamaño ilimitado[2] en la que almacenaremos aquellos fragmentos de mensajes que se están recibiendo de un vértice cuyo destino es el nodo actual. Una vez recibidos todos los fragmentos de un mensaje este se colocará en la cola de salida correspondiente al siguiente paso definido en su ruta.

Cola de mensajes de salida (CS). Al igual que las colas de entrada, se trata de una cola de tipos FIFO de tamaño ilimitado, en la que se van depositando los mensajes que salen de un nodo antes de alcanzar el vértice que los conducirán a otro nodo. El tamaño de esta cola nos indicará la saturación de un determinado vértice.

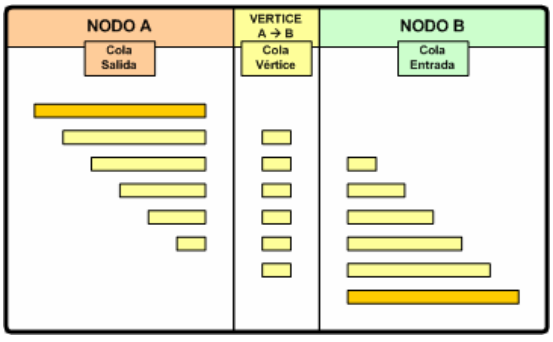


Ilustración 25 . Paso de fragmentos por las distintas colas

[1] First In First Out: El primer elemento que entre en la cola, será el primero en salir de esta.

[2] El conjunto del tamaño de todos las colas de tamaño ilimitado usadas en la simulación está limitado por la cantidad de memoria disponible en la máquina en la que se ejecute la aplicación '***Géminis Server***'

Vértices

Son el canal mediante el cual, un mensaje que está en un nodo, puede pasar a otro nodo. Cada vértice tiene como propiedad característica, su *ancho de banda*, el cual puede ser variado durante la ejecución de la simulación.

Para implementar el comportamiento de un vértice, usaremos la siguiente estructura de datos:

Cola de mensajes en vértice (CV). Al igual que en el caso de las colas de entrada y las colas de salida, descritas anteriormente, estas colas serán de tipo FIFO, contendrán fragmentos de mensajes, pero serán tamaño limitado. La limitación de su tamaño está en relación con el ancho de banda del vértice al que pertenecen, por lo que la suma de bytes de todos los fragmentos que contengan, nunca será superior a su ancho de banda.

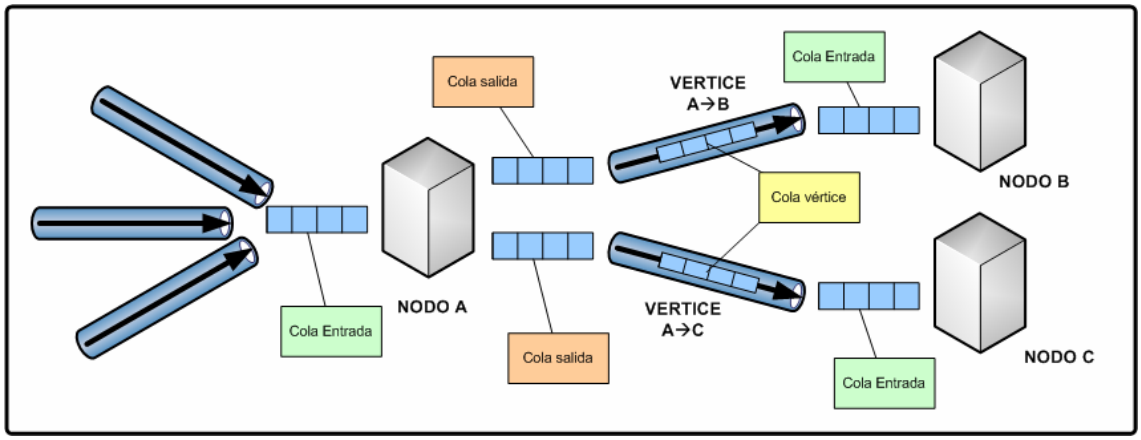


Ilustración 26 . Colas de mensajes en vertices

Reloj de simulación

Para controlar todos los elementos de la simulación, tendremos un reloj que generará impulsos continuamente, según la frecuencia que hayamos especificado. A cada impulso de reloj, actualiza el estado de todos los elementos de la red según lo detallado en el apartado 4.4.3 (Simulación).

El ritmo del reloj es posible acelerarlo o disminuirlo durante la simulación.

3.3.2 Características de la 'Red Géminis'

Tipos de Comunicación

La ***comunicación asíncrona***, conocida como «async», es probablemente la forma de conexión más extendida. Esto es debido a que *async* se desarrolló para utilizar las líneas telefónicas.

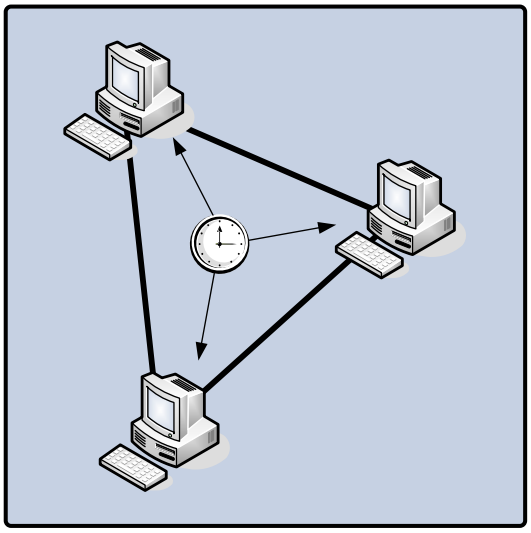
Cada carácter (letra, número o símbolo) se introduce en una cadena de bits. Cada una de estas cadenas se separa del resto mediante un bit de inicio de carácter y un bit de final de carácter. Los dispositivos emisor y receptor deben estar de acuerdo en la secuencia de bit inicial y final. El equipo destino utiliza los marcadores de bit inicial y final para planificar sus funciones relativas al ritmo de recepción, de forma que esté preparado para recibir el siguiente byte de datos.

La comunicación no está sincronizada. No existe un dispositivo reloj o método que permita coordinar la transmisión entre el emisor y el receptor. El equipo emisor sólo envía datos y el equipo receptor simplemente los recibe. A continuación, el equipo receptor los comprueba para asegurarse de que los datos recibidos coinciden con los enviados. Entre el 20 y el 27 por 100 del tráfico de datos en una comunicación asíncrona se debe al control y coordinación del tráfico de datos.

La ***comunicación síncrona*** confía en un esquema temporal coordinado entre dos dispositivos para separar los grupos de bits y transmitirlos en bloques conocidos como «tramas». Se utilizan caracteres

especiales para comenzar la sincronización y comprobar periódicamente su precisión.

Dado que los bits se envían y se reciben en un proceso controlado (sincronizado) y cronometrado, no se requieren los bits de inicio y final. Las transmisiones se detienen cuando se alcanza el final de una trama y comienzan, de nuevo, con una nueva. Este enfoque de inicio y final es mucho más eficiente que la transmisión asíncrona, especialmente cuando se están transfiriendo grandes paquetes de datos. Este incremento en eficiencia es menos destacable cuando se envían pequeños paquetes. Si aparece un error, el esquema de corrección y detección de errores síncrono genera una retransmisión.



En la simulación de la '***Red Géminis***' hemos decidido usar la transmisión *síncrona*, ya que al existir un reloj que controle dicha transmisión, nos permite la posibilidad de detenerlo, acelerarlo o decelerarlo según nuestras necesidades de simulación. Asegurándonos que el estado de

todos los elementos de nuestra red son actualizados al mismo tiempo, ya que la actualización de su estado se realizará como respuesta a un pulso del reloj de la simulación.

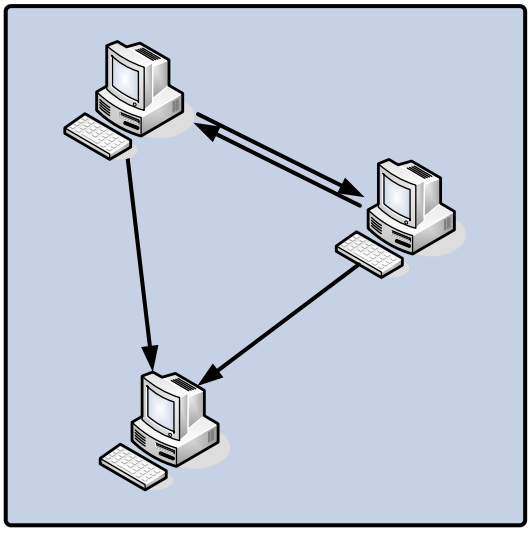
Dirección de la comunicación

La comunicación puede producirse en tres modos de diálogo:

Simple (Simplex). Un nodo transmite de manera exclusiva mientras otro recibe de manera exclusiva.

Semidúplex (Half-duplex). Un solo nodo puede transmitir en un momento dado, y los nodos se turnan para transmitir.

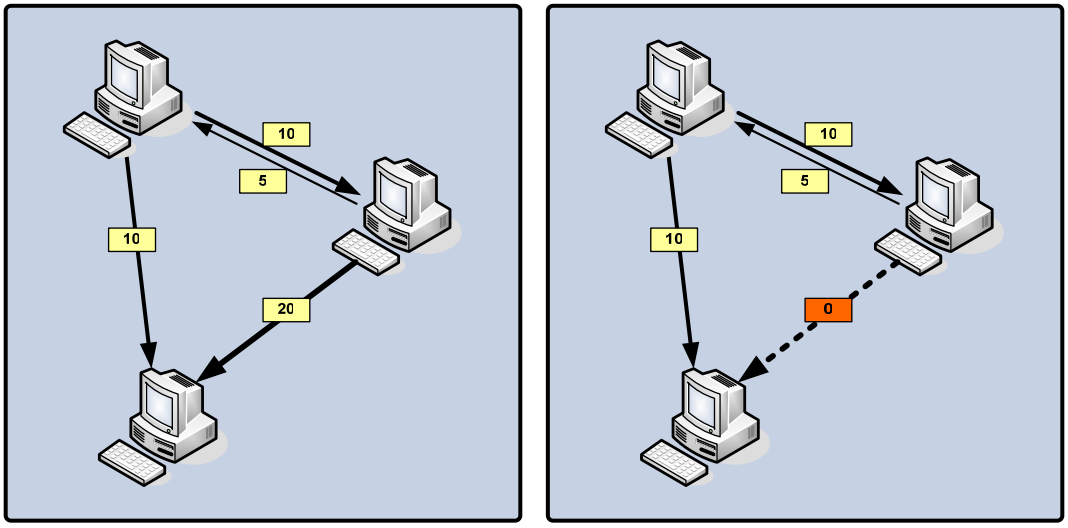
Dúplex total (Full-duplex). Los nodos pueden transmitir y recibir simultáneamente. La comunicación dúplex total suele requerir un control de flujo que asegure que ninguno de los dispositivos envía datos a mayor velocidad de la que el otro dispositivo puede recibir.



Entre dos nodos de la '***Red Géminis***', la comunicación se realizará en modo 'simplex'. Pudiendo implementar una comunicación 'full-duplex', estableciendo un canal entre cada nodo para cada sentido, pudiendo ser el ancho de banda de los canales distintos para cada sentido.

Capacidad de los canales (Ancho de banda)

Cada canal de nuestra red que une dos nodos, tendrá una capacidad distinta que vienes definida en el momento de crearlo, existiendo la posibilidad de poder variarlo en tiempo de ejecución, variando entre un máximo prefijado y el valor nulo en caso de que queramos anular un canal durante la ejecución de la simulación. Esta capacidad variable nos será de utilidad para agilizar la red en caso de congestión o para provocar saturaciones intencionadas.



Ruido

A pesar de que usualmente en las redes reales existe el ruido, lo que provoca la necesidad de la retransmisión de tramas y paquetes, en nuestra 'Red Géminis' sacrificaremos esta característica para simplificar el modelo, pudiendo sin embargo implementar una aproximación a este ruido, realizando una perdida variable de mensajes.

Enrutamiento de paquetes

Cada vez que un mensaje entra en un nodo, hay que decidir por que canal debe salir, para dirigirse a su destino. La selección de esta ruta se hará en el momento de creación del mensaje y se da la posibilidad de elegir entre dos algoritmos distintos: uno que nos proporcione el camino mas corto y otro que no proporcione un camino aleatorio. En cualquiera de los dos casos el mensaje nunca pasará dos veces por un mismo nodo.

3.4 Protocolo de comunicación entre cliente y servidor

Para poder comunicar cliente y servidor, se ha definido un protocolo implementado sobre 'sockets' TCP/IP al que llamaremos '***Géminis Protocol***'. Este protocolo permite comunicar a las distintas partes, con independencia de la implementación de cada una de ellas y la plataforma en la que se estén ejecutando.

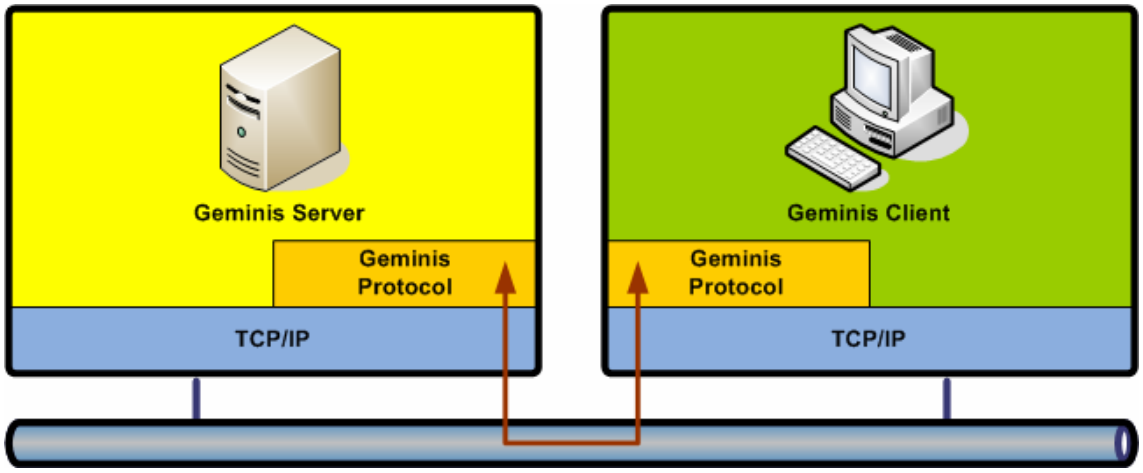


Ilustración 27 . Géminis Protocol

Los puertos usados por defecto para el intercambio de mensajes son el 3030 para el servidor y el 3031 para el cliente. Aunque pueden ser modificados fácilmente en caso de que entrasen en conflicto con otras aplicaciones existentes en la red.

Dividiremos los mensajes en tres categorías teniendo en cuenta el estado en que se encuentre la relación entre el cliente y el servidor:

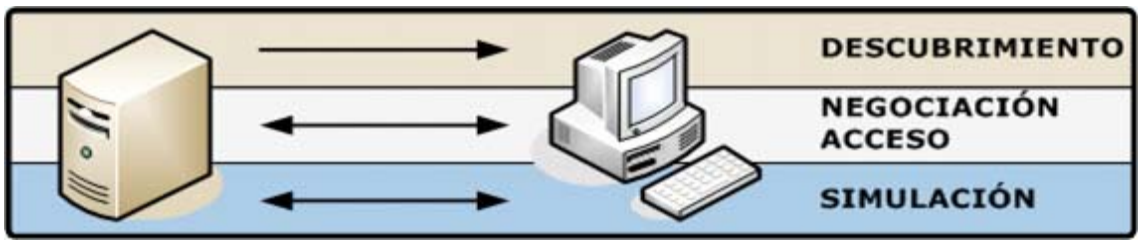


Ilustración 28 . Etapas en la negociación entre cliente y servidor

Descubrimiento: En la etapa de descubrimiento, el servidor anunciará sus servicios a todas las máquinas que se encuentren en su mismo segmento de red mediante un mensaje de '*broadcast*'[3], en el que se anunciará la localización del servidor en la red y el grupo de laboratorio al que pertenece el servidor. Este anuncio de los servicios del servidor, se producirá durante toda la simulación, existiendo la posibilidad de que haya más de un servidor anunciando sus servicios simultáneamente, siendo el cliente entonces quien decida que servidor quiere utilizar.

Negociación de acceso: En esta etapa el cliente pedirá al servidor información para poder autoconfigurarse, de este modo evitamos tener que configurar los clientes indicándoles, por ejemplo, donde se encuentra el servidor LDAP donde deben validar los usuarios. Será el cliente quien pedirá esta información al servidor, el cual la recuperará de una base de datos donde almacenamos los parámetros de los clientes. Una vez el cliente haya autentificado los usuarios en el servidor LDAP, pasará sus credenciales al servidor, el cual comprobará que pertenecen al grupo adecuado y les enviará la información necesaria para iniciar la simulación.

[3] Un mensaje de '*broadcast*', es aquel que no está dirigido a nadie en concreto, se difunde en la red y puede ser interceptado por cualquier cliente, aunque solo podrán interpretarlo por aquellos clientes que conozcan el protocolo en que se ha codificado el mensaje.

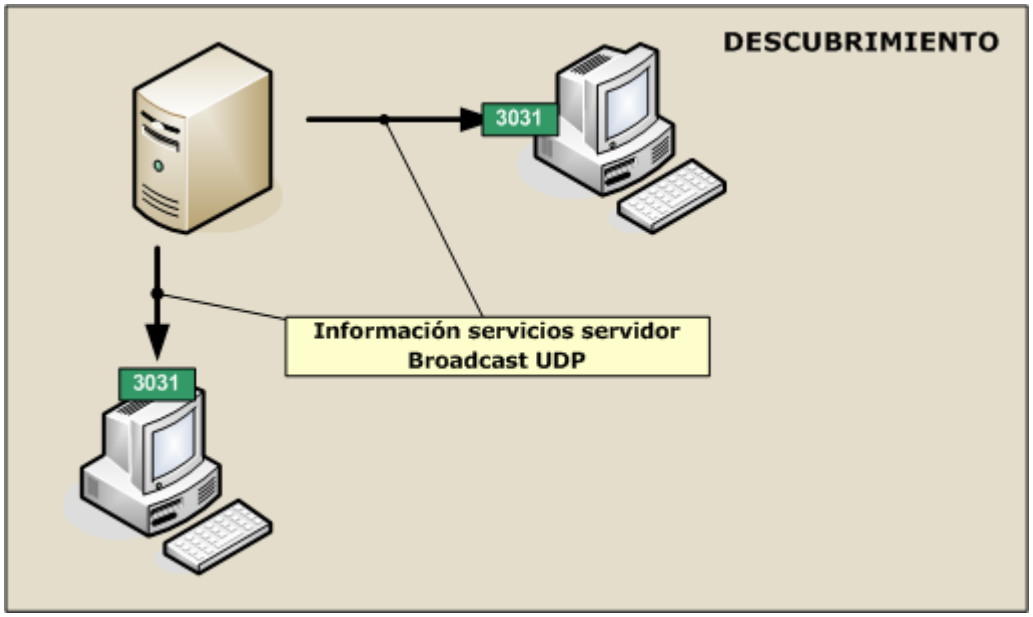
Simulación: En esta etapa los clientes enviarán eventos a la simulación y recibirán información de su estado a cada paso de esta o bien mediante una petición concreta.

El formato general de los mensajes será una cabecera indicando el tipo de mensaje, seguido de una serie de parámetros separados por comas. El contenido del mensaje es de tipo texto y se usan comas para separar cabecera y mensajes.

A continuación detallamos el formato específico para cada uno de los mensajes intercambiados entre el cliente y el servidor, así como los distintos valores que pueden tomar cada uno de los parámetros.

3.4.1 Mensajes de Descubrimiento

El servidor enviará cada segundo un mensaje para anunciar sus servicios en la red, de este modo los clientes tendrán la información necesaria para poder saber a que servidor deben conectarse La información enviada será: puerto de escucha del servidor, grupo al que atiende peticiones el servidor y profesor responsable de la sesión del servidor



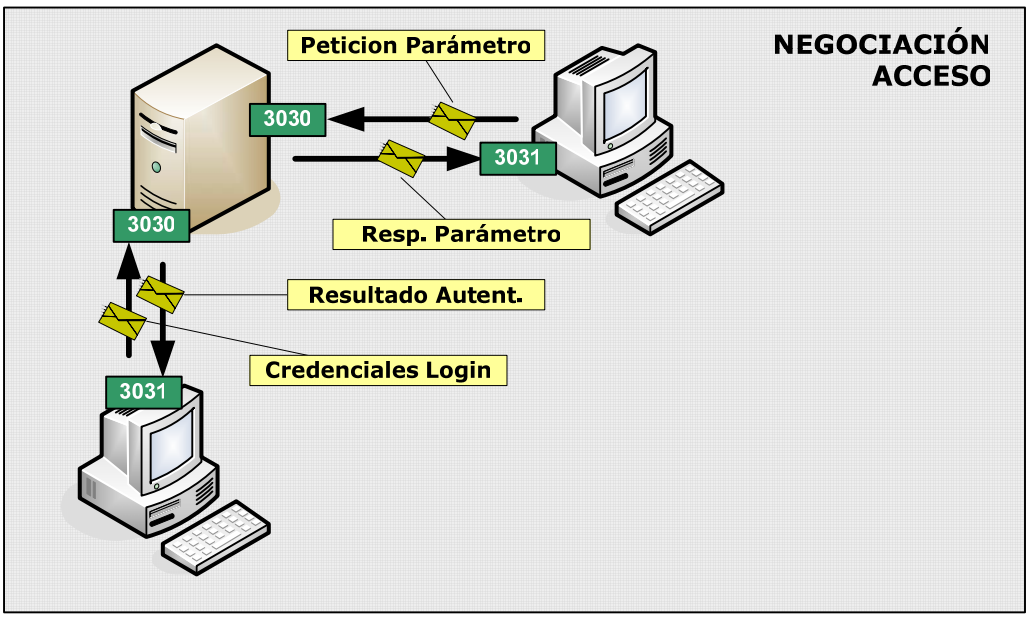
Descubrimiento Servidor

Enviamos al cliente la información sobre la sesión iniciada en el servidor. Para poder seleccionar el servidor en caso de que haya más de uno anunciando el servicio, se incluye en la información enviada al cliente, el grupo que esta permitido acceder al servidor y el profesor que ha iniciado la sesión.

<table>
<tr><th colspan="2">GeminisServer (Broadcast) : Anunciar servicio de red a los clientes</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Broadcast (cualquier cliente en el mismo segmento de red)</td></tr>
<tr><td>Permisos</td><td>Cualquier cliente puede recibir estos mensajes</td></tr>
<tr><td>Formato</td><td>1 cabecera + 2 parámetros
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“GeminisServer”</td></tr>
<tr><td>Parametro1</td><td>Puerto servidor</td><td>Número</td><td>0…65536</td></tr>
<tr><td>Parámetro2</td><td>Grupo+profesor</td><td>Texto</td><td>*</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>GeminisServer,3030,Grupo 30 (Profesor X)</td></tr>
</table>

3.4.2 Mensajes de Negociación de Acceso

Estos mensajes nos ayudarán a configurar los clientes y a validar los usuarios en el servidor LDAP de la universidad, para posteriormente pasar esas credenciales a '**Géminis Server**' para que así haga partícipe al usuario o grupo de usuarios de una máquina en la simulación. Para cada petición que se haga de acceso a la simulación des de una misma dirección de red, asignaremos un mismo nodo en la simulación, por lo que un nodo siempre tendrá asociado una dirección de red y uno o varios identificadores de alumnos.



Petición de parámetros

Una vez el cliente ha descubierto un servidor en la red, le preguntará todos los parámetros necesarios para su configuración. Evitando así la necesidad de acceder al sistema de ficheros para leer un fichero de configuración (lo que nos permite que nuestro cliente sea Applet, Servlet, etc) y evitando también el acceso de todos los clientes a la base de datos, siendo únicamente los servidores los que se conecten a esta.

<table>
<tr><th colspan="2">ParamAsk : petición de parámetro de cliente al servidor</th></tr>
<tr><td>Origen</td><td>Cliente</td></tr>
<tr><td>Destino</td><td>Servidor</td></tr>
<tr><td>Permisos</td><td>Cualquier cliente puede enviar estos mensajes</td></tr>
<tr><td>Formato</td><td>1 cabecera + 1 parámetro
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“ParamAsk”</td></tr>
<tr><td>Parametro1</td><td>Nombre del parámetro</td><td>Texto</td><td>*</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>ParamAsk,LDAPServer</td></tr>
</table>

Respuesta de parámetros

El servidor accederá a la base de datos para buscar el valor del parámetro pedido por el cliente, en caso de encontrarlo, devolverá dicho valor mediante un mensaje de texto o bien le devolverá un mensaje con un error.

<table>
<tr><th colspan="2">ParamRes : Respuesta a la petición de parámetros de cliente</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Cliente</td></tr>
<tr><td>Permisos</td><td>El servidor enviará de quien haya recibido una petición</td></tr>
<tr><td>Formato</td><td>1 cabecera + 3 parámetros
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“ParamRes”</td></tr>
<tr><td>Parametro1</td><td>Nombre del parámetro</td><td>Texto</td><td>*</td></tr>
<tr><td>Parámetro2</td><td>Resultado</td><td>Texto</td><td>“OK”,”ERROR”</td></tr>
<tr><td>Parámetro3</td><td>Valor del Parámetro</td><td>Texto</td><td>*</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>ParamRes,LDAPServer,OK,195.165.125.125:7070
ParamRes,LDAPServer,ERROR</td></tr>
</table>

Validación usuario

Una vez el cliente ha validado las credenciales de los alumnos en el servidor LDAP de la universidad, se enviará al servidor los identificadores, el cual comprobará si están trabajando en el grupo adecuado y si hay nodos libres que aun no estén usados por otros clientes, asignará un nodo a la dirección IP donde se encuentre el cliente.

<table>
<tr><th colspan="2">LoginAsk : petición inicio de sesión en el servidor</th></tr>
<tr><td>Origen</td><td>Cliente</td></tr>
<tr><td>Destino</td><td>Servidor</td></tr>
<tr><td>Permisos</td><td>Cualquier cliente puede enviar este mensaje</td></tr>
<tr><td>Formato</td><td>1 cabecera + N parámetros
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“LoginAsk”</td></tr>
<tr><td>ParametroX</td><td>Id de alumno</td><td>Texto</td><td>*</td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>LoginAsk,33000000,33500000
LoginAsk,33000000</td></tr>
</table>

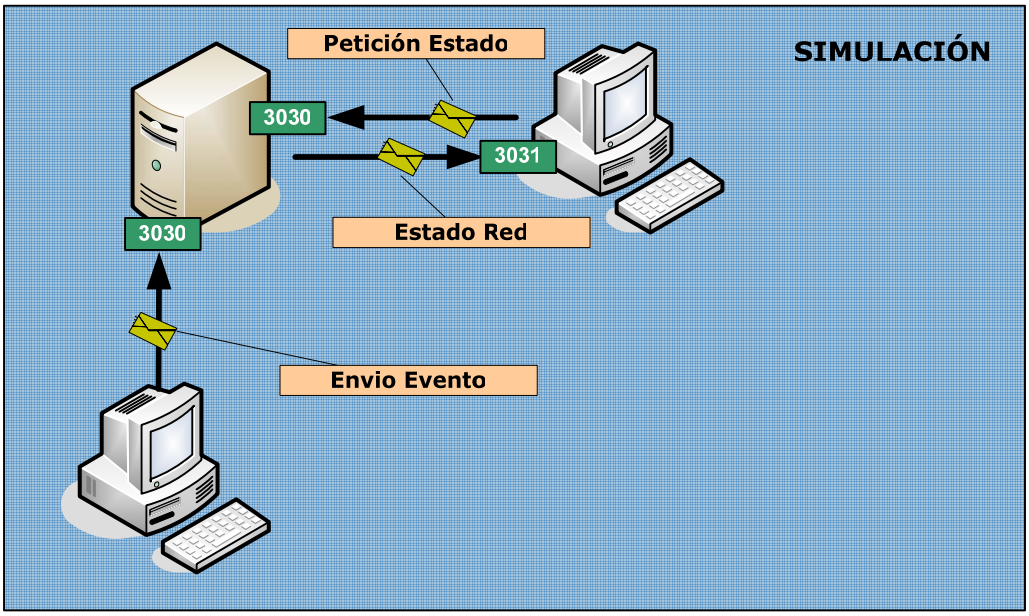
Respuesta de validación usuario

Al responder a un cliente que ha hecho una petición de validación, se le retornará en caso correcto el identificador que ha sido validado y el no do que ha sido asignado al cliente en que se encuentra el alumno.

<table>
<tr><th colspan="2">LoginRes : Resultado de la petición de inicio de sesión en el servidor</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Cliente</td></tr>
<tr><td>Permisos</td><td>El servidor enviará un mensaje por identificador de la petición</td></tr>
<tr><td>Formato</td><td>1 cabecera + 3 parámetros
<table>
<tr><th colspan="2">Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td colspan="2">Cabecera</td><td>Cabecera</td><td>Texto</td><td>“LoginRes”</td></tr>
<tr><td colspan="2">Parametro1</td><td>Id del Alumno</td><td>Texto</td><td>*</td></tr>
<tr><td></td><td>Parám.2</td><td>Resultado</td><td>Texto</td><td>“OK”</td></tr>
<tr><td></td><td>Parám.3</td><td>Nodo asignado</td><td>Texto</td><td>*</td></tr>
<tr><td></td><td>Parám.2</td><td>Resultado</td><td>Texto</td><td>”ERROR”</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>LoginResp,38500600,OK,NodoA
LoginResp,38500600,ERROR,El usuario no pertenece al grupo</td></tr>
</table>

3.4.3 Mensajes de Simulación

Son los mensajes que interactúan con la simulación y transmiten el estado de esta a los clientes. En la definición del protocolo se ha intentado minimizar al máximo la cantidad de información que se enviará por la red, por lo que al enviar el estado de la red, únicamente se envía la información de nodos, vértices y los identificadores de mensajes que están en cada uno de ellos, siendo tarea del cliente mantener la información de los mensajes que se han ido creando en el sistema, o bien pedir esa información en caso de que no la conozca.



Creación de mensaje

El cliente podrá enviar una petición de creación de mensaje al servidor. El servidor aceptará la creación de mensajes desde cualquier origen, siendo el cliente el que evite que el origen del mensaje sea distinto que el del nodo asignado al cliente.

<table>
<tr><th colspan="2">MsgCreate : Creación de un nuevo mensaje</th></tr>
<tr><td>Origen</td><td>Cliente</td></tr>
<tr><td>Destino</td><td>Servidor</td></tr>
<tr><td>Permisos</td><td>El cliente debe estar registrado en la simulación</td></tr>
<tr><td>Formato</td><td>1 cabecera + 3 parámetros
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“MsgCreate”</td></tr>
<tr><td>Parametro1</td><td>Nodo Origen</td><td>Texto</td><td>*</td></tr>
<tr><td>Parametro2</td><td>Nodo Destino</td><td>Texto</td><td>*</td></tr>
<tr><td>Parametro3</td><td>Mensaje</td><td>Texto</td><td>*</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>MsgCreate,A,B,”Mensaje enviado al servidor”</td></tr>
</table>

Respuesta de creación de mensaje

En caso de que se haya podido crear un mensaje correctamente se devolverá el identificador del mensaje al cliente. En caso de error, se le devolverá el motivo del error.

La creación de un mensaje devolverá un mensaje de error en alguno de los siguientes casos:

- origen o destino no existen
- origen y destino, son el mismo nodo.
- No existe una ruta entre origen y destino

<table>
<tr><th colspan="2">MsgCreateRes : Respuesta a la creación de mensaje</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Cliente</td></tr>
<tr><td>Permisos</td><td>El servidor enviará un mensaje por identificador de la petición</td></tr>
<tr><td>Formato</td><td>1 cabecera + 2 parámetros
<table>
<tr><th colspan="2">Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td colspan="2">Cabecera</td><td>Cabecera</td><td>Texto</td><td>“MsgCreateRes”</td></tr>
<tr><td rowspan="2"></td><td>Parám.1</td><td>Resultado</td><td>Texto</td><td>“OK”</td></tr>
<tr><td>Parám.2</td><td>Id. mensaje</td><td>Número</td><td>*</td></tr>
<tr><td></td><td>Parám.1</td><td>Resultado</td><td>Texto</td><td>”ERROR”</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>MsgCreateRes,OK,35
MsgCreateRes,ERROR,No se ha podido crear el mensaje</td></tr>
</table>

Petición de estado: Información Mensaje

Cada vez que se crea un nuevo mensaje en la red, el servidor envía un mensaje a los clientes para informarles sobre el nuevo mensaje creado en la simulación, siendo tarea del cliente almacenar esta información para su uso.

En caso de volver a necesitar esta información o requerir la información de un mensaje que se creo antes de unirse a la simulación, puede hacerlo mediante esta llamada.

<table>
<tr><th colspan="2">MsgInfo : Petición de información sobre un mensaje</th></tr>
<tr><td>Origen</td><td>Cliente</td></tr>
<tr><td>Destino</td><td>Servidor</td></tr>
<tr><td>Permisos</td><td>El cliente debe estar registrado en la simulación</td></tr>
<tr><td>Formato</td><td>1 cabecera + 1 parámetro
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“MsgInfo”</td></tr>
<tr><td>Parametro1</td><td>Id. mensaje</td><td>Número</td><td>*</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>MsgInfo,001</td></tr>
</table>

Envío de estado: Información mensaje

Cada vez que se cree un nuevo mensaje, ya sea por la acción de un usuario o bien automáticamente, se notificará a todos los clientes, los cuales deberán guardar debidamente esta información, para no tener que preguntarla continuamente al servidor, disminuyendo de este modo el tráfico de red.

También esta será la respuesta a la petición de información de un mensaje por parte de un cliente.

<table>
<tr><th colspan="2">MsgInfo : Información sobre un mensaje</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Cliente</td></tr>
<tr><td>Permisos</td><td>El cliente esta registrado en la simulación</td></tr>
<tr><td>Formato</td><td>1 cabecera + 5 parámetros
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“MsgInfo”</td></tr>
<tr><td>Parametro1</td><td>Id. mensaje</td><td>Número</td><td>*</td></tr>
<tr><td>Parametro2</td><td>Nodo Origen</td><td>Texto</td><td>*</td></tr>
<tr><td>Parametro3</td><td>Nodo Destino</td><td>Texto</td><td>*</td></tr>
<tr><td>Parametro4</td><td>Ruta</td><td>Lista</td><td>[,,,,]</td></tr>
<tr><td>Parametro5</td><td>Texto</td><td>Texto</td><td>*</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>MsgInfo,001,A,F,[A,B,E,F],En un lugar de la mancha…</td></tr>
</table>

Petición de estado: Estado de la Red

El cliente podrá a volver a pedir la información sobre un determinado mensaje. La respuesta a esta petición, será el envío al cliente de un paquete con el estado del mensaje solicitado

<table>
<tr><th colspan="2">NetInfoAsk : Pericón estado de red</th></tr>
<tr><td>Origen</td><td>Cliente</td></tr>
<tr><td>Destino</td><td>Servidor</td></tr>
<tr><td>Permisos</td><td>El cliente debe estar registrado en la simulación</td></tr>
<tr><td>Formato</td><td>1 cabecera
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“NetInfoAsk”</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>NetInfoAsk</td></tr>
</table>

Envío de estado: Estado de la Red

El servidor enviará a intervalos regulares de tiempo el estado de la red a los clientes, o bien depuse de una petición por parte de un cliente. Para reducir el tráfico de red, se enviará una lista de nodos con los mensajes que hay en cada uno de ellos. En caso de estar un mensaje en más de un nodo, lo interpretaremos como un mensaje en transito entre dos nodos.

<table>
<tr><th colspan="2">NetInfoRes : Estado de la red</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Cliente</td></tr>
<tr><td>Permisos</td><td>El cliente esta registrado en la simulación</td></tr>
<tr><td>Formato</td><td>1 cabecera + N parámetros
('N' será la suma de todos los nodos y vértices de la simulación)

<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>"NetInfoRes"</td></tr>
<tr><td>Parametro1</td><td>Info</td><td>InfoNodo</td><td>*</td></tr>
<tr><td>Parametro2</td><td>Info</td><td>InfoVertice</td><td>*</td></tr>
<tr><td>…</td><td></td><td></td><td></td></tr>
</table>

InfoNodo=N[NombreNodo,ListaFragmentosEnNodo]
InfoVertice=V[NodoOrigen,NodoDestino,AnchoDeBanda,
ListaFragmentosEnVertice]</td></tr>
<tr><td>Ejemplo</td><td>NetInfoRes,N[A,[]],V[A,B,10,[1,2]],V[A,C,15,[]],N[B,[1]],N[C,[]]</td></tr>
</table>

Finalización de la simulación

Cuando el profesor decide finalizar la simulación, se notificará dicha situación a todos los alumnos que estén conectados y se dejará de enviar el estado de la simulación

<table>
<tr><th colspan="2">NetInfoAsk : Pericón estado de red</th></tr>
<tr><td>Origen</td><td>Servidor</td></tr>
<tr><td>Destino</td><td>Cliente</td></tr>
<tr><td>Permisos</td><td>El cliente esta registrado en la simulación</td></tr>
<tr><td>Formato</td><td>1 cabecera + 0 parámetros
<table>
<tr><th>Elemento</th><th>Nombre</th><th>Tipo</th><th>Rango de valores</th></tr>
<tr><td>Cabecera</td><td>Cabecera</td><td>Texto</td><td>“SimEnd”</td></tr>
</table></td></tr>
<tr><td>Ejemplo</td><td>SimEnd</td></tr>
</table>

4. Diseño de Aplicación

4.1 Análisis de requerimientos

Como ya hemos comentado anteriormente el objetivo de este proyecto es poner en funcionamiento una simulación que permita a varios clientes ver simultáneamente su estado y poder participar en ella. Tenemos entonces, un profesor que definirá las características de la simulación y unos alumnos que mediante la interficie que proporciona el protocolo de comunicación definido, podrán conocer el estado de la simulación y realizar sobre ellas las acciones que el protocolo les permita.

Por eso nuestro servidor tiene dos partes fundamentales:

- **Gestión de las comunicaciones** entre cliente y servidor: lo que implica la definición de un protocolo de comunicación entre ambos, que define en que condiciones se realiza el intercambio de información entre ambos.

- **Gestión de la simulación**: a partir de una definición de red hay crear los objetos necesarios (nodos y vértices) para simular el comportamiento del modelo descrito.

Nuestra aplicación será accedida por dos perfiles de usuarios:

- **Profesores**, los cuales acceden directamente desde la interficie gráfica de la aplicación para definir los servicios que se ponen a disposición de los clientes.
-
- **Alumnos**, los cuales acceden al servidor, mediante la interficie que proporciona 'Géminis Protocol'.

Los datos que manejará la aplicación son de distinto tipos:

- **Gestión de usuarios**. Tanto el acceso de alumnos, como de profesores han de pasar por una autenticación en el directorio LDAP y una localización en la base de datos de la aplicación

- **Registro de actividad**. Se registrará la actividad de los alumnos.

- **Parametrización**. La parametrización tanto de clientes como del propio servidor, ser realizará mediante unas tablas que serán accedidas directamente por el servidor o por los clientes a través del protocolo 'Géminis Protocol'.

- **Definiciones de red**. El simulador recuperará definiciones de red de la base de datos, las cuales interpretará para crear los objetos en ellas indicados.

4.2 Análisis de los datos

A continuación realizaremos el estudio de los datos usados por la aplicación, detallando que información que necesitará leer la aplicación y que información necesitará almacenar de forma permanente para posteriormente consultarla.

Por una parte necesitamos identificar que ***profesores*** tendrán privilegios para iniciar el servidor, para ello recuperaremos información referente a sus identificadores junto con información más descriptiva para hacer saber a los clientes que conectan con el servidor quien es el profesor que ha iniciado el servicio y que será de utilidad en el caso de que haya más de un servidor funcionando en la red.

También necesitamos identificar que ***alumnos*** tienen permiso para ejecutar la aplicación de cliente, así como poder comprobar a que ***grupo*** pertenecen, para poder así verificar que están iniciando una sesión en el servidor adecuado.

El servidor también ***registra*** el uso que hacen los clientes de la simulación, lo cual puede servir de ayuda al profesor para conocer el grado de asistencia de los alumnos a las sesiones y la duración de estas.

El servidor también necesita ***parámetros de configuración*** tanto para configurarse a si mismo, como para configurar a los clientes que se vayan conectando a la simulación.

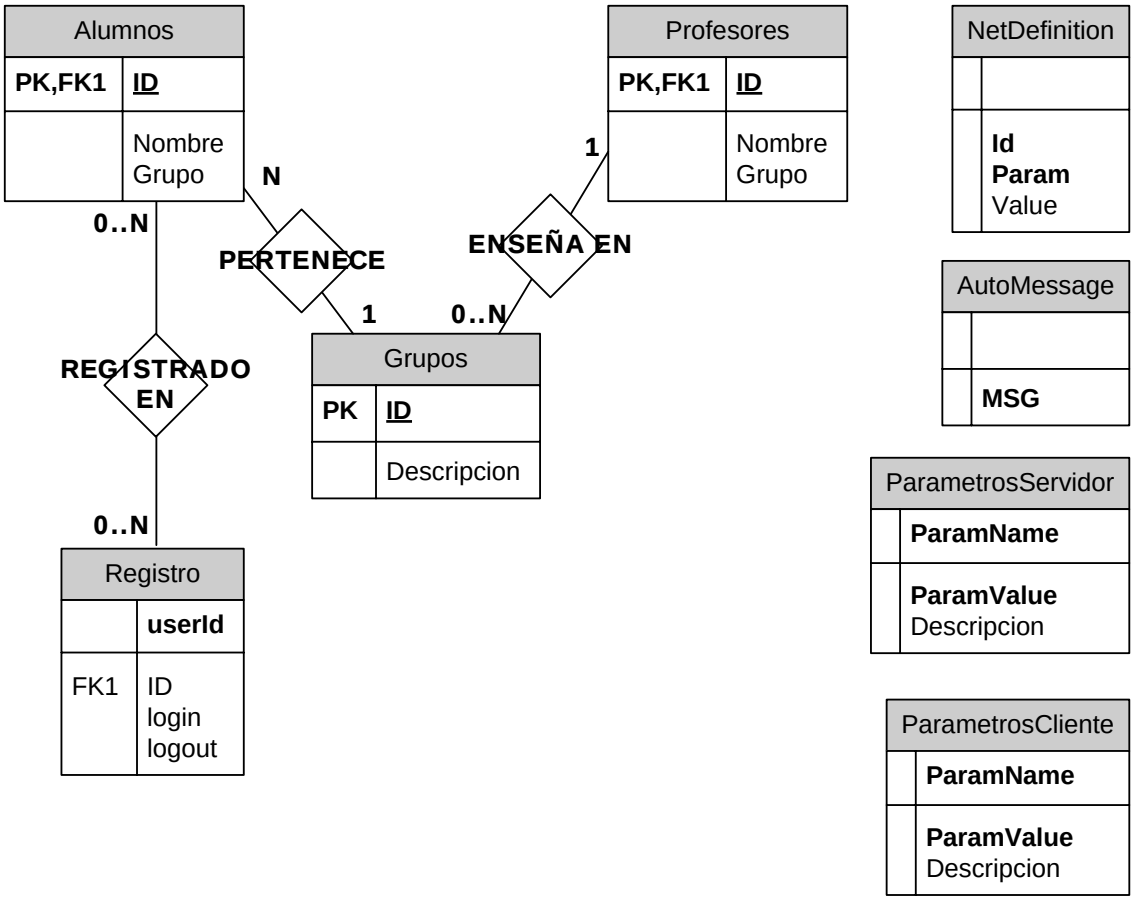
Al iniciar una sesión, cargaremos una ***definición de red*** que previamente ha sido definida y almacenada en la base de datos.

El sistema además de los mensajes que crean los propios usuarios, es capaz de generar ***mensajes automáticamente*** para de este modo garantizar que siempre hay tráfico entre los nodos de la simulación. El contenido de estos mensajes los recuperará de la base de datos.

4.3 Modelo de datos

Una vez realizado el análisis de los datos, los modelizaremos usando el modelo conceptual Entidad-Relación

4.3.1 Modelo Entidad-Relación



4.3.2 Definición de las entidades

Nombre	Profesores
Objeto	Almacenar la información relativa de los profesores autorizados a iniciar una sesión de 'Géminis Server'
Cardinalidad	10 profesores
Crecimiento	Poca variación
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS[4] deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	Como identificador del profesor, almacenaremos el mismo identificador que usa para autenticarse en el servidor LDAP

Nombre	Alumnos
Objeto	Almacenar la información relativa de los alumnos autorizados a iniciar una sesión de 'Géminis Server'
Cardinalidad	500
Crecimiento	Poca variabilidad
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	Como identificador del alumno, almacenaremos el mismo identificador que usa para autenticarse en el servidor LDAP

Nombre	Grupos
Objeto	Enumerar grupos de alumnos
Cardinalidad	25
Crecimiento	Poca variabilidad
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	

[4] RDBMS: Relational Database Management Sistem (Sistema Gestor de bases de datos relacionales)

Nombre	Registro
Objeto	Registrar la actividad de los alumnos en la simulación
Cardinalidad	6000 (25 grupos x 20 alumnos x 12 semanas)
Crecimiento	6000 registros cada cuatrimestre
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	

Nombre	NetDefinition
Objeto	Almacenar las distintas definiciones de red que serán usadas en las distinta sesiones iniciadas por los profesores
Cardinalidad	5
Crecimiento	2 semestralmente
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	El formato en que se almacena la información referente a la definición de una red se detalla en el anexo 1

Nombre	AutoMessage
Objeto	Almacenar los mensajes que el sistema generará automáticamente
Cardinalidad	1500
Crecimiento	Actualización anual, no crece
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	

Nombre	ParametrosServidor
Objeto	Almacenar los parámetros de configuración del servidor
Cardinalidad	10
Crecimiento	No crecen
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	Consultar en el Anexo 2, el detalle de los parámetros configurables en el servidor

Nombre	ParametrosCliente
Objeto	Almacenar los parámetros de configuración del cliente
Cardinalidad	5
Crecimiento	No crecen
Derechos de Acceso	Para garantizar la confidencialidad de esta entidad, el RDBMS deberá solicitar un usuario y una contraseña para visualizar los elementos de la misma.
Observaciones	Consultar en el Anexo 2, el detalle de los parámetros configurables en el servidor

4.3.3 Traducción del modelo Entidad-Relación

```
CREATE TABLE Profesores(
        Id              VARCHAR(50),
        Nombre          VARCHAR(150),
        PRIMARY KEY(Id))

CREATE TABLE Grupos(
        Id              NUMERIC(2),
        Descripción     VARCHAR(50),
        Profesor        VARCHAR(50)     REFERENCES Profesores(Id),
        PRIMARY KEY(Id))

CREATE TABLE Alumnos(
        Id              VARCHAR(50),
        Nombre          VARCHAR(150),
        Grupo           NUMERIC(2)      REFERENCES Grupo(Id),
        PRIMARY KEY(Id))

CREATE TABLE Registro(
        UserID          VARCHAR(50)     REFERENCES Alumnos(Id),
        Login           DATE,
        Logout          DATE)
```

```
CREATE TABLE ParametrosServidor(
        ParamName VARCHAR(50),
        ParamValue VARCHAR(150),
        Description  VARCHAR(150)
        PRIMARY KEY (ParamName))

CREATE TABLE ParametrosCliente(
        ParamName VARCHAR(50),
        ParamValue VARCHAR(150),
        Description  VARCHAR(150)
        PRIMARY KEY (ParamName))

CREATE TABLE AutoMessage(
        MSG         VARCHAR(255))

CREATE TABLE NetDefinition(
        Id          NUMERIC(3),
        Param       VARCHAR(50),
        Value       VARCHAR(255))
```

4.4 Diseño Lógico de la aplicación

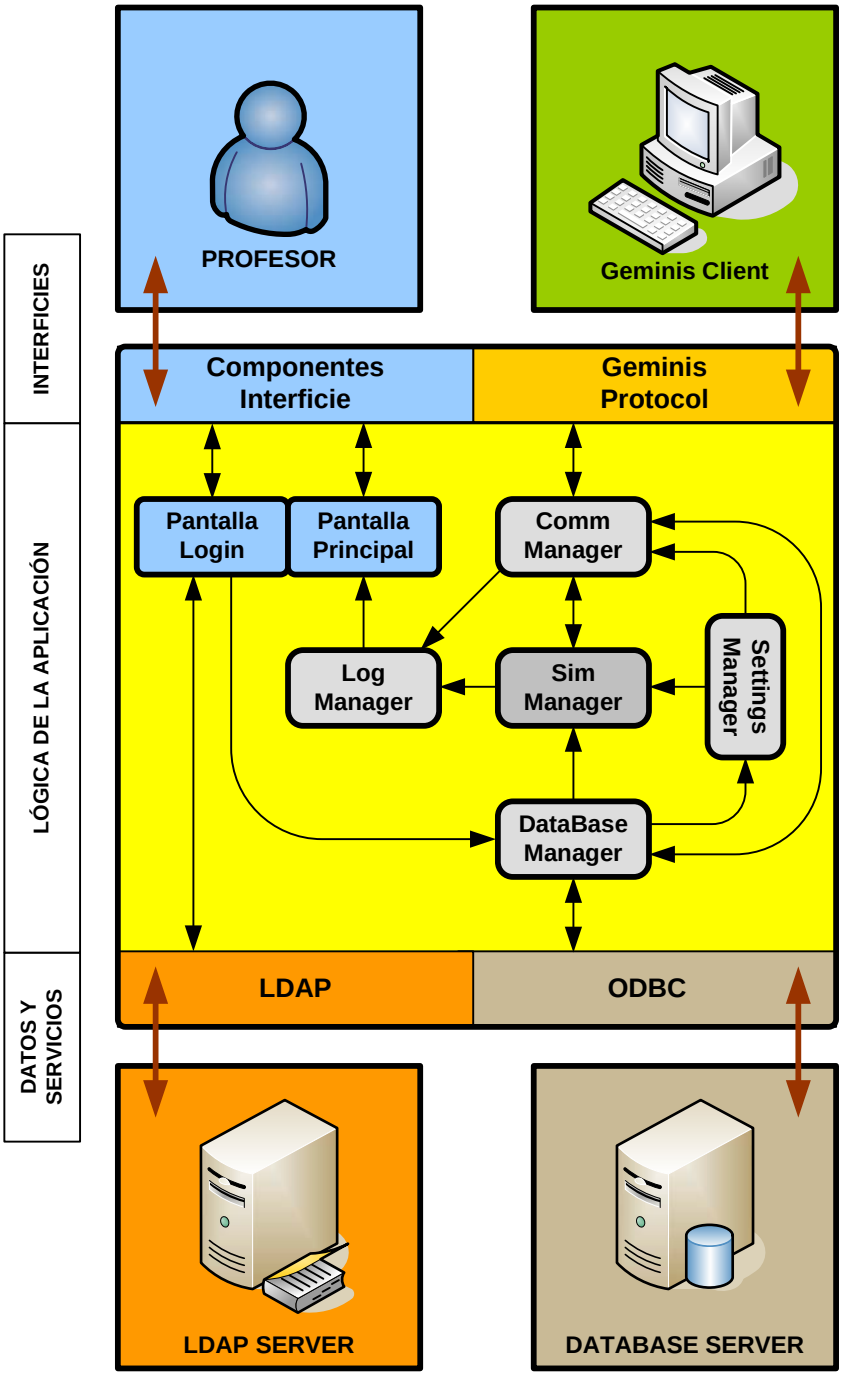
La arquitectura usada para implementar 'Géminis Server' se centra en el modelo de 3 capas.

La capa de interficie externa, esta formada por una parte, por los Componentes de Interficie de Usuario (IU), y los componentes de proceso de IU. Los componentes de IU son la parte con la cual interactuar con el usuario, en este caso el profesor. Los componentes de proceso de IU podríamos asociarlos a clases de tipo controladora en UML. Es decir estos encapsulan lógica de navegación y control de eventos de la interfase. Hemos añadido en esta capa también los componentes de Interficie de Cliente (IC), y los componentes de proceso de IC ya que nuestra aplicación también interactúa con un usuario remoto el cual usara el protocolo definido para este caso ('Géminis Protocol').

La capa de lógica de la aplicación, contiene aquellas clases que directamente implementan el objetivo de nuestro proyecto: un simulador. También incluirá otras clases que se usarán como soporte las cuales iremos especializando según el tipo de operaciones a realizar.

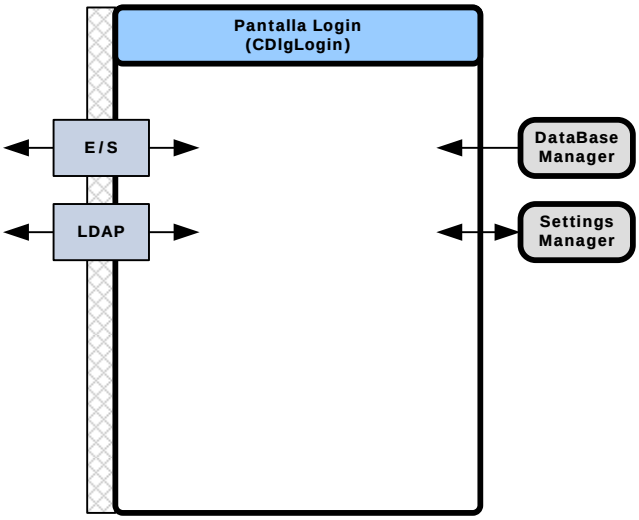
La capa de acceso a datos que contiene clases que interactúan con la base de datos. Estas clases surgen como una necesidad de mantener la cohesión o clases altamente especializadas que ayuden a reducir la dependencia entre las clases y capas. Aquí podemos encontrar también una clase con métodos estáticos que permiten uniformizar las operaciones de acceso a datos a través de un único conjunto de métodos.

4.4.1. Diagrama de flujo de datos



4.4.2 Módulos de la aplicación

Pantalla Login



Funcionalidad

Esta es la primera pantalla que aparecerá al arrancar la aplicación. Su función es la de pedir las credenciales del usuario, en este caso, del profesor. Estas credenciales serán las mismas que las usadas en cualquier aplicación donde la gestión de las contraseñas está centralizada en el directorio de servicios LDAP de la facultad.

Una vez obtenidas las credenciales del usuario se realizaran las siguientes acciones:

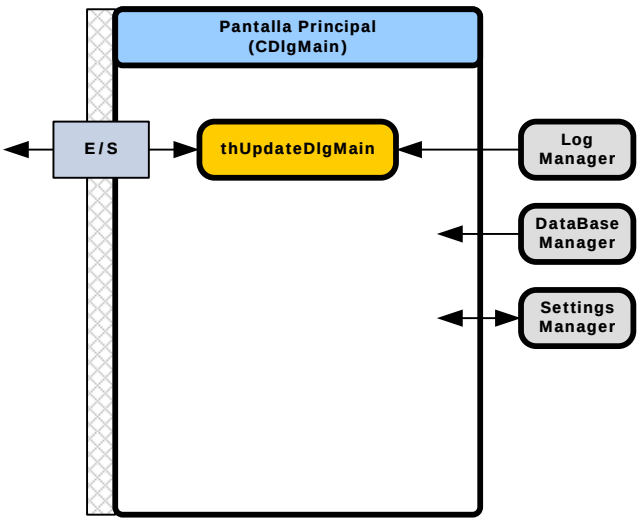
- Validar el usuario y password en el árbol de directorios LDAP. La información sobre la localización de este servicio de red, será suministrada por el módulo 'Settings Manager'.

- Comprobar que el usuario está dado de alta en la base de datos de la aplicación, para ello consultaremos la información proporcionada por el módulo 'Database Manager'
- Si el usuario está validado en el directorio LDAP y está dado de alta en la BBDD de la aplicación, se procederá a registrar su identificador mediante el módulo 'Settings Manager'. En caso contrario se pide de nuevo las credenciales al usuario. A la tercera validación fallida se terminará la aplicación.

Relación con otros módulos

- Database Manager
    - Recibe: Consulta existencia del usuario en la BBDD
- Settings Manager
    - Recibe: Parámetros para localizar servicio LDAP
    - Envía: identificador del usuario validado

Pantalla Principal



Funcionalidad

Esta pantalla es la principal interficie de usuario entre el servidor y el profesor, permite realizar las siguientes tareas:

- Operaciones sobre el servidor
  - Modificar puerto de escucha de servidor
  - Seleccionar profesor responsable de la simulación
  - Seleccionar grupo autorizado a iniciar sesión en la simulación
  - Seleccionar la red que queremos usar en la simulación

- Operaciones sobre la simulación
  - Especificar el tiempo medio entre mensajes automáticos

  - Especificar el número de pulsos del reloj de la simulación por segundo
  - Iniciar y detener la simulación
- Operaciones sobre la información de log de los módulos
  - Especificar nivel de detalle del log
  - Detener y mostrar log generado
  - Borrar log mostrado en pantalla

Procesos

Este módulo mientras está la pantalla visible ejecuta el proceso 'thUpdate DlgMain', se trata de un thread que se encarga de actualizar la información que podemos ver en la pantalla, en concreto el log y informaciones sobre el tiempo que lleva ejecutándose, así como el número de clientes conectados y el número de mensajes generados en la simulación.

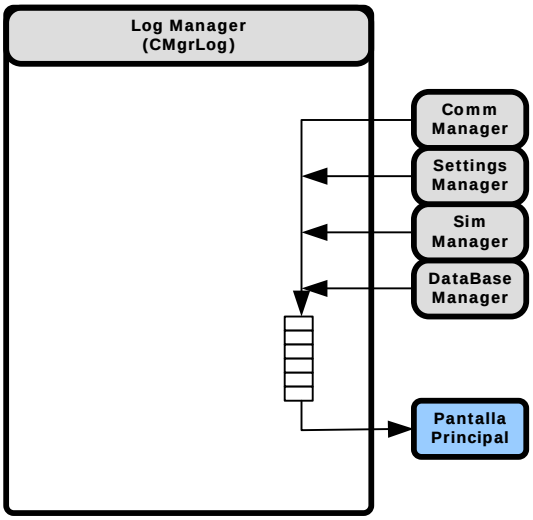
Relación con otros módulos

- Database Manager
  - Recibe: información sobre redes definidas en la bbdd, lista de grupos disponibles, lista de profesores dados de alta.
- Log Manager
  - Recibe: información de log generada por otros módulos de la aplicación
- Settings Manager
  - Recibe: Información sobre valores de los parámetros: tiempo medio entre mensajes, números de pulsos de simulación por

segundo, nivel de log por defecto, puerto del servidor por defecto, etc

- Envía: Aquellos parámetros que han sido modificados por el usuario

Log Manager



Funcionalidad

Este módulo es el encargado de recoger información sobre el funcionamiento de otros módulos de la aplicación. Puede recibir información de error, cuando se produce un error en algún punto del programa o bien recibir información sobre sucesos que ocurren en el sistema. Los sucesos informativos se generan asociados a un nivel de importancia, el cual se usará al mostrar al usuario dependiendo de la cantidad de información que este interesado en visualizar. Se han definido 4 niveles.

Estructura de Datos

- Buffer de mensajes.

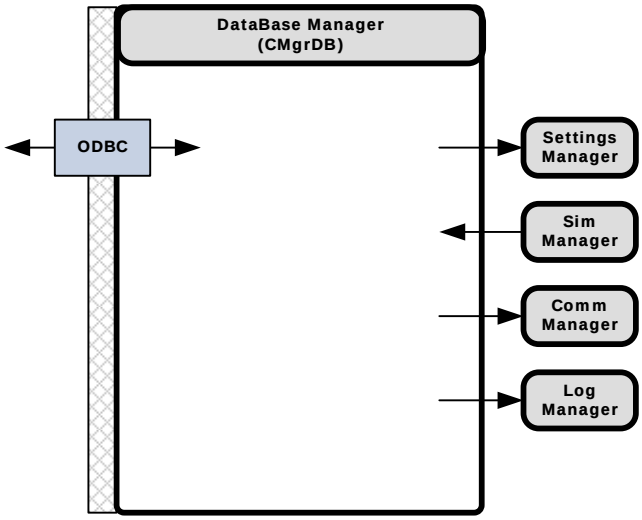
  Todos los módulos al enviar información, la depositarán en un buffer, la información del cual será leída por el proceso que actualiza la pantalla principal a intervalos regulares. El tamaño del buffer está

limitado a la memoria disponible en la máquina en la que se esté ejecutando la aplicación.

Relación con otros módulos

- Settings Manager
    - Recibe: El nivel de log que se mostrará por defecto
- Database Manager
    - Recibe: Recibe mensajes informativos y de error
- Simulation Manager
    - Recibe: Recibe mensajes informativos y de error
- Communicartions Manager
    - Recibe: Recibe mensajes informativos y de error
- Pantalla Principal
    - Envía: los mensajes recibidos de otros módulos

Database Manager



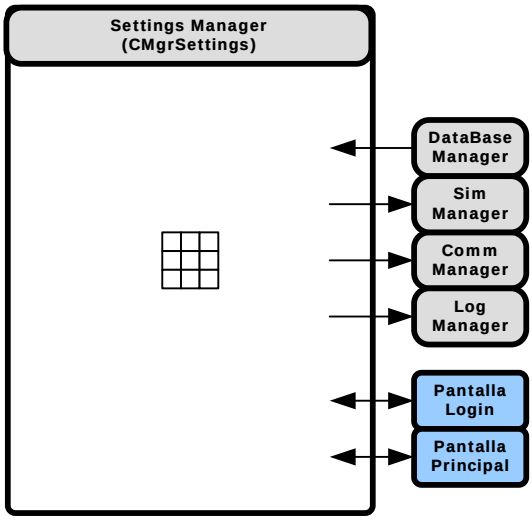
Funcionalidad

Este módulo, encapsula el acceso a los datos y ofrece servicios de acceso a los módulos que lo necesiten. A su vez este módulo usa el protocolo ODBC (Open Database Connectivity) para poder acceder a los datos independientemente del RDBMS que esté gestionando dichos datos (Access, MS SQL Server, Oracle, DB/2, etc). La localización de los datos, será suministrada por el componente 'Settings Manager', el cual le suministrará un nombre DSN para acceder a los datos (por defecto GeminisData) o bien la cadena de conexión a los datos.

Relación con otros módulos

- Communications Manager
    - Envía: Información de parámetros de cliente, como respuesta a peticiones realizadas por los clientes
    - Recibe: credenciales de usuarios que han entrado o salido de la simulación, para registrar su actividad.
- Simulation Manager
    - Envía: La definición de la red que quiere cargar el simulador
- Settings Manager
    - Recibe: Parámetros para localizar servicios de datos
- Log Manager
    - Envía: información sobre los procesos ejecutados por el módulo

Settings Manager



Funcionalidad

Este modulo tiene como finalidad, intercambiar información de estado entre los distintos módulo. Al iniciarse carga los valores de algunas propiedades de la base de datos y también según los parámetros de la línea de comandos. El resto de módulos, podrán acceder a estos parámetros o bien modificarlos según sus necesidades.

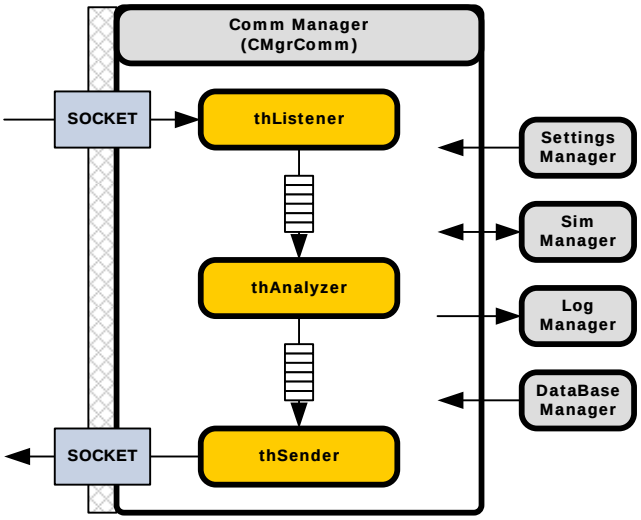
Estructura de Datos

Tenemos una tabla con información de los valores que toman cada uno de los parámetros que leen o escriben cada uno de los módulos, y que se inicializan según información cargada de la base de datos y bien por la línea de comandos.

Relación con otros módulos

- Database Manager
    - Recibe: Lee de la BBDD los parámetros por defecto de los componentes de la aplicación
- Simulation Manager
    - Envía: Información sobre valor de parámetros como: tiempo medio entre mensajes, número de pulsos de simulación por segundo
- Communication Manager
    - Envía: Información sobre valor de parámetros como: puerto por defecto del servidor y de los clientes
- Log Manager
    - Envía: Información sobre valor de parámetros como: Nivel de log a mostrar por defecto
- Pantalla de Login
    - Envía: información sobre localización de servicios LDAP
    - Recibe: identificador del usuario validado
- Pantalla principal
    - Envía: Información sobre el valor de varios parámetros
    - Recibe: modificaciones realizadas por el usuario de los parámetros mostrados

Communications Manager



Funcionalidad

La función de este módulo, es la de encapsular la comunicación TCP/IP entre el servidor y los clientes, codificación el intercambio de mensajes según el protocolo 'Géminis Protocol' que se detalla en el apartado 3.5.

Estructura de Datos

Para gestionar los mensajes que entran y salen del servidor, tenemos dos colas de tipo FIFO llamadas 'inputBuffer' y 'outputBuffer' respectivamente, en las que vamos almacenando dicha información. El tamaño de estas colas está limitado por la cantidad de memoria disponible en la máquina en la que se ejecuta el servidor.

Procesos

Este modulo es junto con el propio módulo de simulación de los más críticos del sistema, ya que la comunicación entre los clientes ha de ser lo más fluida posible, ya que nuestra intención es que todos ellos visualicen simultáneamente el estado de la simulación, lo que requiere el envío de una gran cantidad de información a distintos clientes y la recepción de las acciones que los alumnos realizan sobre la red que simulamos. Es por eso que este módulo usa tres 'Threads' para agilizar la comunicación entre cliente y servidor y el proceso de la información recibida, a continuación detallamos cada uno de los procesos usados:

- thListener. Es el proceso encargado de recibir todos los mensajes que llegan de los clientes. A medida que van llegando los va encolando en la cola 'inputBuffer' para que sean tratados posteriormente por el proceso 'thAnalyzer'.

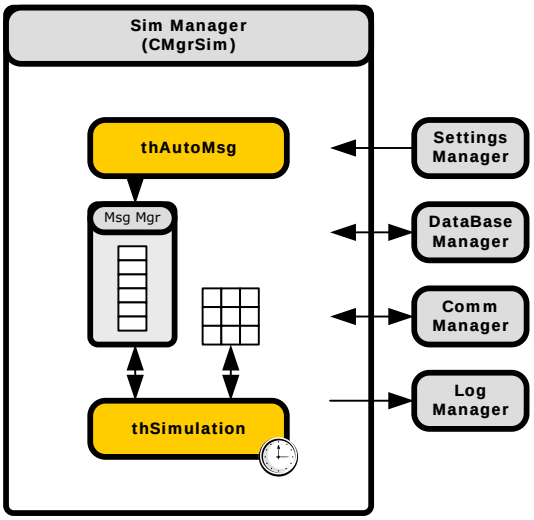
- thAnalyzer. Es el proceso encargado de recoger del buffer de entrada de mensajes 'inputBuffer' los mensajes, analizarlos, verificar que cumplen con la definición del protocolo 'Géminis Protocol' y procesarlos. En caso de tener que enviar una respuesta, esta se dejara en el buffer de mensajes de salida 'outputBuffer' y será enviado posteriormente por el proceso 'thSender'

- thSender. Es el proceso encargado de enviar a los clientes, los mensajes que otros procesos han dejado en la cola de mensajes de salida 'outputBuffer'.

Relación con otros módulos

- Database Manager
    - Recibe: Consulta existencia del usuario en la BBDD cuando recibe una petición de login de un cliente y consulta el valor de un parámetro pedido por un cliente.
- Settings Manager
    - Recibe: Parámetros referentes a características del servidor
- Log Manager
    - Envía: información sobre los procesos ejecutados por el módulo
- Simulation Manager
    - Envía: eventos a la simulación, según mensajes recibidos de los usuarios
    - Recibe: estado de la red y notificación de nuevos mensajes creados, para enviar los correspondientes mensajes a los clientes.

Simulation Manager



Funcionalidad

Este módulo es el núcleo de la aplicación, ya que es el encargado de implementar nuestro modelo, mantener su estado y actualizarlo a intervalos regulares, según vaya dictando el reloj de la simulación.

Estructura de Datos

El componente llamado 'Message Manager' (CMsgMgr) será el encargado de encapsulamos toda la gestión de mensajes y fragmentos.

La gestión de nodos la realizará con una matriz de 255 entradas, por lo que este número será el máximo número de nodos que se pueden definir en la simulación. Es posible añadir nuevos nodos durante la ejecución de la simulación

La gestión de vértices la realizaremos con una matriz de 255 x 255 entradas. Es posible añadir nuevos nodos durante la simulación.

Procesos

El modulo de simulación junto con el encargado de las comunicaciones con los clientes, son considerados críticos dentro de la aplicación, es por este motivo que se han creado dos 'threads' encargados de ejecutar tareas relacionadas con el modulo de simulación:

- thAutoMsg. Es el proceso encargado de generar mensajes automáticos (previamente leídos de la base de datos) entre dos nodos de la red. La frecuencia de generación de estos mensajes viene determinada por un parámetro que puede ser modificado por el profesor antes de iniciar la simulación o durante la ejecución de la misma.

- thSimulation. Es el proceso encargado de actualizar el estado de la simulación a intervalos regulares que vienen marcados por el *reloj de la simulación*, cuya frecuencia podrá aumentar o disminuir el profesor según las necesidades de cada momento.

  A cada paso de reloj, se actualiza el estado de la simulación siguiendo los siguientes pasos detallados en el apartado 'Simulación', del capitulo 4.4.3.

Relación con otros módulos

- Log Manager
    - Envía: información sobre los procesos ejecutados por el módulo
- Comm Manager
    - Envía: el estado de la red y notifica de la creación de nuevos mensajes a los clientes que están conectados a la simulación.
    - Recibe: eventos que modifican el estado de la simulación que han sido provocados por los clientes.
- Database Manager
    - Recibe: La definición de la simulación, así como el contenido de los mensajes automáticos que generará el sistema.
    - Envía: información de los usuarios asociados a un nodo, así como registra cuando un usuario abandona un nodo.
- Settings Manager
    - Recibe: Parámetros para localizar servicio LDAP

4.4.3 Implementación Red Géminis

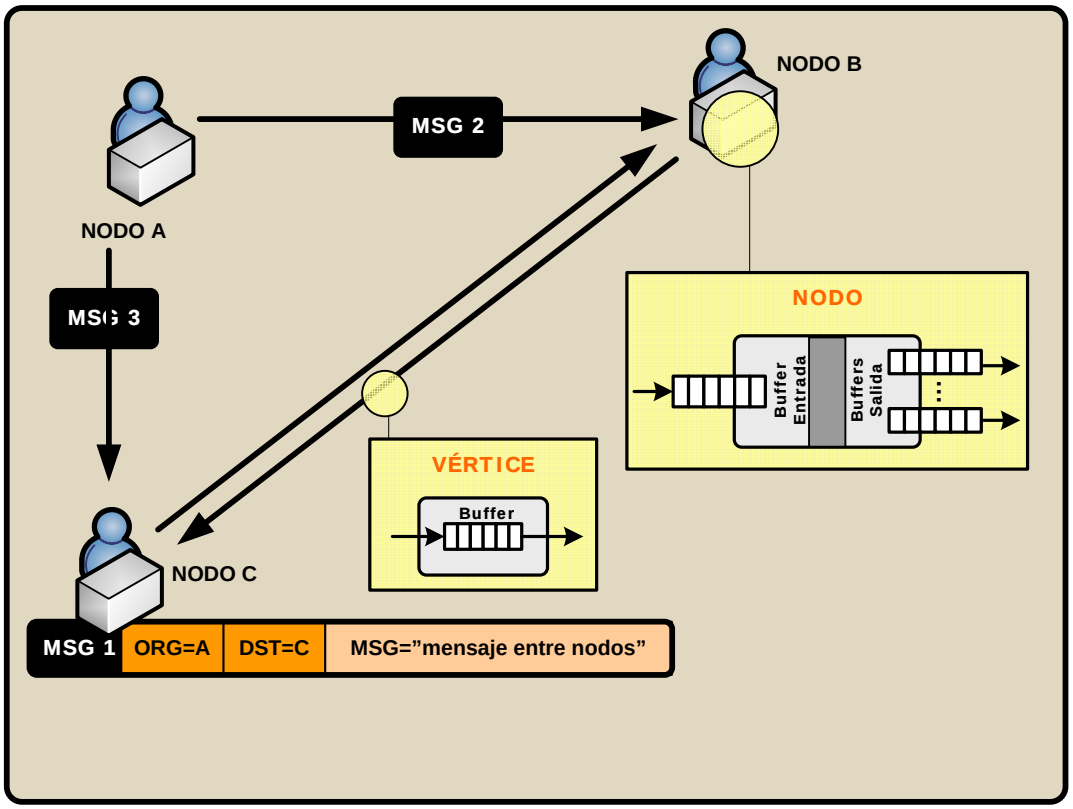


Ilustración 29 . Detalles de implementació de 'Red Géminis'

A continuación vamos a detallar como se han implementado los distintos elementos que forman la 'Red Géminis'. Al implementar la aplicación con un lenguaje orientado a objetos, se ha aprovechado para encapsular cada elemento de la red en una clase distinta, definiendo sus propiedades. Las operaciones entre los elementos de la red, se realizarán a través del 'Sim Manager', que es el encargado de manipular todos los objetos que modelizan la red.

Mensaje

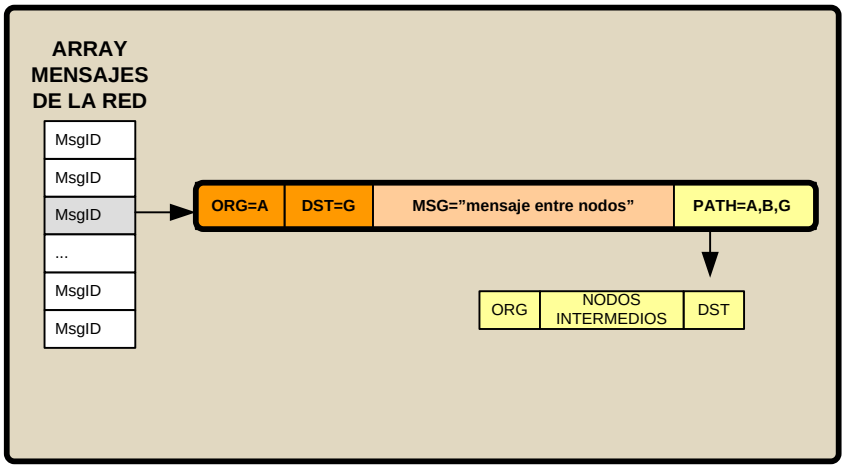


Ilustración 30 . Implementación de un mensaje

Un mensaje está implementado como una clase con las siguientes propiedades:

- *MsgId*: Identificador de mensaje. Cada mensaje de la simulación tiene un identificador único generado secuencialmente.Todos los mensajes están almacenados en un array dinámico al que se accede por este Id.
- *Origen*: Identificador del nodo donde se originó el mensaje
- *Destino*: Identificador del nodo donde se originó el mensaje
- *Ruta*: Al crear el mensaje se calcula su ruta, la cual es una lista de nodos.

Todos los mensajes creados durante la simulación, se almacenarán en el servidor de modo que estará disponible su información a petición de cualquier cliente de la simulación.

Colas de Fragmentos

Todos los mensajes que viajan entre los distintos elementos de la red, son divididos en fragmentos de distintos tamaños, dependiendo de la capacidad del vértice que han de atravesar. Posteriormente estos fragmentos pueden ser reagrupados para formar uno mayor, siendo el fragmento máximo, el propio mensaje.

Nodos y vértices, usan colas de fragmentos para almacenar aquellas partes de mensajes que no han podido ser aun transmitidas o están en curso. Estas colas de fragmentos son idénticas para ambos elementos, con la única diferencia que para las colas de entrada y salida de nodos, su tamaño es ilimitado (dependiendo de la memoria disponible en el ordenador) y para vértices, el tamaño de dicha cola está limitado al ancho de banda con que se ha definido.

Nodo

Un nodo está identificado por su nombre o por la posición que ocupa en la tabla en que se almacenan en el servidor. Esta tabla es tiene 255 entradas, lo que establece el límite de nodos en la simulación.

Para implementar un nodo de la 'Red Géminis', hemos usado las colas de fragmentos vistas anteriormente, de modo que cada nodo tendrá:

- *una cola de entrada*: en la que cualquier otro vértice que tenga como destino dicho modo, dejará en esta cola todo los fragmentos que vayan pasando por el.

- *Varias colas de salida*: tantas colas como vértices tengan como origen dicho nodo. En estas colas se almacenaran aquellos fragmentos que no han podido aun pasar por el vértice.

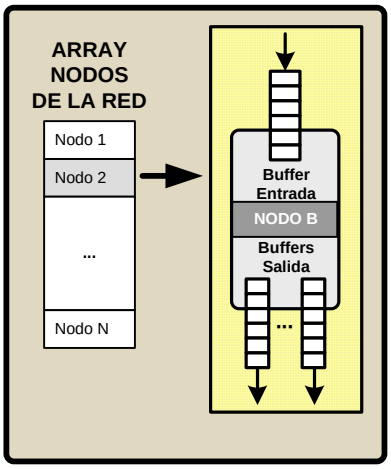


Ilustración 31 . Implementación de un nodo

Vértice

Un vértice está identificado por la posición que ocupa en la matriz de vértices, cuya dimensión es de 255x255.

Implementaremos un vértice como una cola de fragmentos, que a diferencia de las colas de fragmentos de los nodos, está limitada su tamaño por el ancho de banda del vértice.

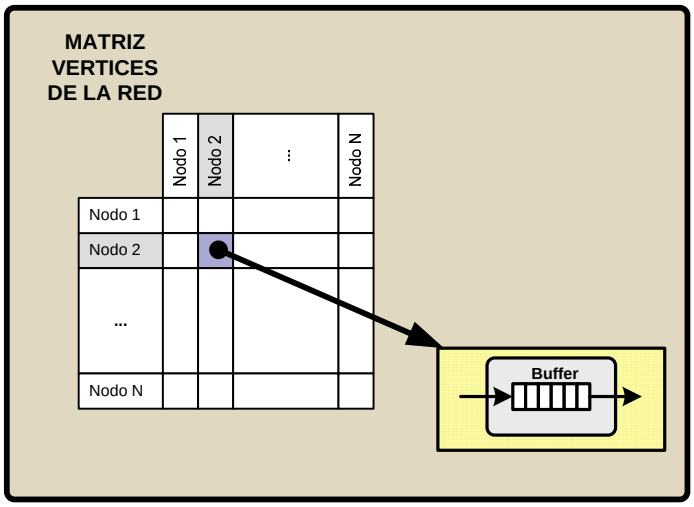
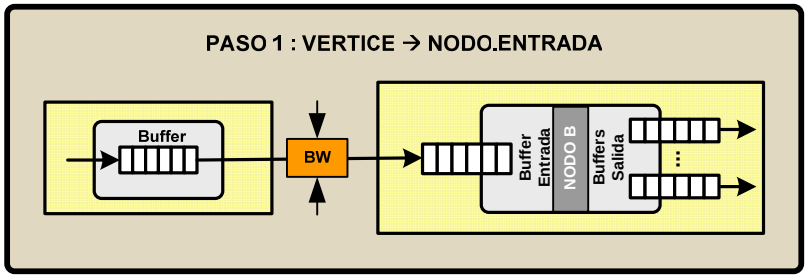


Ilustración 32 . Implementación de un vértice

Simulación

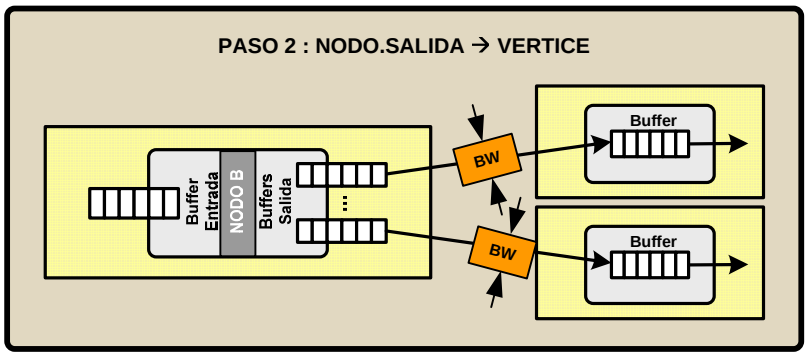
A continuación detallamos los pasos realizados a cada impulso de reloj de la simulación:

Paso 1: Vértice → Nodo.Entrada



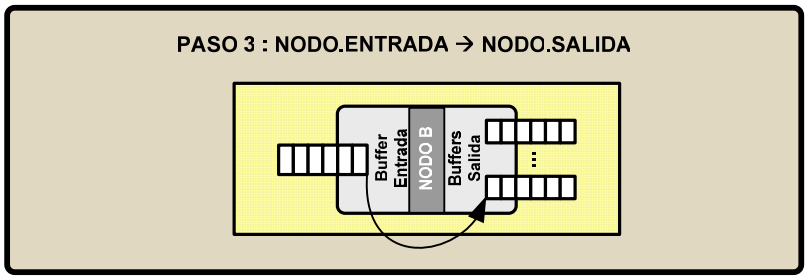
Todos los fragmentos que hay en un vértice, se pasan a la cola de entrada del nodo al que está conectado su destino, de este modo se vacían los vértices para poder dejar paso a nuevos fragmentos de mensajes.

Paso 2: Nodo.Salida → Vértice



Para cada nodo de salida de cada vértice se, pasan tantos fragmentos de mensajes necesarios, hasta que el tamaño de las colas de fragmentos de los vértices tengan como máximo el mismo número de bytes que el ancho de banda con que se han definido.

Paso 3: Nodo.Entrada → Nodo.Salida



Por último aquellos fragmentos de una cola de entrada que agrupados forman el mensaje original, se pasan a la cola de salida según el siguiente paso que viene determinado por su ruta.

4.5 Diseño externo de la aplicación

La aplicación 'Géminis Server' es básicamente un servicio que se ejecuta en un punto de red y que atiende peticiones de varios clientes, por lo que su interficie grafica no es lo más destacado, no obstante se ha tratado de realizar un cuidadoso diseño de las pantallas para facilitar la configuración usando componentes visuales que eviten cometer errores en la entrada de datos, por ejemplo, en lugar de que la entrada del grupo con el que trabajaremos, sea un texto libre que luego hay que validar en la base de datos, se ha extraído previamente esta información y se ha mostrado en forma de lista. De este modo se ha tratado simplificar la entrada de datos y reducir los posibles errores en su introducción.

4.5.1 Diseño de pantallas

Pantalla de Login

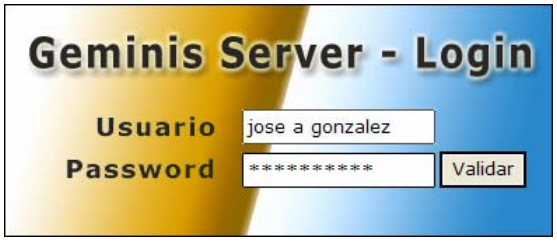


Ilustración 33 . Pantalla de login

La 'pantalla de login', nos permite introducir las credenciales del profesor (usuario y password), las cuales en caso de ser válidas (tanto en el directorio LDAP, como en la base de datos de la aplicación), permitirán dar paso a la 'pantalla principal'.

Pantalla Principal

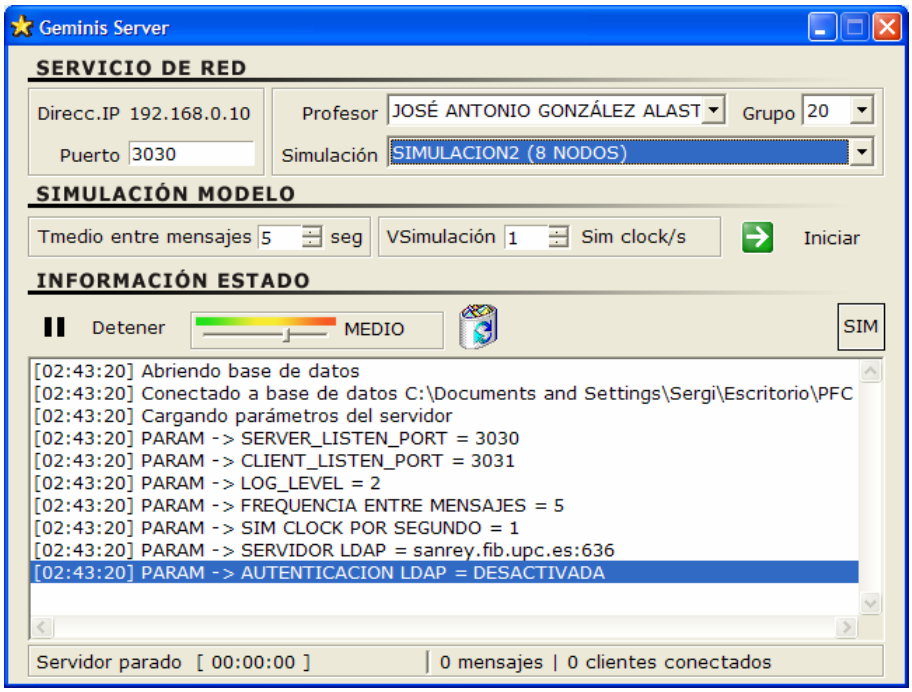


Ilustración 34 . Pantalla principal

La pantalla principal nos permite configurar el servicio que pondremos en funcionamiento. Esta pantalla se ha dividido en 3 partes según su funcionalidad:

- **Servicio de red**. Permite definir los parámetros con los que pondremos en funcionamiento el servicio de red:

  - **Puerto de escucha del servidor**. Puerto al que los clientes enviaran sus mensajes.

  - **Grupo de la sesión**. Grupo de laboratorio al que va dirigido la sesión

- **Simulación**. Definición de red que utilizaremos en la sesión que iniciaremos.

- **Profesor responsable de la sesión**. Profesor que normalmente se encarga de las sesiones de laboratorio de un grupo.

- **Simulación del modelo**. Permite actuar sobre el modelo, variando su estado y características:

  - **Tiempo medio entre mensajes**. Frecuencia media entre los mensajes que el sistema genera automáticamente. El rango oscila entre 1 y 10 segundos entre mensajes.

  - **Velocidad de simulación**. Número de impulsos de simulación generados cada segundo. Su valor por defecto es 1, lo que quiere decir que cada segundo se producirá un impulso de la simulación. Podemos acelerar el ritmo de la simulación, aumentando la velocidad de simulación en un rango entre 1 y 20 impulsos de simulación por segundo.

  - **Iniciar de simulación**. Inicia los procesos de recepción y envío de mensajes y el generador de mensajes automáticos y el de impulsos de la simulación.

  - **Detener simulación.** Detiene los procesos de recepción y envío de mensajes, la generación automática de mensajes y el reloj de la simulación.

- **Información de estado**. Nos muestra información del estado de la simulación y de todos los módulos que dan soporte a su ejecución. Podemos realizar una serie de acciones sobre el log que se genera en el servidor:
    - **Modificar nivel de detalle.**
    - **Detener generación de log**
    - **Reanudar generación de log**
    - **Borrar información histórica de log**

Pantalla información simulación

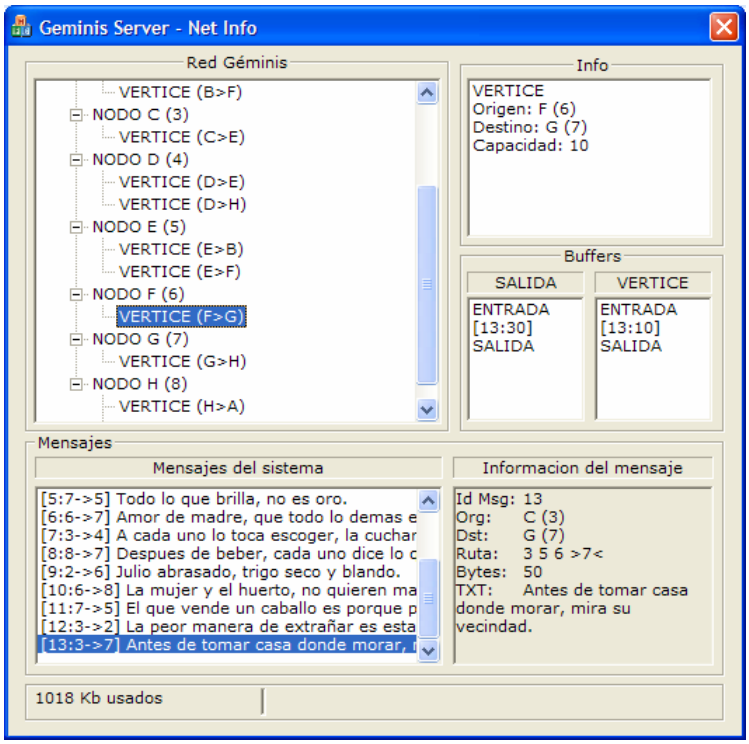


Ilustración 35 . Pantalla información simulación

Esta pantalla sirve para mostrar la actividad que esta teniendo lugar en la simulación. No se trata de una visión visual de la red, ya que este es el objetivo del proyecto 'Géminis Client', simplemente es una manera de poder observar que se están generando mensajes y ver que vértices presentan una mayor saturación. Esta pantalla es una adaptación de la presentada por la aplicación de pruebas 'SimTester' comentada en el apartado 5.3.

5. Evaluación experimental

5.1 Introducción

Durante el proceso de desarrollo, se han ido haciendo pruebas exhaustivas de los distintos módulos por separado a medida que se iban desarrollando, dando una mayor importancia a las pruebas realizadas en los módulos de comunicación y simulación, ya que son la base de nuestra aplicación. Al trabajar con un lenguaje como C++ que se caracteriza por ser orientado a objetos y modular, nos ha facilitado la prueba de los distintos módulos.

Para ello en el desarrollo de cada uno de los módulos se ha seguido una metodología similar: en primer lugar hemos definido la clase a implementar, durante la implementación se han creado pequeños juegos de pruebas para probar todos los procedimientos de las clases, paso de parámetros, mensajes de error, etc. Integrando finalmente el módulo en el desarrollo del servidor y extendiéndolo posteriormente según nuestras necesidades.

A pesar de que se han realizado pruebas con cada, en este documento solamente detallaremos las realizadas sobre los módulos de comunicaciones y simulación, por tratarse de las más críticas y cuyas pruebas han sido mas amplias y extensivas que el resto de componentes de la aplicación.

5.2 Diseño de pruebas y validación comunicación

Por tratarse de una aplicación que trabaja en un entrono cliente/servidor, es de vital importancia el rendimiento del módulo de comunicaciones (*Comm Manager*), ya que necesitábamos conocer si el servidor sería capaz de dar respuesta a un gran número de peticiones de distintos clientes, en un tiempo razonable, tratando de establecer los límites, ya que si estos límites no encajan en nuestros requerimientos, deberíamos replantearnos el diseño de nuestro módulo.

Para probar las comunicaciones se ha desarrollado un pequeño programa llamado '*NetTester*', cuya finalidad es enviar uno o más paquetes de texto a un servidor y puerto que nosotros decidamos. Como resultado obtendremos el tiempo que se ha tardado en realizar la operación, el número de mensajes ý el número de bytes enviados. Finalmente se nos mostrará el promedio de mensajes y bytes por segundo que ha sido capaz de transmitir el cliente al servidor.

Para el desarrollo del programa de pruebas se ha usado el mismo módulo de comunicaciones que en el servidor, con los mismos procesos (listener, analyzer y sender) por lo que podremos probar el rendimiento tanto al enviar mensajes del cliente al servidor, como al recibir mensajes el servidor del cliente.

Las pruebas se han ejecutado en vario entornos:

Entorno 1

Servidor en máquina A (Pentium III, 800Mhz, 384Mb RAM)

Cliente en máquina A (Pentium III, 800Mhz, 384Mb RAM)

Entorno 2

Servidor en máquina A (Pentium III, 800Mhz, 512Mb RAM)

Cliente en máquina B (Pentium III, 800Mhz, 1024Mb RAM)

Entorno 3

Servidor en máquina A (Pentium III, 800Mhz, 512Mb RAM)

Cliente 1 en máquina B (Pentium III, 800Mhz, 1024Mb RAM)

Cliente 2 en máquina B (Pentium III, 800Mhz, 1024Mb RAM)

A continuación mostramos algunas pantallas obtenidas durante la ejecución de las distintas pruebas:

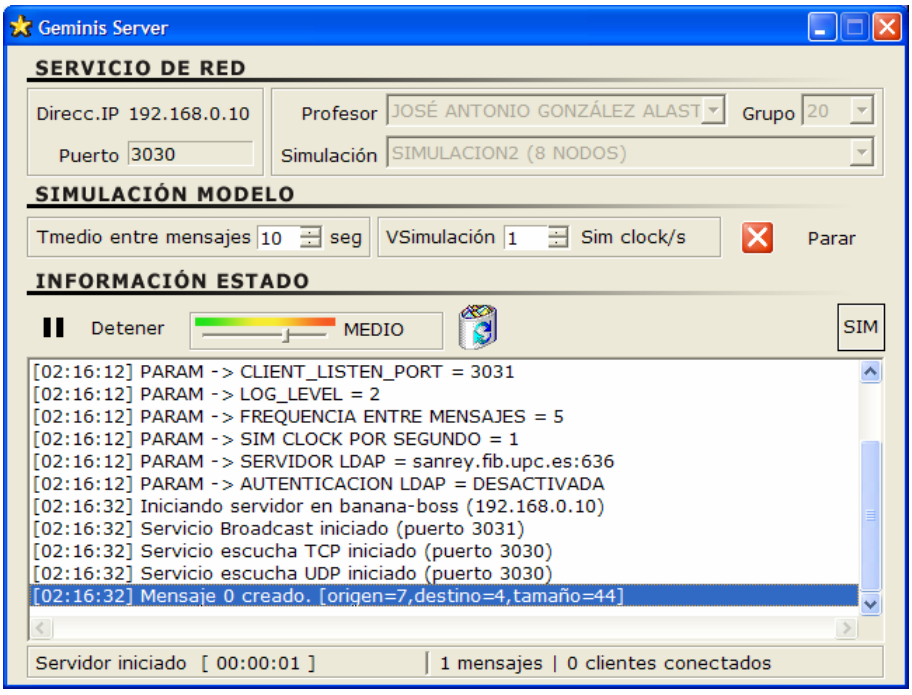


Ilustración 36 . Pantalla del servidor

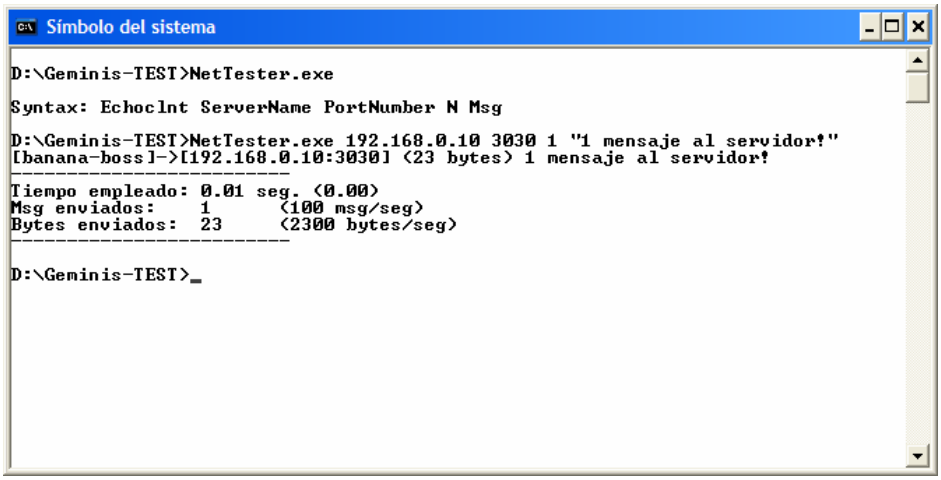


Ilustración 37 . Envío de un mensaje al servidor

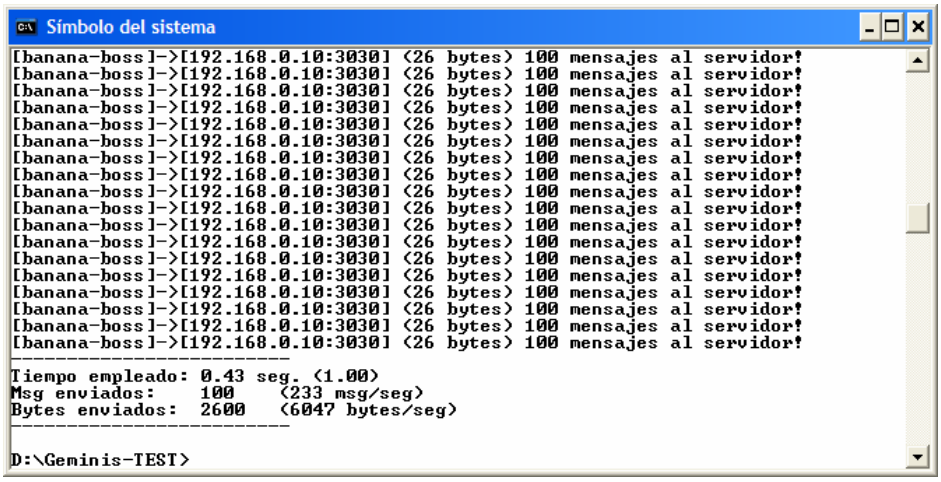


Ilustración 38 . Envío de 100 mensaje al servidor

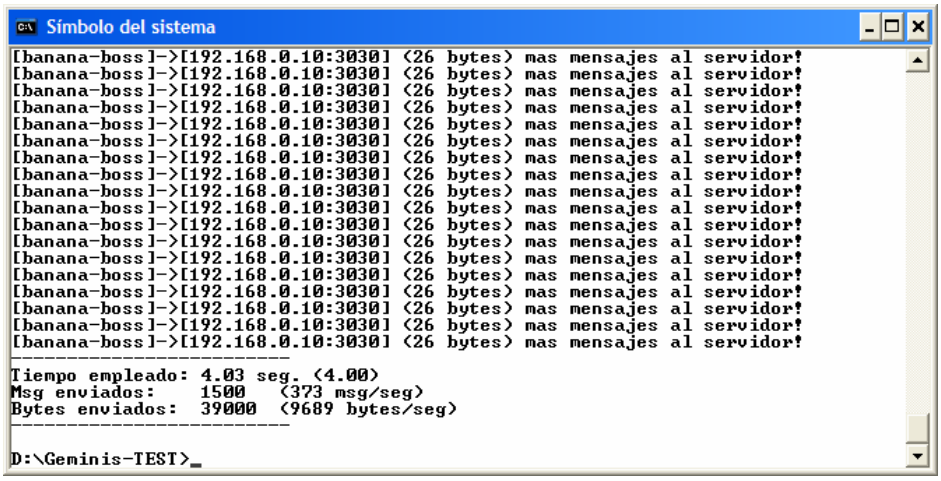


Ilustración 39 . Envío de 1500 mensaje al servidor

En la siguiente tabla, podemos ver los resultados de todas las pruebas que se han realizado en los distintos entornos.

Prueba	Entorno 1	Entorno 2	Entorno 3
Envío 100 msg	233 msg/s	139 msg/s	119,126 msg/s
Envío 1500 msg	373 msg/s	220 msg/s	235,195 msg/s
Envío 3000 msg	453 msg/s	172 msg/s	174,164 msg/s

Tabla 4 . Resultados de pruebas de comunicación

Generación de mensajes por sesión::

Cliente → Servidor

20 alumnos (o parejas) por sesión.

Enviando 2 mensajes por segundo.

El servidor recibe 40 msg/segundo

Servidor → Cliente

Velocidad de simulación máxima (20 impulsos/ segundo), a cada impulso de simulación enviamos estado de red a todos los clientes.

1 mensaje generado automáticamente cada segundo, que hay que notificar a los clientes.

40 mensajes generados por usuarios cada segundo, que hay que notificar a los clientes.

Mensajes de descubrimiento, 1 por segundo.

El cliente recibe 62 msg/segundo

Con los resultados obtenidos, podemos observar que el número de mensajes enviados y recibidos es suficiente para el entorno en el que trabajaremos, y nos da un margen de trabajo suficiente para trabajar en condiciones en que la red este saturada o con máquinas con menos capacidad de procesos que las usadas en las pruebas.

5.3 Diseño de pruebas y validación motor de simulación

Para poder validar el funcionamiento de la simulación, se desarrollo independientemente de la aplicación 'Géminis Server' el módulo de simulación. Al iniciar la aplicación 'SimTester' cargaremos un fichero con la definición de red, que se mostrará por pantalla y nos permite introducir mensajes en el sistema, para posteriormente verificar que se tratan según lo esperado hasta llegar al destino.

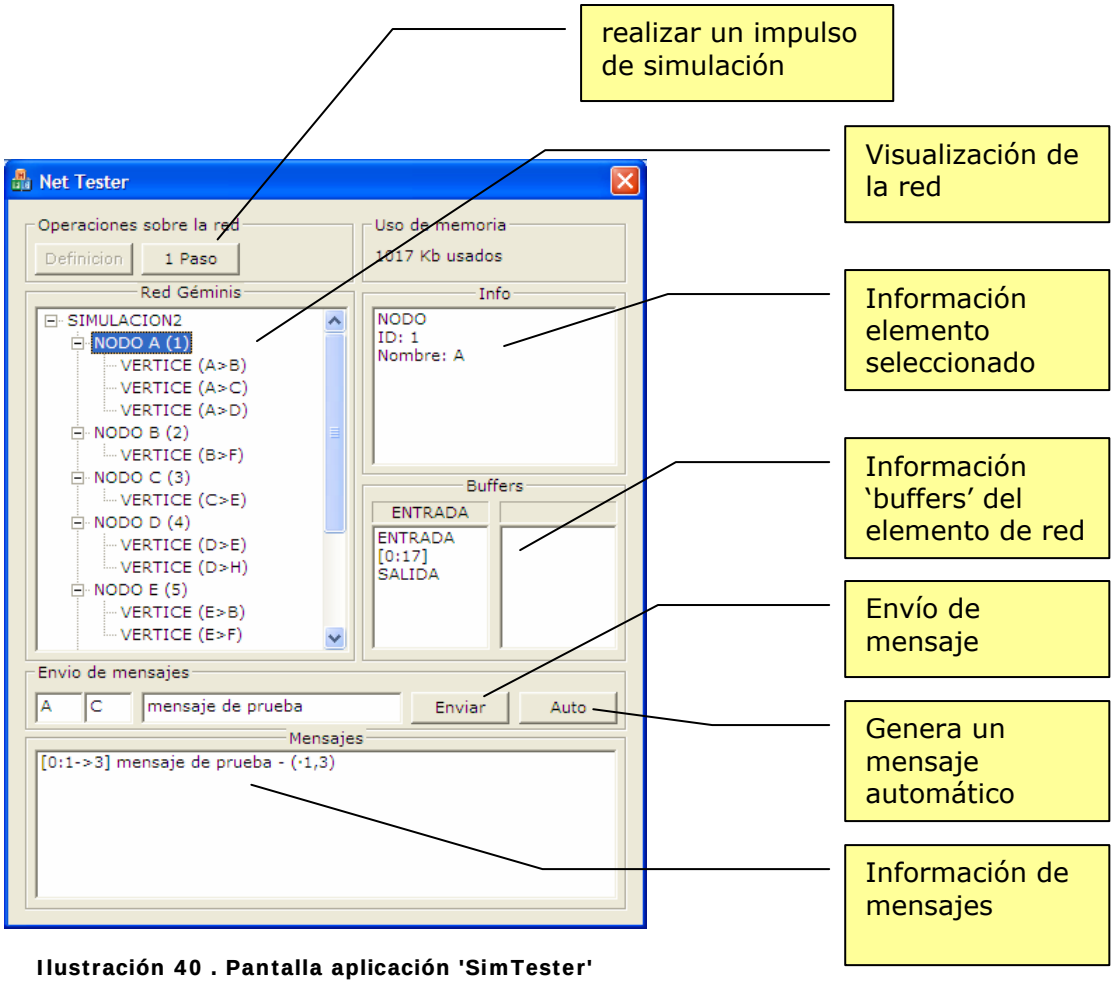


Ilustración 40 . Pantalla aplicación 'SimTester'

Según la red cargada, vamos a simular los pasos a seguir de un mensaje de mas de 10 bytes del nodo A, al nodo G

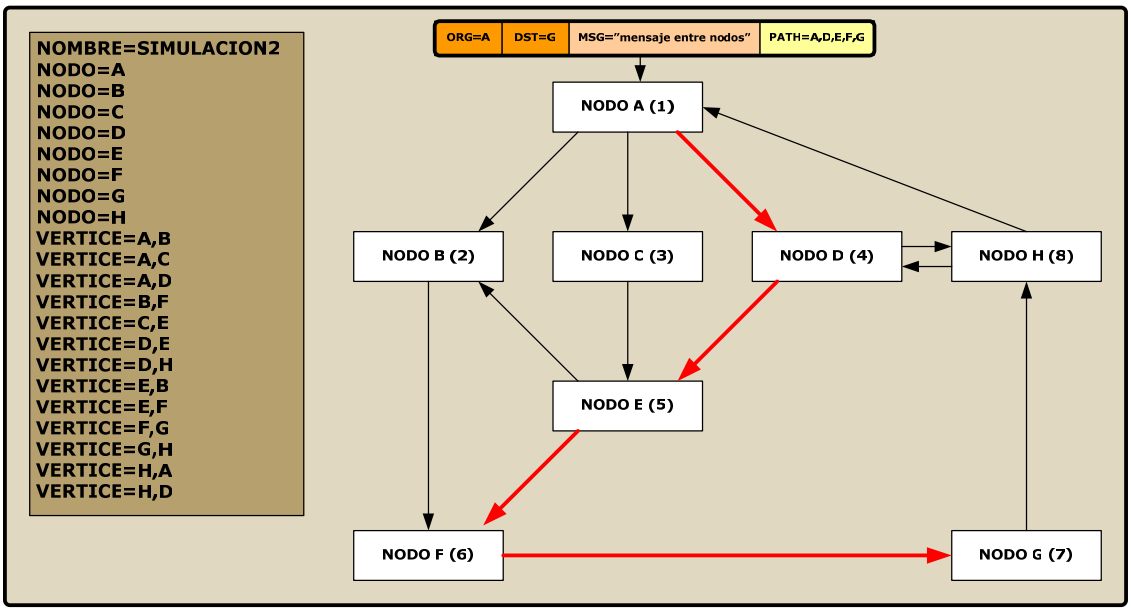
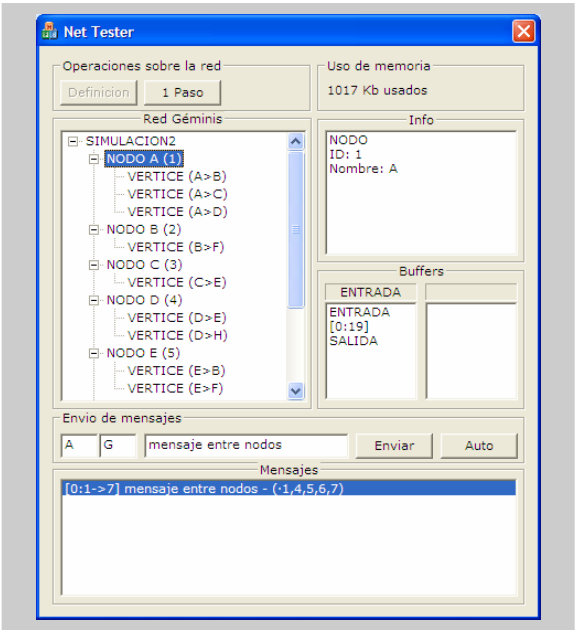
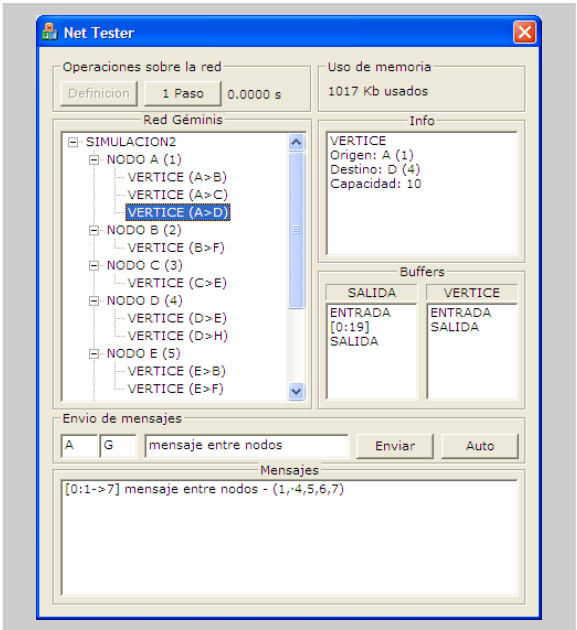


Ilustración 41 . Detalle ejemplo simulación

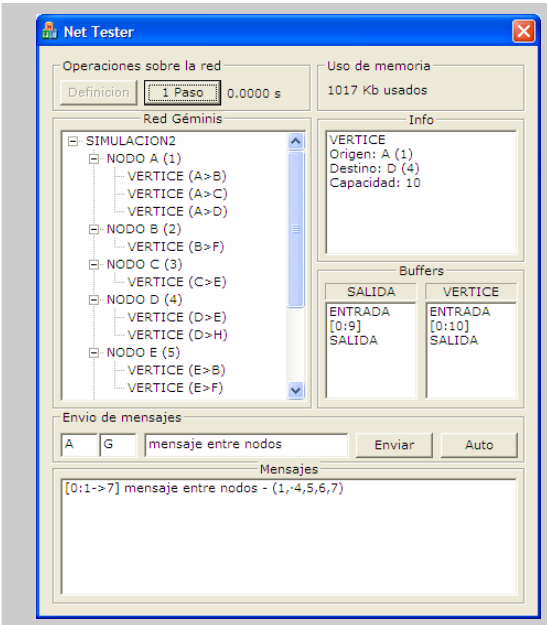
Paso 1: Añadimos mensaje al nodo inicial. Podemos observar que se encuentra en el buffer de entrada del nodo A. con un tamaño de 19 bytes.



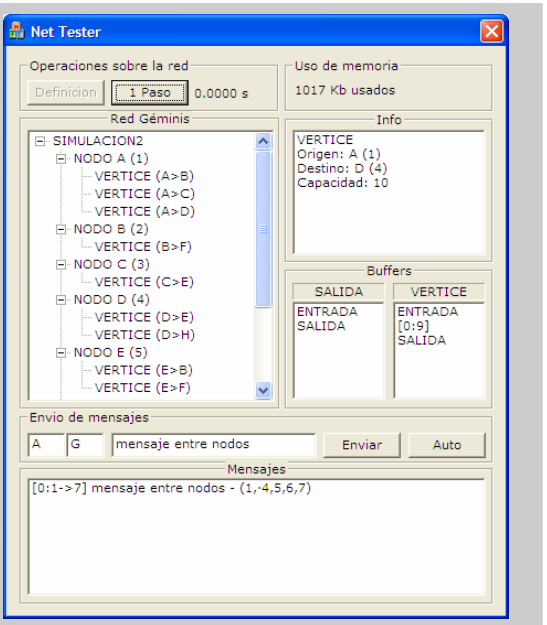
Paso 2: El mensaje pasa al buffer de salida con destino al vértice D



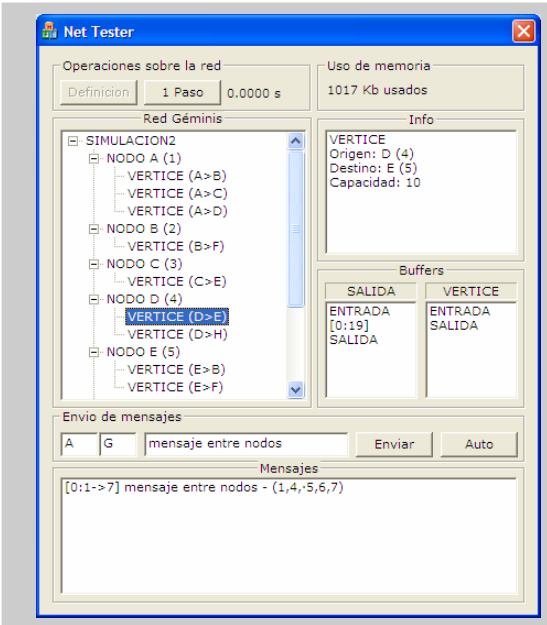
Paso 3: Un fragmento de 10 bytes está en el vértice entre los nodos A y D, el resto de 9 bytes permanece en el buffer de salida del nodo A.



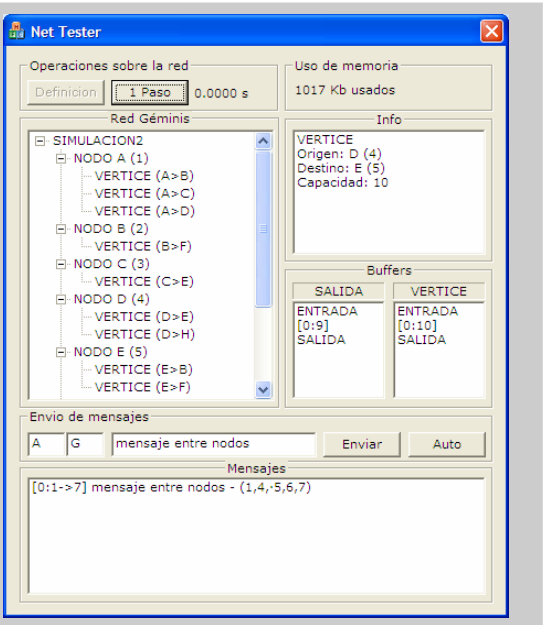
Paso 4: El fragmento que en el paso anterior estaba en el vértice, ha pasado al buffer de entrada del nodo D. El fragmento que quedaba en el buffer de salida, está ahora en el vértice.



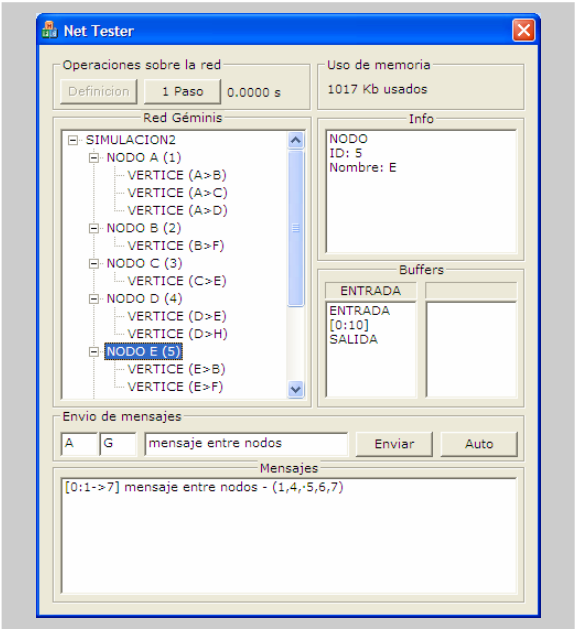
Paso 5: Al llegar todos los fragmentos al buffer de entrada del nodo D, se agrupan y pasan al buffer de salida del siguiente nodo, el nodo E.



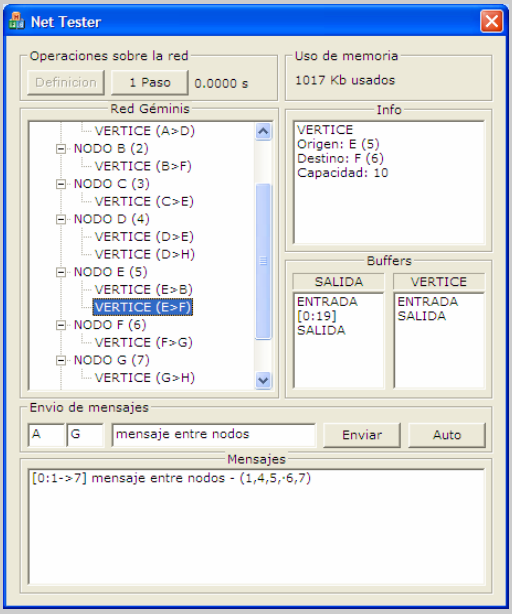
Paso 6: Un fragmento de 10 bytes está en el vértice entre los nodos D y E, el resto de 9 bytes permanece en el buffer de salida del nodo E.



Paso 7: El fragmento que en el paso anterior estaba en el vértice, ha pasado al buffer de entrada del nodo E. El fragmento que quedaba en el buffer de salida, está ahora en el vértice.



Paso 9: Al llegar todos los fragmentos al buffer de entrada del nodo E, se agrupan y pasan al buffer de salida del siguiente nodo, el nodo F.



En los siguientes pasos, el mensaje se comportará del mismo modo que el descrito, hasta llegar a su destino. Podemos observar que el tiempo empleado en cada paso es menor que 1 milisegundo, lo que es un tiempo bastante razonable para nuestros propósitos.

5.4 Impacto en la red

Al poner en marcha el servidor, hemos monitorizado la red, para ver si su ejecución podía llegar a saturar la red local en la que nos encontramos, ya que va transmitiendo de forma periódica mensajes de 'broadcast' para anunciar el servicio y mensajes sobre el estado de la red.

Se ha usado el software 'Optiview' de 'Fluke Networks', el cual nos proporciona una imagen del estado de saturación de la red.

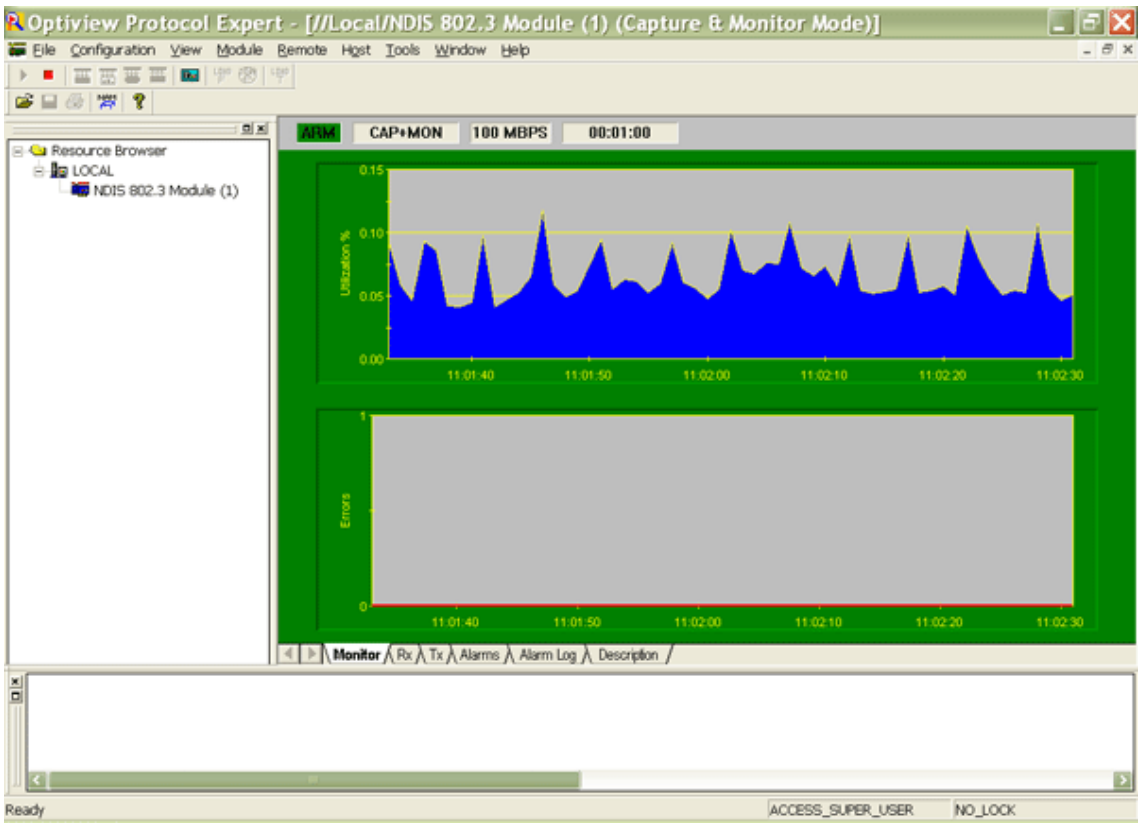


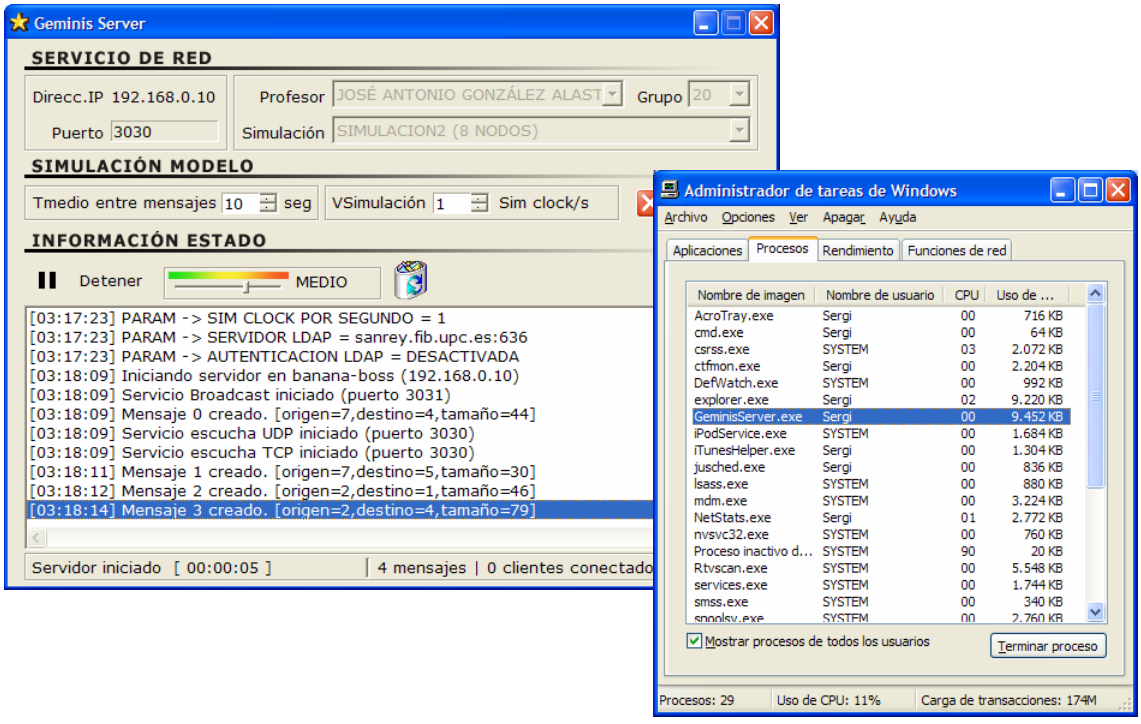
Ilustración 42 . Información rendimiento de red

De los resultados obtenidos de la observación, se puede decir que la puesta en funcionamiento de 'Géminis Server' en la red, no afectan negativamente a su rendimiento, ya que este no es superior al 0,10%.

5.5 Estabilidad del servidor y gestión de recursos

Se han realizado también pruebas para comprobar si el comportamiento del servidor sigue siendo el esperado después de una sesión. Normalmente el servidor estará ejecutándose durante 1 hora, que es el tiempo que dura una sesión de laboratorio. Se han realizado diversas pruebas, siendo la más larga de hasta 18 horas, en las que 'Géminis Server' y el 'MiniClient' han estado en contacto.

En las primeras pruebas se observaba un consumo excesivo de recursos de memoria, por lo que fue necesario depurar la gestión de ésta dentro de la aplicación. En las pruebas finales el consumo de memoria es el esperado y en las siguientes capturas de pantalla se muestra la ejecución del servidor durante más de 5 horas.



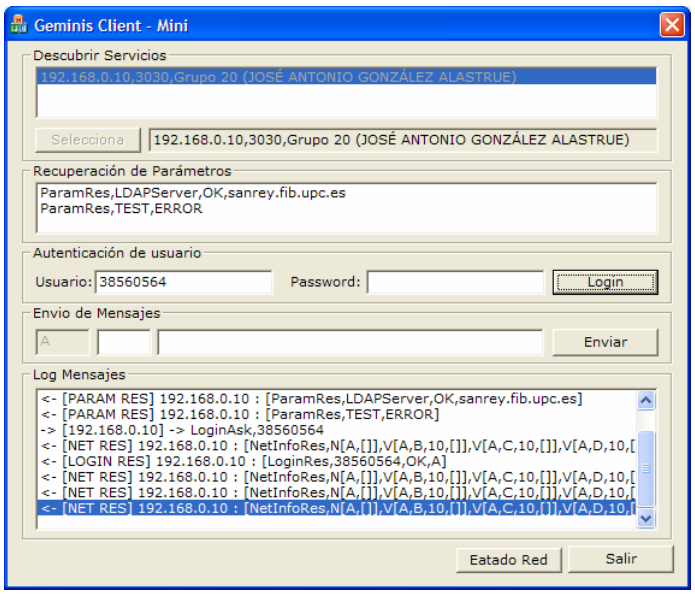
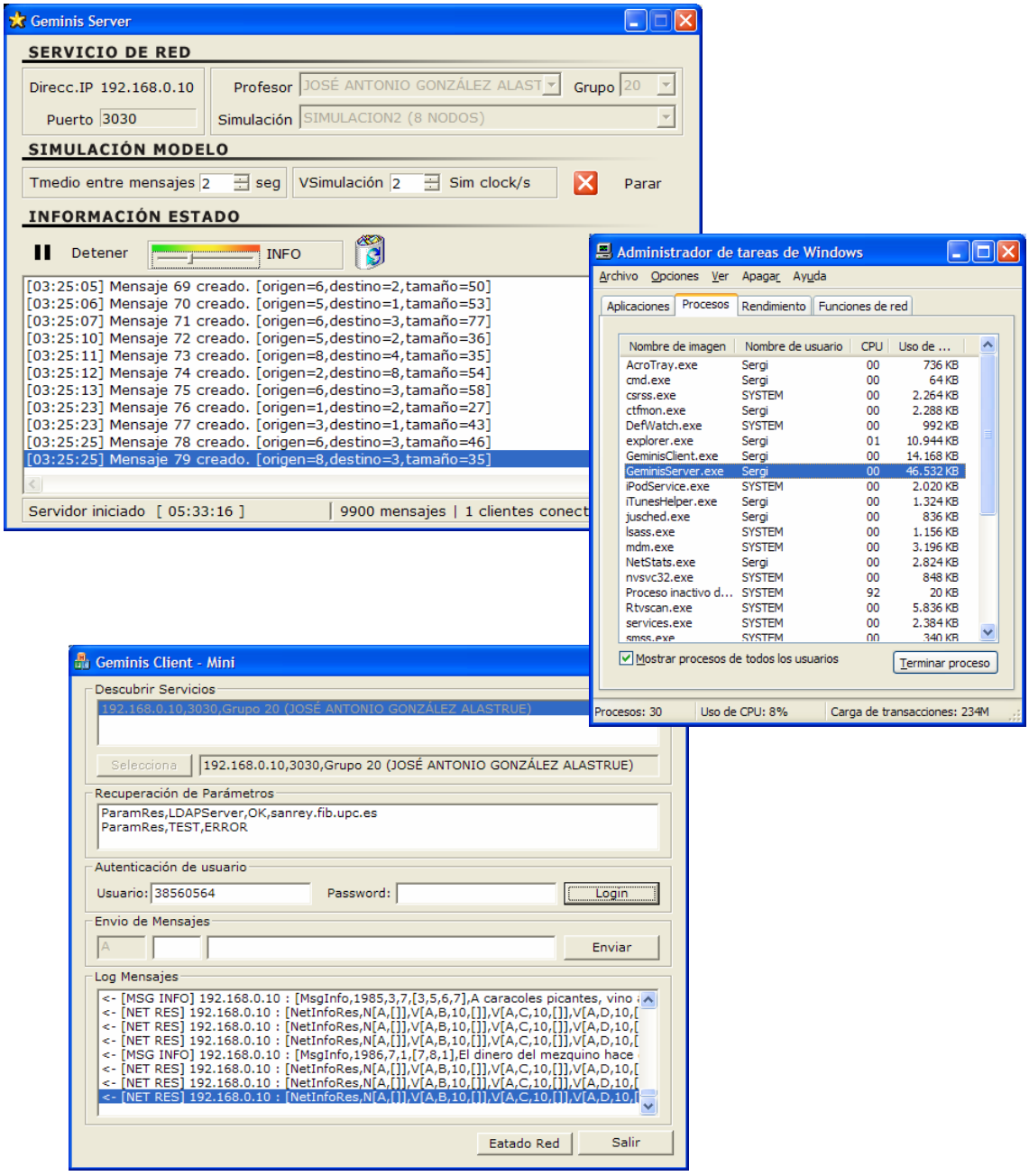


Ilustración 43 - MiniClient

Después de **5 horas y 33 minutos en ejecución**, podemos observar que el servidor ha generado y enviado al cliente **9.900 mensajes** (no se muestran en la ventana de información, debido a que se ha reducido el nivel de log mostrado). Al haber configurado la velocidad de simulación a 2 impulsos de reloj por segundo, el cliente debería haber recibido por cada paso de la simulación, un mensaje con la información del estado de la red, es decir, el cliente habrá recibido unos **40.000 mensajes** con información de **estado de la red**. En total el servidor ha enviado unos 50.000 mensajes, los mismos que ha recibido el cliente.

Podemos observar que después de este periodo de tiempo, tanto servidor como cliente siguen enviando y recibiendo mensajes sobre el estado de la red, y el uso de memoria por parte del servidor se mantiene en un tamaño razonable, ya que a medida que avanza la simulación va requiere más memoria para almacenar la información de los nuevos objetos creados.


Geminis Server
SERVICIO DE RED
Direcc.IP 192.168.0.10
Puerto 3030
Profesor JOSÉ ANTONIO GONZÁLEZ ALAST
Grupo 20
Simulación SIMULACION2 (8 NODOS)
SIMULACIÓN MODELO
Tmedio entre mensajes 2 seg
VSimulación 2 Sim clock/s
Parar
INFORMACIÓN ESTADO
Detener
INFO
[03:25:05] Mensaje 69 creado. [origen=6,destino=2,tamaño=50]
[03:25:06] Mensaje 70 creado. [origen=5,destino=1,tamaño=53]
[03:25:07] Mensaje 71 creado. [origen=6,destino=3,tamaño=77]
[03:25:10] Mensaje 72 creado. [origen=5,destino=2,tamaño=36]
[03:25:11] Mensaje 73 creado. [origen=8,destino=4,tamaño=35]
[03:25:12] Mensaje 74 creado. [origen=2,destino=8,tamaño=54]
[03:25:13] Mensaje 75 creado. [origen=6,destino=3,tamaño=58]
[03:25:23] Mensaje 76 creado. [origen=1,destino=2,tamaño=27]
[03:25:23] Mensaje 77 creado. [origen=3,destino=1,tamaño=43]
[03:25:25] Mensaje 78 creado. [origen=6,destino=3,tamaño=46]
[03:25:25] Mensaje 79 creado. [origen=8,destino=3,tamaño=35]
Servidor iniciado [05:33:16]
9900 mensajes | 1 clientes conect
Administrador de tareas de Windows
Archivo Opciones Ver Apagar Ayuda
Aplicaciones Procesos Rendimiento Funciones de red
Nombre de imagen Nombre de usuario CPU Uso de ...
AcroTray.exe Sergi 00 736 KB
cmd.exe Sergi 00 64 KB
csrss.exe SYSTEM 00 2.264 KB
ctfmon.exe Sergi 00 2.288 KB
DefWatch.exe SYSTEM 00 992 KB
explorer.exe Sergi 01 10.944 KB
GeminisClient.exe Sergi 00 14.168 KB
GeminisServer.exe Sergi 00 46.532 KB
iPodService.exe SYSTEM 00 2.020 KB
iTunesHelper.exe Sergi 00 1.324 KB
jusched.exe Sergi 00 836 KB
lsass.exe SYSTEM 00 1.156 KB
mdm.exe SYSTEM 00 3.196 KB
NetStats.exe Sergi 00 2.824 KB
nvsvc32.exe SYSTEM 00 848 KB
Proceso inactivo d... SYSTEM 92 20 KB
Rtvscan.exe SYSTEM 00 5.836 KB
services.exe SYSTEM 00 2.384 KB
Mostrar procesos de todos los usuarios
Terminar proceso
Procesos: 30
Uso de CPU: 8%
Carga de transacciones: 234M
Geminis Client - Mini
Descubrir Servicios
192.168.0.10,3030,Grupo 20 (JOSÉ ANTONIO GONZÁLEZ ALASTRUE)
Selecciona
192.168.0.10,3030,Grupo 20 (JOSÉ ANTONIO GONZÁLEZ ALASTRUE)
Recuperación de Parámetros
ParamRes,LDAPServer,OK,sanrey.fib.upc.es
ParamRes,TEST,ERROR
Autenticación de usuario
Usuario: 38560564
Password:
Login
Envio de Mensajes
A
Enviar
Log Mensajes
<- [MSG INFO] 192.168.0.10 : [MsgInfo,1985,3,7,[3,5,6,7],A caracoles picantes, vino
<- [NET RES] 192.168.0.10 : [NetInfoRes,N[A,[]],V[A,B,10,[]],V[A,C,10,[]],V[A,D,10,
<- [NET RES] 192.168.0.10 : [NetInfoRes,N[A,[]],V[A,B,10,[]],V[A,C,10,[]],V[A,D,10,
<- [NET RES] 192.168.0.10 : [NetInfoRes,N[A,[]],V[A,B,10,[]],V[A,C,10,[]],V[A,D,10,
<- [MSG INFO] 192.168.0.10 : [MsgInfo,1986,7,1,[7,8,1],El dinero del mezquino hace
<- [NET RES] 192.168.0.10 : [NetInfoRes,N[A,[]],V[A,B,10,[]],V[A,C,10,[]],V[A,D,10,
<- [NET RES] 192.168.0.10 : [NetInfoRes,N[A,[]],V[A,B,10,[]],V[A,C,10,[]],V[A,D,10,
<- [NET RES] 192.168.0.10 : [NetInfoRes,N[A,[]],V[A,B,10,[]],V[A,C,10,[]],V[A,D,10,
Eatado Red
Salir

6. Documentación

6.1 Requerimientos

Es necesario para la ejecución de '***Géminis Server***', en un ordenador o servidor las siguientes características:

- ***Sistema operativo***

  El software del servidor puede ejecutarse sobre cualquier ordenador personal en el que ejecute una versión de sistema operativo *Windows de 32 bits*, siendo preferible las versiones Windows 2000/2003, Windows XP o superior.

- ***Red TCP/IP***

  El ordenador en el que se ejecutará 'Géminis Server' debe estar conectado a una red y tener las librerías para operar con el protocolo TCP/IP. Los clientes deberán estar situados en el mismo rango de red (sin ningún '*router*' o '*firewall*' entre ellos), para permitir a los clientes descubrir los servicios del servidor, que son publicados en la red mediante mensajes de 'broadcast'.

  No es necesaria ninguna *dirección IP* específica para el servidor, ya que una vez que lo iniciemos este anunciará sus servicios en el segmento de red en el que se encuentre, incluyendo en su anuncio su dirección de red.

  Los *puertos* usados por el servidor y los clientes para comunicarse son el '3030' y el '3031' respectivamente, pudiendo ser modificados

mediante la tabla de parámetros 'ParametrosServidor' que encontraremos en la base de datos de la aplicación.

- ***'Open Database Connectivity' (ODBC)***

  Es necesario tener instalado en el ordenador en el que se ejecute la aplicación la capa de abstracción de acceso a datos ODBC. Este software viene instalado por defecto en todos los ordenadores de la plataforma Windows 32 bits, dentro del paquete MDAC (Microsoft Data Access Components).

- ***Librerías 'Microsoft Foundation Clas's (MFC)***

  Las librerías MFC proporcionan un marco de trabajo de aplicaciones para la programación en Microsoft Windows. Desarrollada en C++, MFC proporciona gran parte del código necesario para administrar ventanas, menús y cuadros de diálogo; realizar operaciones básicas de entrada y salida de archivos; almacenar colecciones de objetos de datos, etc.

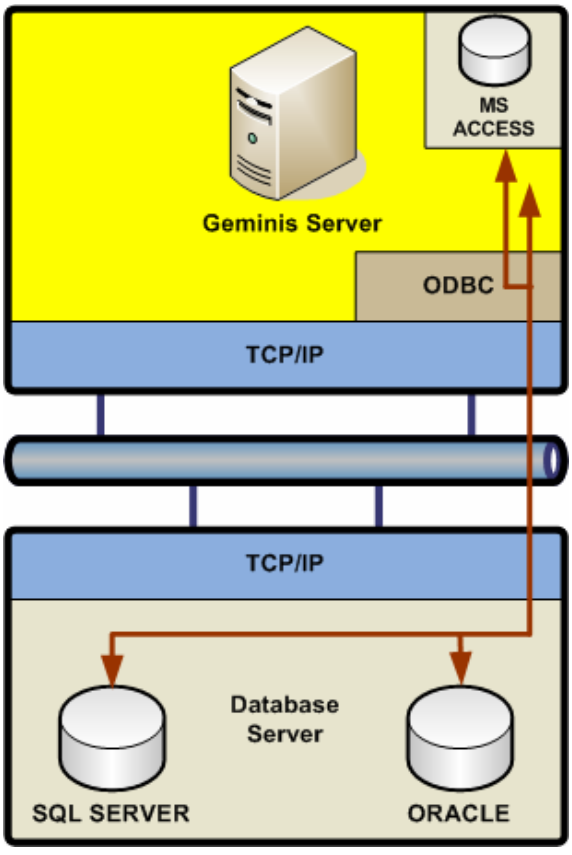
  En el desarrollo de '***Géminis Server***' se ha usado la *versión 7.1* de dichas librerías, siendo posible trabajar con versiones superiores. Estas librerías vienen instaladas en la mayoría de versiones del sistema operativo Windows 32 bits.

  Para evitar la aparición de conflictos con otras versiones existentes en el mismo ordenador, se incluyen junto con el ejecutable del servidor, las librerías MFC (.dll). Es suficiente colocarlas en el mismo directorio que el ejecutable del servidor para que use estas en lugar de otras instaladas en el sistema operativo.

6.2. Manual de instalación

6.2.1 ODBC

El acceso a la base de datos se realiza desde el servidor, por tanto solo ha de ser en este equipo en el que deba configurarse un acceso a la base de datos. Al utilizar la capa ODBC para acceder a los datos, estos pueden localizarse en distintas plataformas (MS Access, MS SQL Server, Oracle, etc) y/o localizaciones (local, remoto) de forma que resulte transparente para programa.

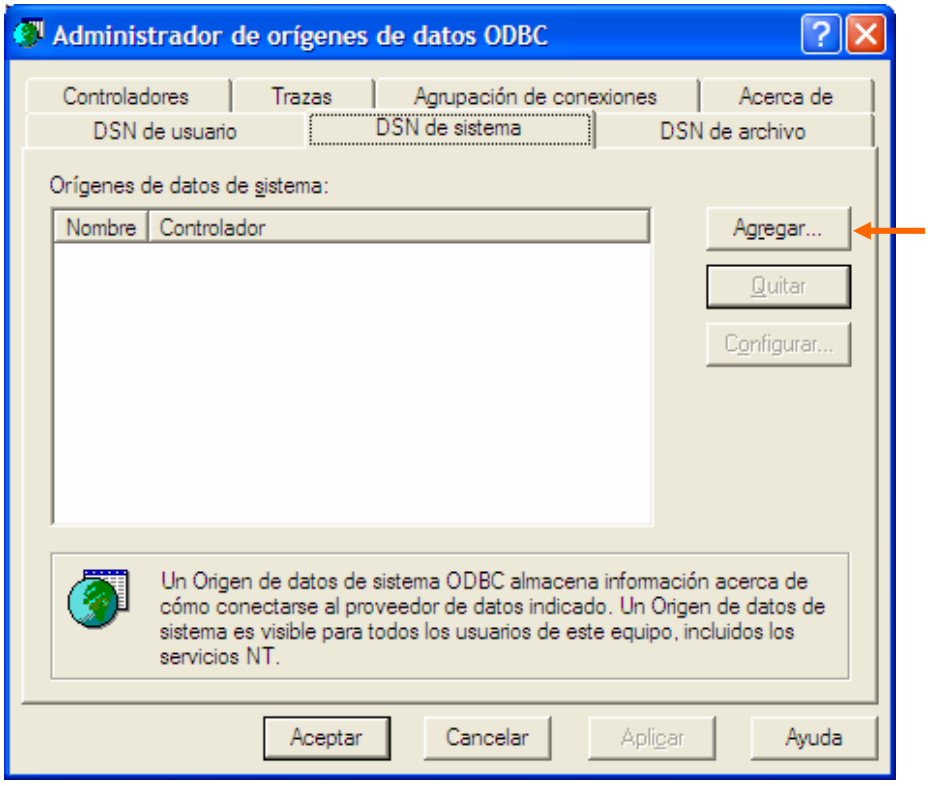


44. Acceso a datos locales o remotos

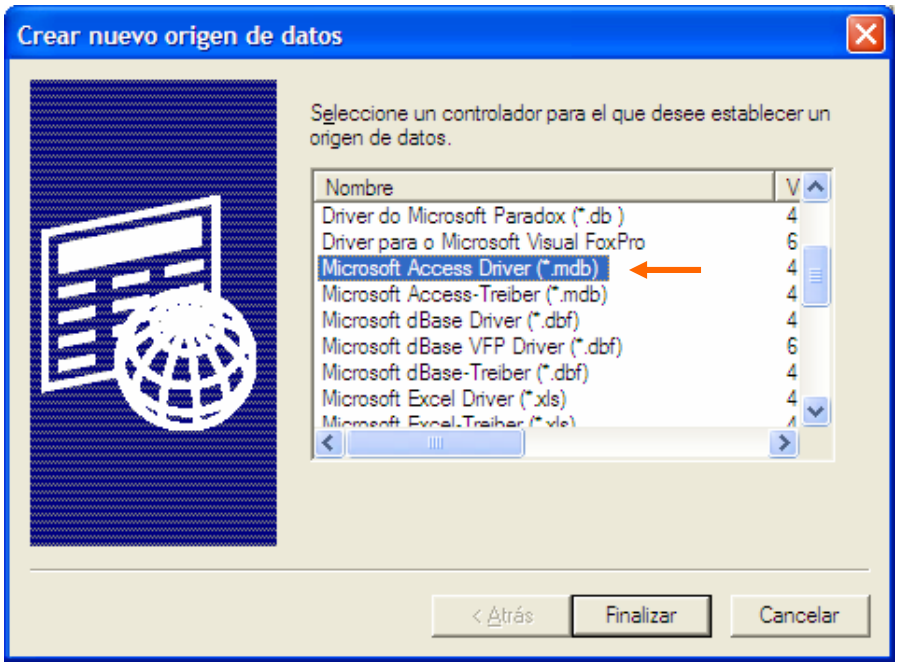
Para ellos, debemos crear una entrada DSN que siempre se llamará 'GeminisData' en el administrador de ODBC siguiendo los siguientes pasos:

- Para una base de datos local MS Access:
  - Abrimos el 'Administrador de ODBC', en equipos Windows 2000,2003 y Windows XP, lo encontraremos en el 'panel de control', dentro del apartado 'Herramientas administrativas', bajo el nombre de 'Orígenes de datos (ODBC)'

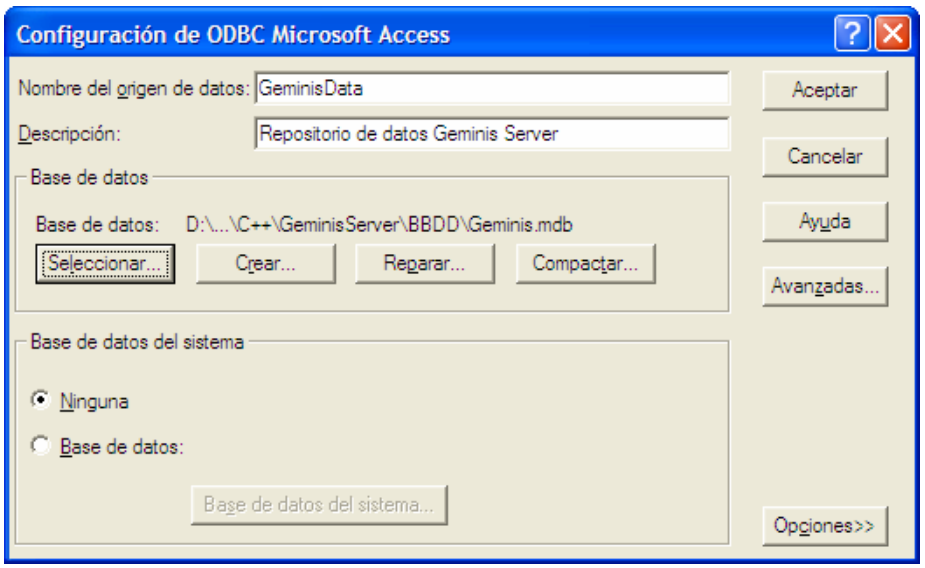
  - Una vez abierto el administrador, nos dirigimos a la pestaña 'DSN sistema' y agregamos una nueva entrada pulsando el botón 'agregar'



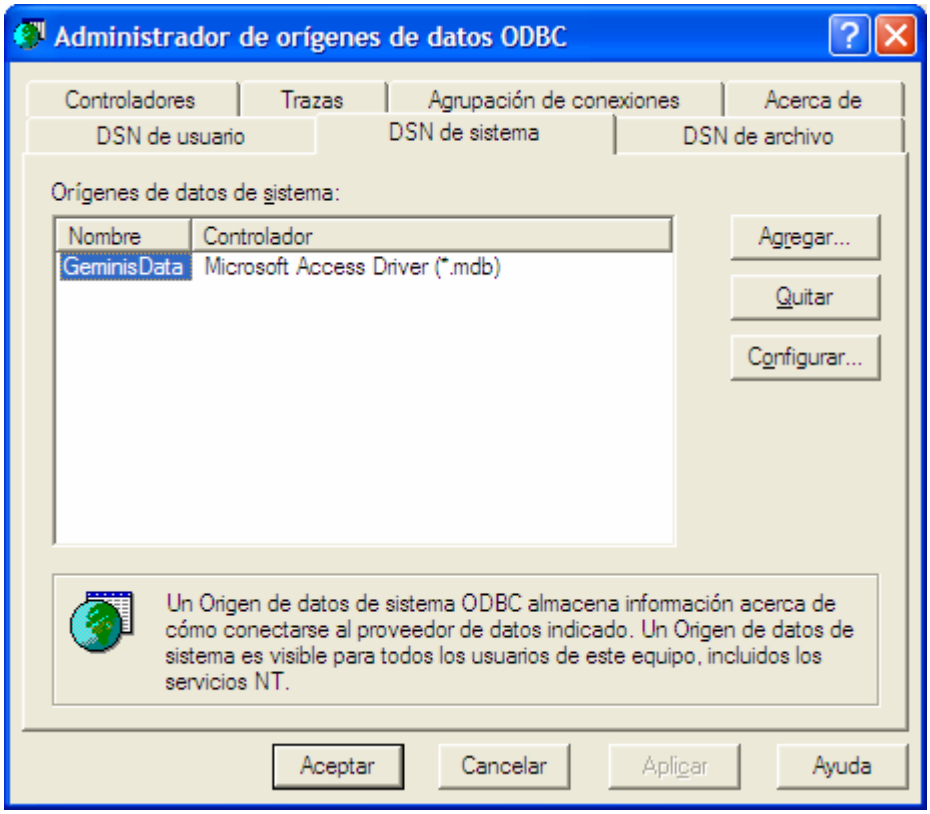
- Seleccionaremos el tipo de origen de datos: 'Microsoft Access Driver'



- El nombre del origen de datos será 'GeminisData' y seleccionamos la base de datos que queremos usar y que será accesible desde el servidor actual



- pulsamos 'Aceptar' y en nuestro administrador de ODBC debería aparecer una entrada similar a la siguiente:

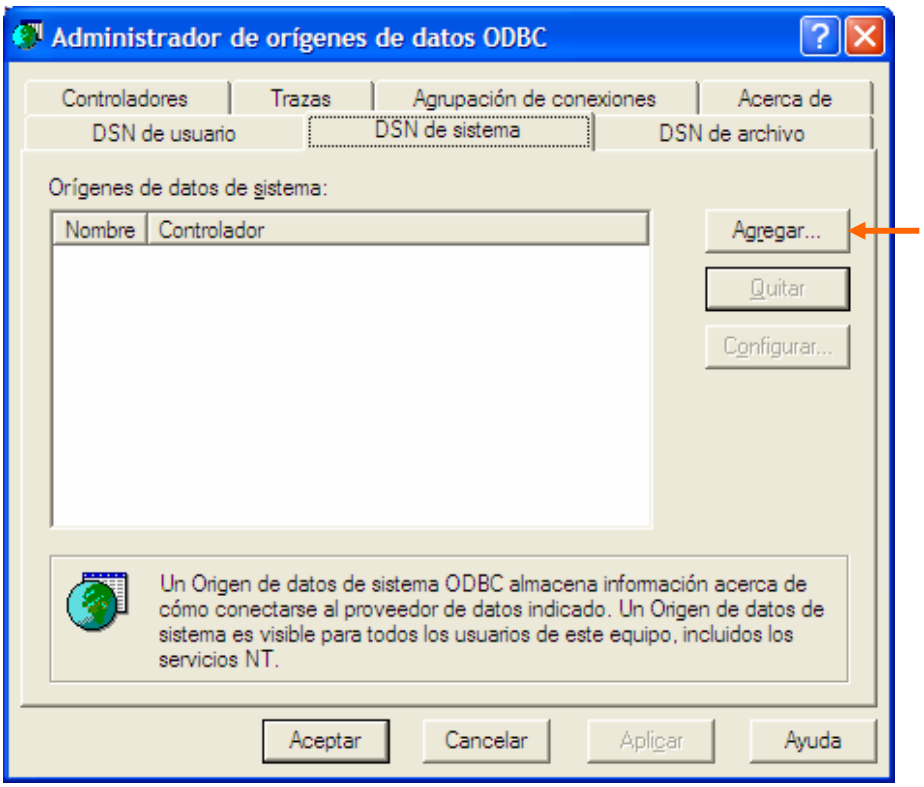


- ya tenemos configurado el acceso al repositorio de datos

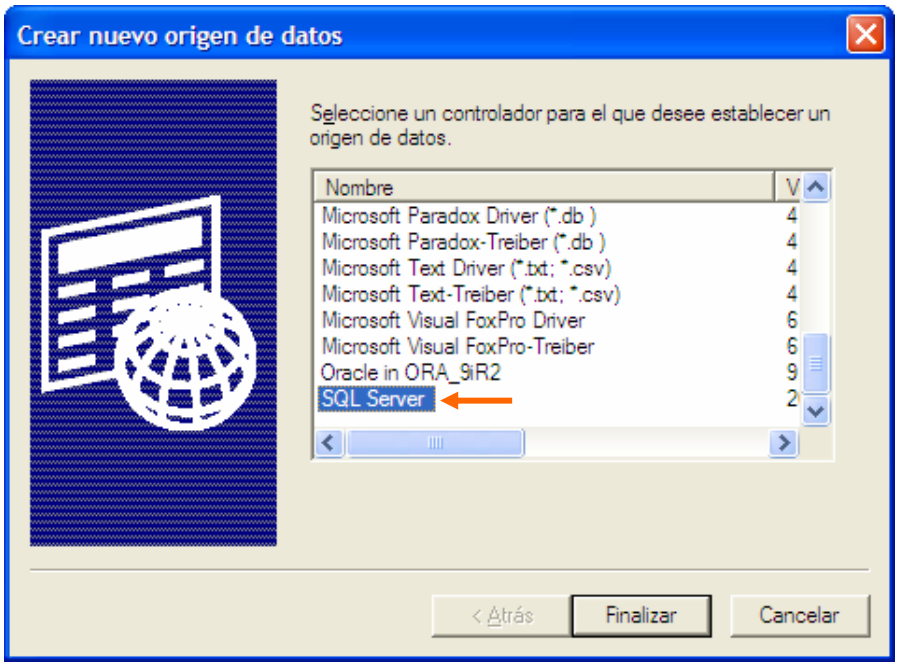
- Para una base de datos en red MS SQL Server:

  - Abrimos el 'Administrador de ODBC', en equipos Windows 2000,2003 y Windows XP, lo encontraremos en el 'panel de control', dentro del apartado 'Herramientas administrativas', bajo el nombre de 'Orígenes de datos (ODBC)'

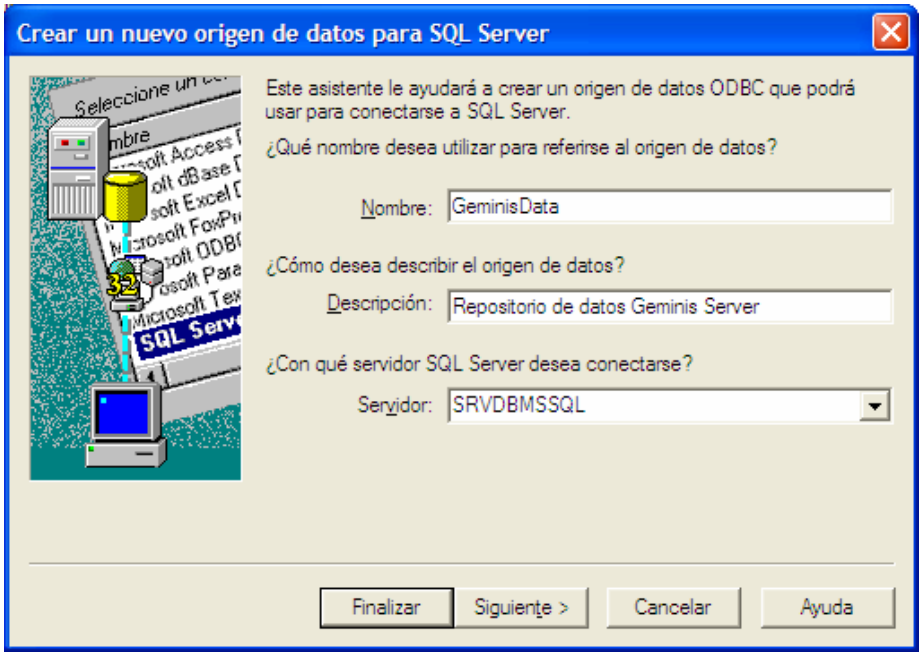
  - Una vez abierto el administrador, nos dirigimos a la pestaña 'DSN sistema' y agregamos una nueva entrada pulsando el botón 'agregar'



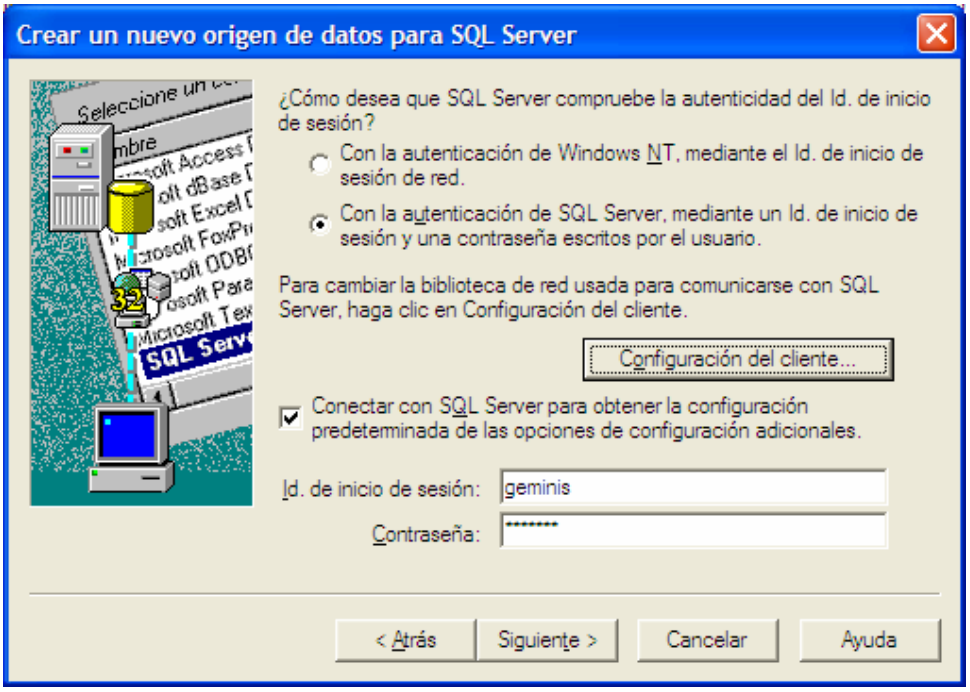
- Al pedirnos el tipo de origen de datos, indicaremos 'SQL Server'



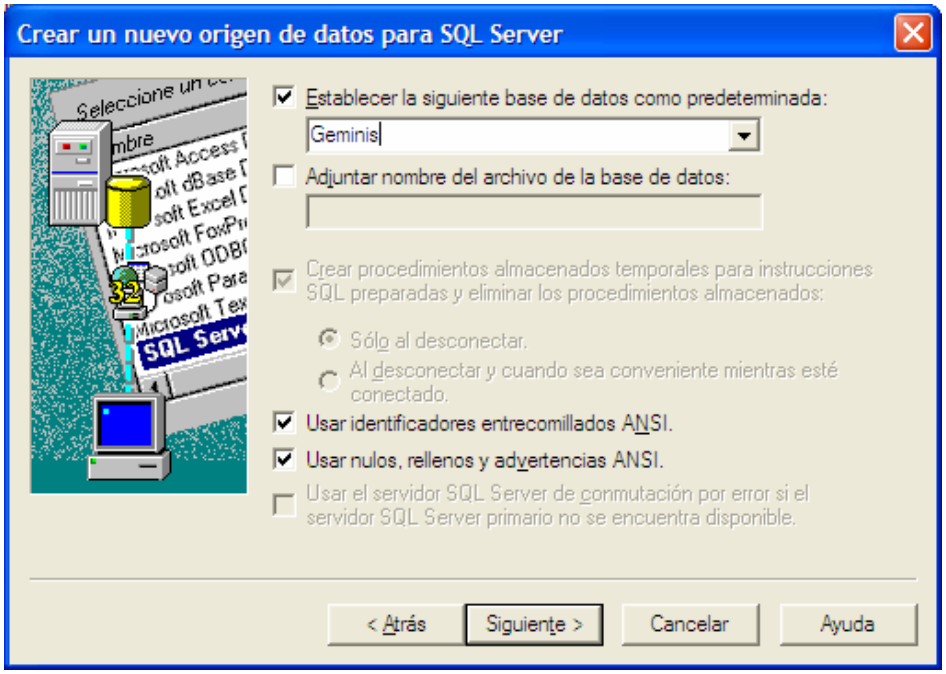
- El nombre del origen de datos será 'GeminisData' y el servidor será aquel en el que se encuentre el repositorio de datos



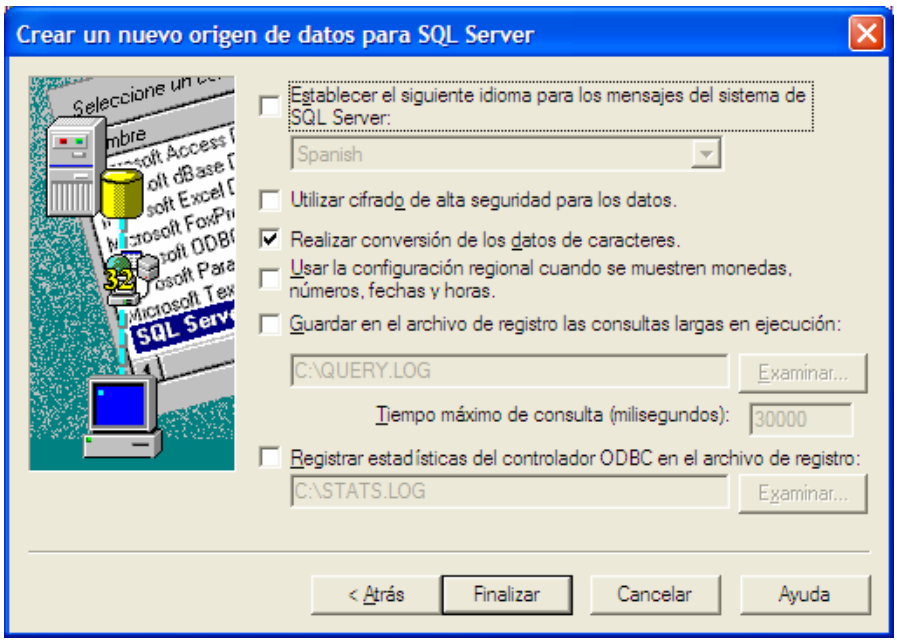
- pulsamos 'Siguiente y proporcionamos la información de autentificación al servidor de base de datos: [Username: geminis / Password: 123powq]



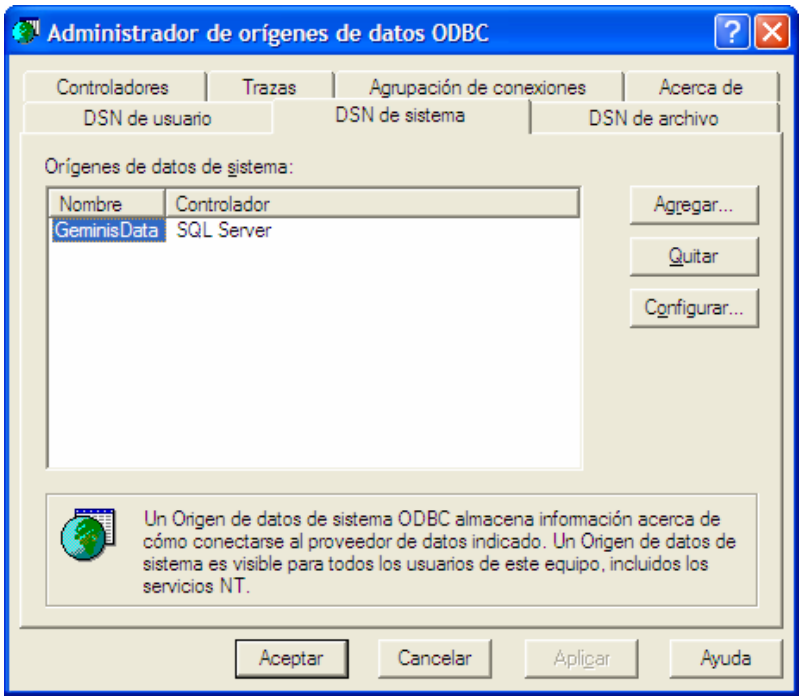
- Como base de datos, debemos seleccionar 'Géminis'



- Pulsamos siguiente



- pulsamos 'Finalizar' y en nuestro administrador de ODBC debería aparecer una entrada similar a la siguiente:



- ya tenemos configurado el acceso al repositorio de datos

Para otros tipos de bases de datos el procedimiento a seguir es similar al descrito anteriormente, seleccionando como tipo origen de datos el controlador proporcionado por el fabricante.

6.2.2 Aplicación

La aplicación '**Géminis Server**' no necesita ninguna instalación adicional, exceptuando la definición de la entrada en el DSN vista en el apartado anterior.

Es posible ejecutarlo en distintos servidores simplemente copiando los ficheros a una ubicación accesible por el servidor (unidad local o unidad de red)

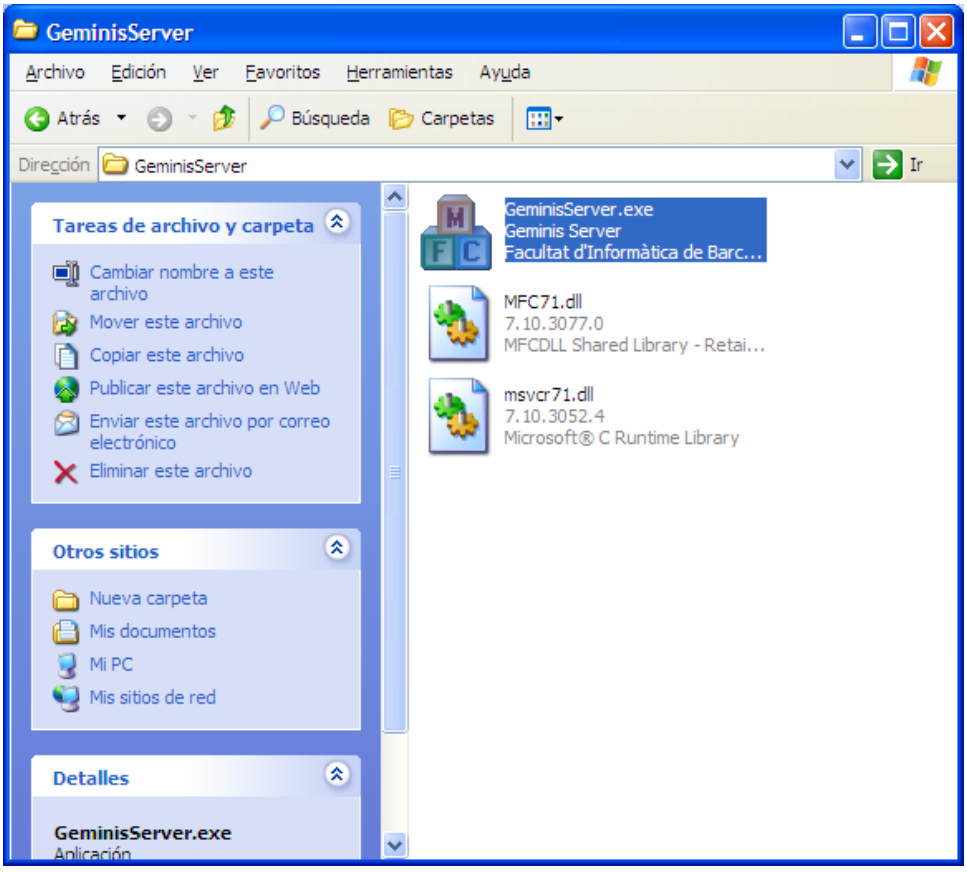


Ilustración 45 . Aplicación 'Géminis Server'

6.3. Manual de usuario

A continuación detallamos las operaciones que realizará el usuario de '***Géminis Server***' para poner en marcha el servicio.

Autenticación

En primer logar hemos de validar al usuario, que será un profesor que esté dado de alta en el servicio LDAP. En la pantalla de login, introduciremos el nombre de usuario y su password correspondiente.

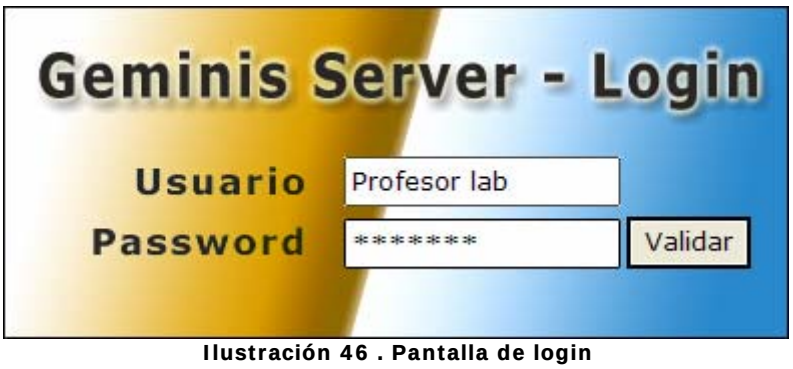


Ilustración 46 . Pantalla de login

Los parámetros que se usaran en este paso para autenticar el usuario los encontraremos en la base de datos de la aplicación, dentro de la tabla '*ParametrosServidor*':

- LDAPServer, nos indicará a que directorio de LDAP dirigiremos nuestra consulta
- LDAPPort, nos indicará el puerto que usaremos para acceder al servicio LDAP
- LDAPContainer, nos indicará en que rama del directorio buscaremos el identificadores de usuario proporcionado por el profesor

Una vez realizada correctamente la autenticación (tenemos 3 intentos), se nos mostrará la pantalla principal de la aplicación, en la que deberemos realizar las siguientes acciones:

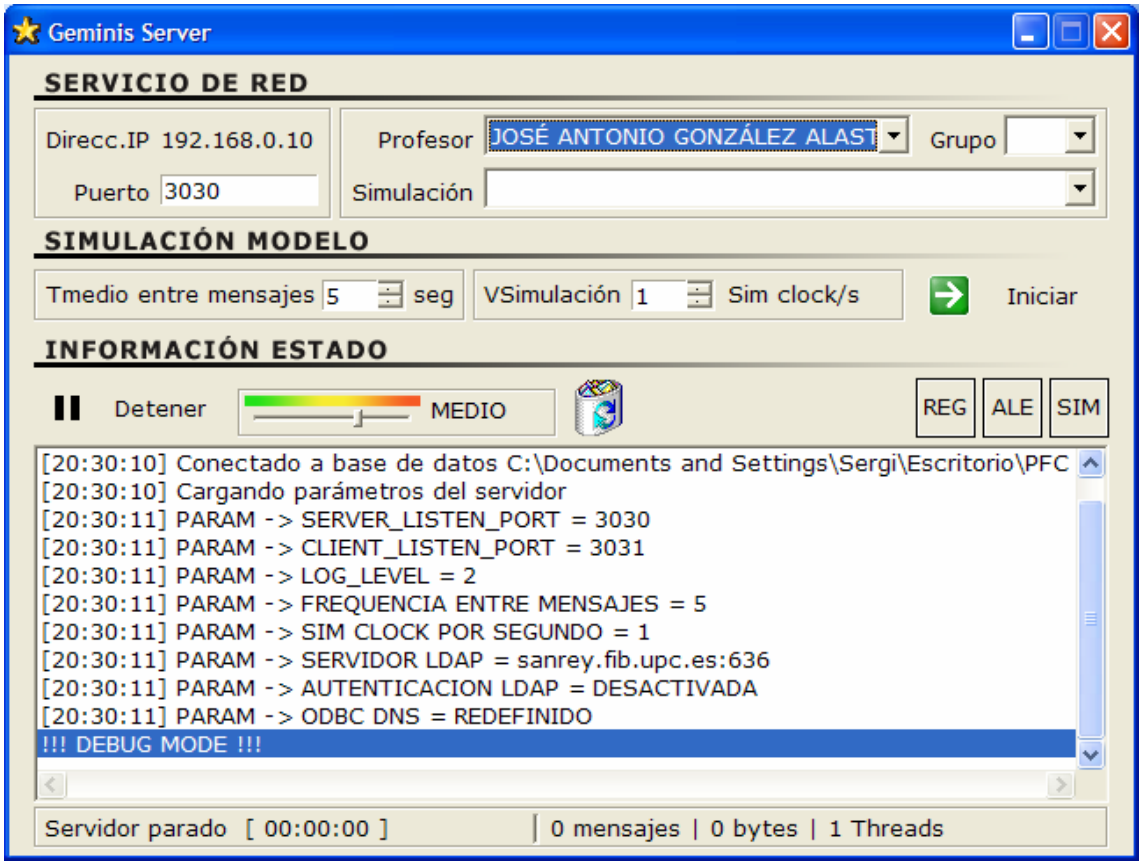


Ilustración 47 . 'Géminis Server'

Configurar servicio de red

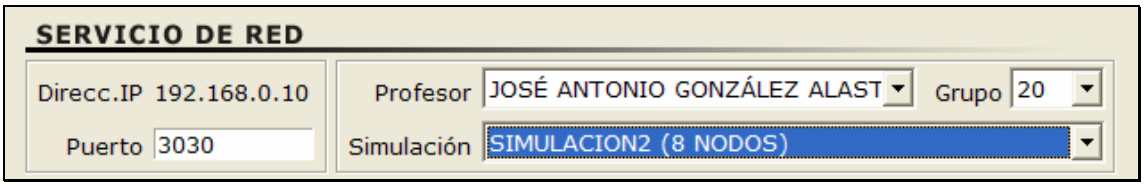


Ilustración 48 . Detalle 'servicio de red'

En primer lugar configuraremos las características del servicio de red que queremos poner a disposición de los alumnos. Para ello configuraremos los siguientes parámetros:

- **Puerto**. El servidor atenderá las peticiones de los clientes por defecto por el puerto '3030', valor que viene indicado el parámetro 'ServerListenPort' de la tabla 'ParamatrosServidor'. Es posible modificar dicho puerto si existe un conflicto con otro software en nuestro servidor que use el mismo puerto. (por ejemplo si ya existiese un 'Géminis Server' activo en la misma máquina)

- **Profesor**. Indicaremos quien será el profesor responsable de la simulación que pondremos en funcionamiento. La lista de profesores se recupera de la base de datos de la aplicación.

- **Grupo**. Indicaremos el grupo que estará permitido participar de la simulación que pondremos en funcionamiento.

- **Simulación**. Indicaremos que definición de red usaremos en nuestra simulación. Estas definiciones de red, están almacenadas en la base de datos de la aplicación.

Opciones sobre el simulador

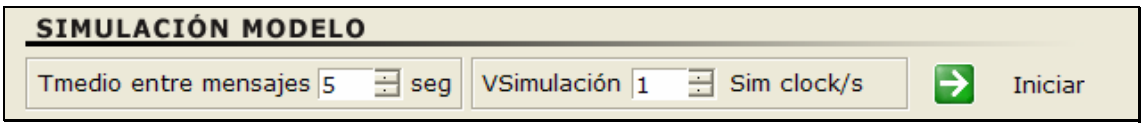


Ilustración 49 . Detalle 'Simulación del modelo'

- **Tiempo medio entre mensajes**. Podemos definir cual será la frecuencia media en que se crearán mensajes automáticos en la red. El rango permitido es de 1 a 10 segundos entre mensajes.

- **Velocidad de simulación**. Podemos indicar cuantos impulsos de simulación se producirán por segundo. El rango permitido es de 1 a 20 impulsos por segundo.

- **Iniciar el simulador**. Podemos iniciar el simulador. Una vez iniciado, en el botón de inicio aparecerá la opción de parar el simulador.

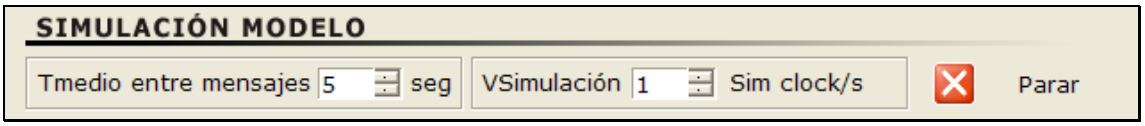


Ilustración 50 . Detalle 'Simulación del modelo'

- **Iniciar el simulador**. Podemos detener el simulador en cualquier momento, para lo cual se nos pedirá confirmación, ya que esta acción implica detener el contacto con los clientes.

Información de Estado

En la parte inferior de la pantalla principal aparecerá información sobre el estado de la aplicación en esa ventana, podremos observar los mensajes generados por cada uno de los componentes de la aplicación.

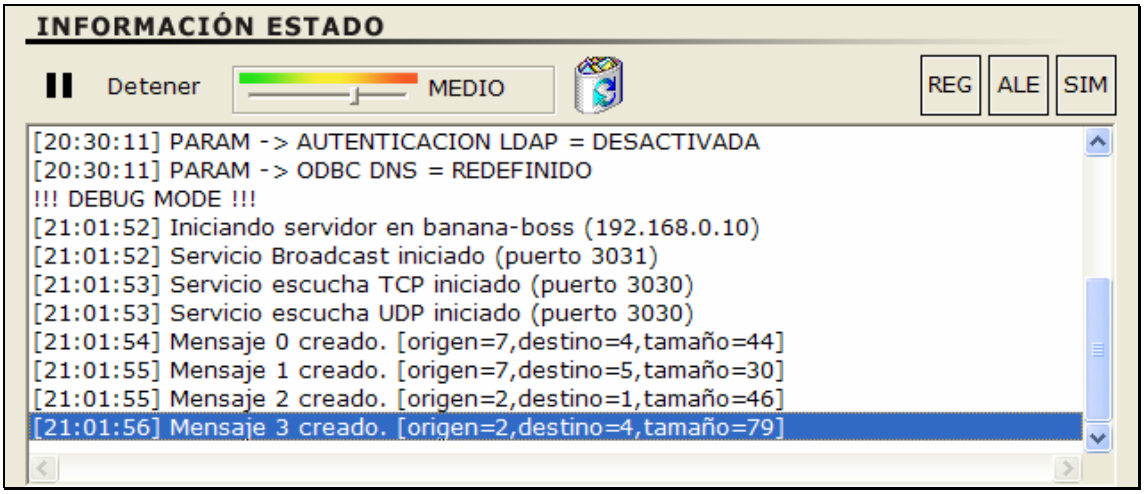


Ilustración 51 . D etalle 'Información de estado'

Podemos realizar varias acciones sobre el la información que se va generando:

- Detener y volver a activar la visualización de la información de Log.
- Vaciar la información mostrada por pantalla hasta el momento.
- Modificar el nivel de detalle de la información proporcionado por los componentes del sistema.

Finalmente en la barra de estado, se nos mostrará el tiempo que la simulación ha estado activa, el numero de mensajes creados en la simulación (tanto automáticos, como los generados por los propios alumnos) y el número de clientes conectado.

7. Análisis económico

7.1 Coste desarrollo

A continuación se detalla el coste de las horas que ha representado el desarrollo del proyecto:

Actividad	Horas
Reuniones definición requerimientos	25
Estudio tecnologías	60
Gestión	10
Especificación	80
Implementación	250
Pruebas	75
Mantenimiento y adaptación	25
Documentación	125
	650 horas

7.2 Coste explotación

Una vez puesto en funcionamiento el proyecto 'Géminis', es necesario la realización de una serie de tareas para el mantenimiento de la información con que se trabaja:

- Mantenimiento de alumnos. Dar de alta alumnos en la base de datos cada cuatrimestre.
- Mantenimiento de profesores. Dar de alta o baja aquellos profesores en la base de datos de la aplicación.
- Mantenimiento de grupos. Dar de alta nuevos grupos si es necesario o modificar los existentes. Asociar los profesores y alumnos a un grupo.
- Mantenimiento de mensajes automáticos. Es posible ir cambiando el contenido de los mensajes automáticos periódicamente.
- Mantenimiento de definiciones de redes. Dar la posibilidad de modificar las definiciones ya existentes, añadir nuevas y dar de baja otras existentes.

En este proyecto no se ha llevado a cabo, pero sería posible el desarrollo de una aplicación que ayudase al mantenimiento de los datos de la aplicación.

7.3 Coste software

A continuación se detalla el software usado en el desarrollo de este proyecto.

Herramientas de desarrollo

Visual C++	
Fabricante	Microsoft
Consultado en	http://msdn.microsoft.com/vstudio/howtobuy/pricing.aspx
Fecha consulta	Junio 2004
Versión	Visual Studio .NET 2003 Professional Full Packaged Product
Precio	$1079 US

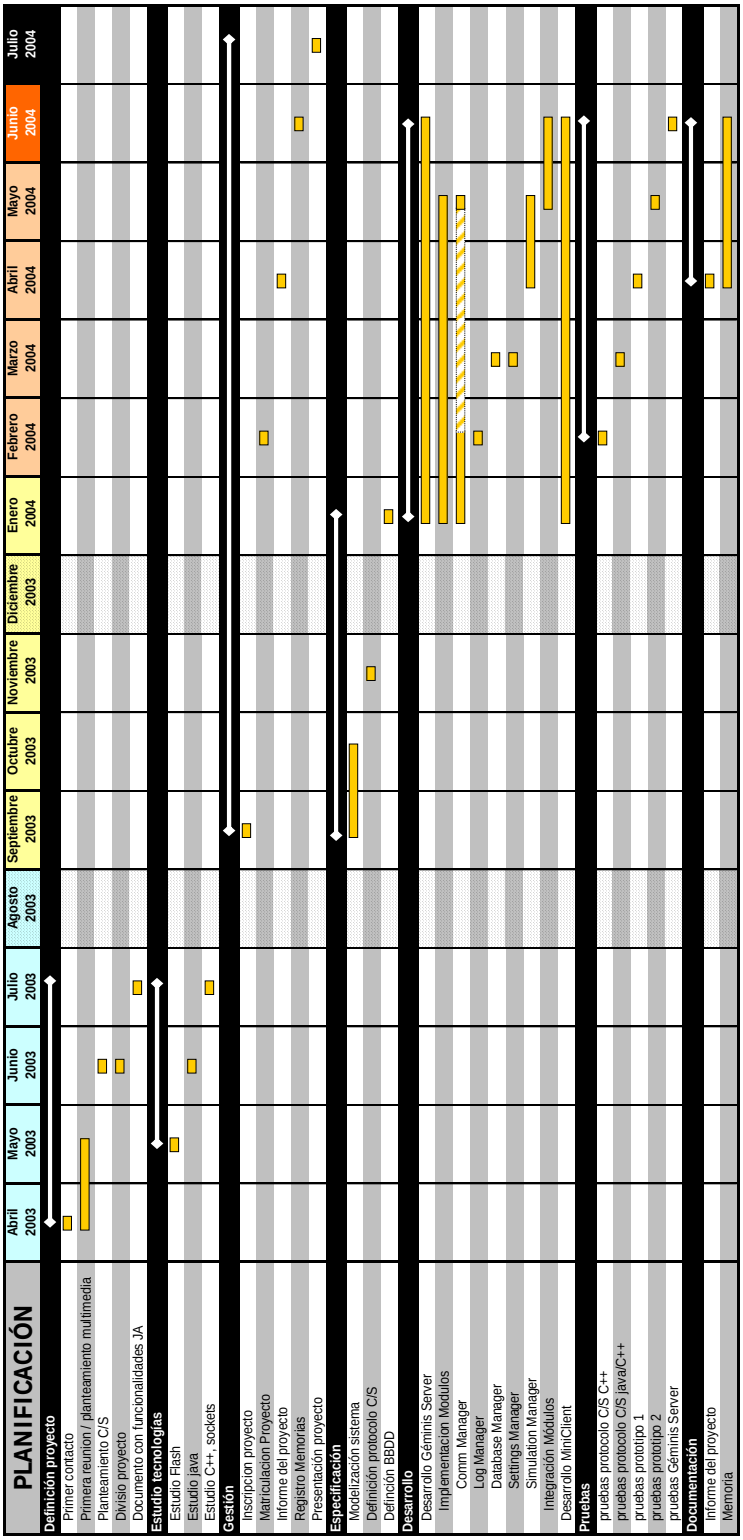
Servicios

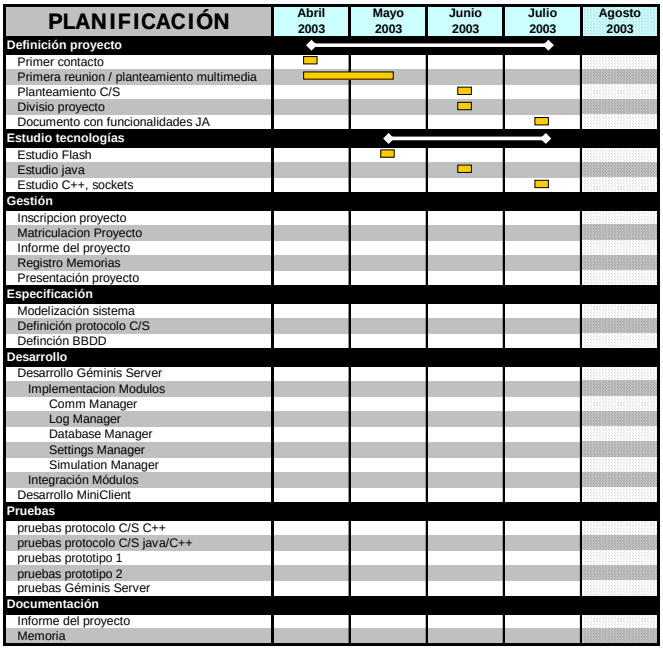
Base de datos : MS SQL Server	
Fabricante	Microsoft
Consultado en	http://www.microsoft.com/sql/howtobuy/default.asp
Fecha consulta	Junio 2004
Versión	Microsoft SQL Server Standard Edition 1 Processor License
Precio	A partir de $ 5000 US
Observaciones	Este producto ya existe en la red donde instalaremos la aplicación

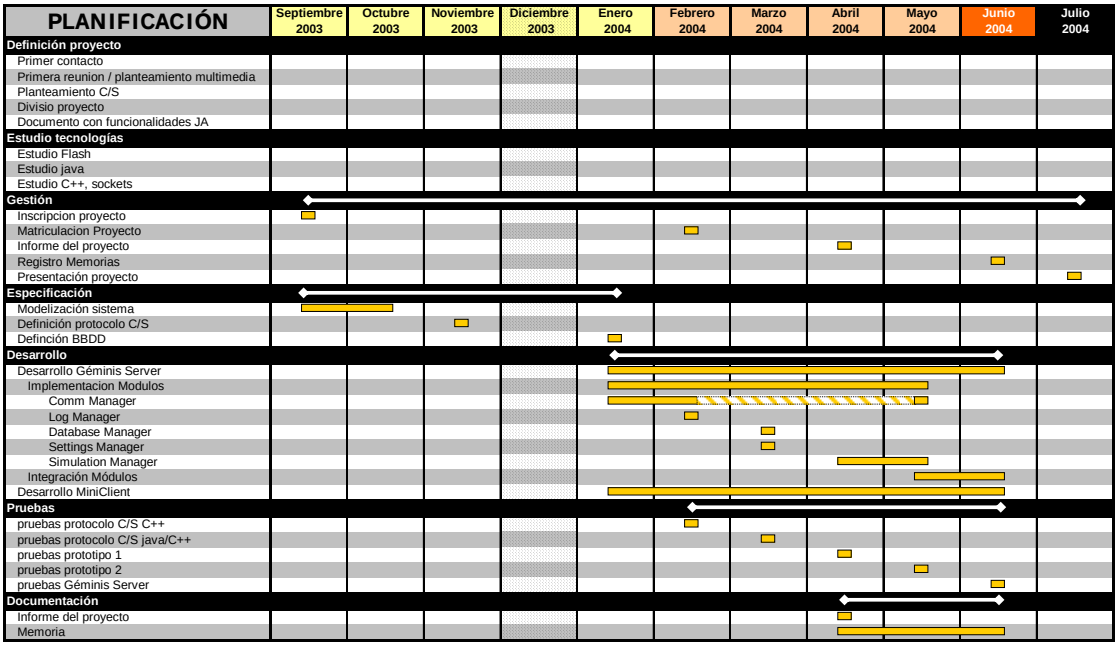
Documentación

Microsoft Office	
Fabricante	Microsoft
Consultado en	http://www.microsoft.com/spain/office/editions/howtobuy/professional.asp
Fecha consulta	Junio 2004
Versión	Office Profesional 2003 PC
Precio	744 €

7.4 Planificación




PLANIFICACIÓN
Abril 2003
Mayo 2003
Junio 2003
Julio 2003
Agosto 2003
Definición proyecto
Primer contacto
Primera reunion / planteamiento multimedia
Planteamiento C/S
Divisio proyecto
Documento con funcionalidades JA
Estudio tecnologías
Estudio Flash
Estudio java
Estudio C++, sockets
Gestión
Inscripcion proyecto
Matriculacion Proyecto
Informe del proyecto
Registro Memorias
Presentación proyecto
Especificación
Modelización sistema
Definición protocolo C/S
Defincón BBDD
Desarrollo
Desarrollo Géminis Server
Implementacion Modulos
Comm Manager
Log Manager
Database Manager
Settings Manager
Simulation Manager
Integración Módulos
Desarrollo MiniClient
Pruebas
pruebas protocolo C/S C++
pruebas protocolo C/S java/C++
pruebas prototipo 1
pruebas prototipo 2
pruebas Géminis Server
Documentación
Informe del proyecto
Memoria


PLANIFICACIÓN
Septiembre 2003
Octubre 2003
Noviembre 2003
Diciembre 2003
Enero 2004
Febrero 2004
Marzo 2004
Abril 2004
Mayo 2004
Junio 2004
Julio 2004
Definición proyecto
Primer contacto
Primera reunion / planteamiento multimedia
Planteamiento C/S
Divisio proyecto
Documento con funcionalidades JA
Estudio tecnologías
Estudio Flash
Estudio java
Estudio C++, sockets
Gestión
Inscripcion proyecto
Matriculacion Proyecto
Informe del proyecto
Registro Memorias
Presentación proyecto
Especificación
Modelización sistema
Definición protocolo C/S
Defincón BBDD
Desarrollo
Desarrollo Géminis Server
Implementacion Modulos
Comm Manager
Log Manager
Database Manager
Settings Manager
Simulation Manager
Integración Módulos
Desarrollo MiniClient
Pruebas
pruebas protocolo C/S C++
pruebas protocolo C/S java/C++
pruebas prototipo 1
pruebas prototipo 2
pruebas Géminis Server
Documentación
Informe del proyecto
Memoria

8. Contenido del CD

Contenido del CD	
Carpeta	Descripción
Ejecutables	Ficheros ejecutables de la aplicación
Código Fuente	Código Fuente de la aplicación
BBDD	Base de datos de ejemplo

Contenido de la carpeta 'Ejecutables'	
Fichero	Descripción
GeminisServer.exe	Géminis Server
GeminisClient.exe	Géminis – MiniClient
MFC71.dll	MFC (Microsoft Foundation Class)
Msvcr71.dll	MFC (Microsoft Foundation Class)
NetTester.exe	Pruebas de red
NetSim.exe	Pruebas de simulación

Bibliografía

Guasch, A.;Piera, M.À.;Casanovas, J.;Figueras,J.

Modelado y Simulación. Edicions UPC, 2002

Jaume Sistac. Documentación de la asignatura de Diseñote bases de datos (DBD), 1998

Kristian Besley, Sham Bangla, Amanda Farr

Flash Mx. Anaya Multimedia, 2003

Lazaro Issi Camy

Action Script para Flash Mx. Anaya Multimedia, 2003

Beck Zaratian

MICROSOFT VISUAL C++ 6.0. MANUAL DEL PROGRAMADOR, McGraw-Hill, 2001

http://fmc.axarnet.es/redes/tema_05.htm

Teoria sobre redes

http://fferrer.dsic.upv.es/cursos/Linux/Avanzado/HTML/index.html

Concepto de LDAP

Anexo 1. Definición Red Géminis

En la base de datos, guardaremos definiciones de redes, que serán seleccionadas por el profesor antes de iniciar la simulación. Estas definiciones nos detallaran todos los elementos de la red a simular.

Una red esta identificada por un número único y la define un nombre, varios nodos y varios vértices:

NOMBRE=<Nombre de la simulación>
NODO=<Nombre del nodo>
VERTICE=<Nodo origen>,<Nodo destino>,<Ancho de banda>

El nombre de la simulación, es una cadena de texto que permite cualquier carácter.

El nombre del nodo, es una cadena de texto. Se recomienda que sea lo más corta posible, ya que al notificar el estado de la red a los clientes se enviara el nombre del nodo, por lo que a mayor tamaño en su nombre, mayor será la información a enviar a los clientes. El máximo número de nodos de una red es de 255. No hay límite en el número de vértices.

El ancho de banda de un vértice, es un número, que indica la cantidad de bytes que cruzan el vértice a cada impulso del reloj de la simulación. En caso de no indicar ningún valor, toma por defecto el valor de '10'.

A continuación se muestran ejemplos de definiciones de red y su grafico correspondiente.

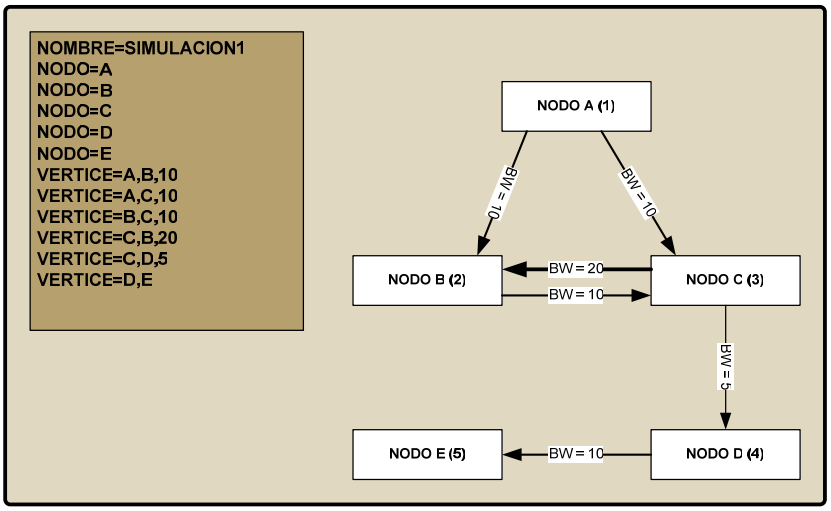


Ilustración 52 . Definición de red y su representación

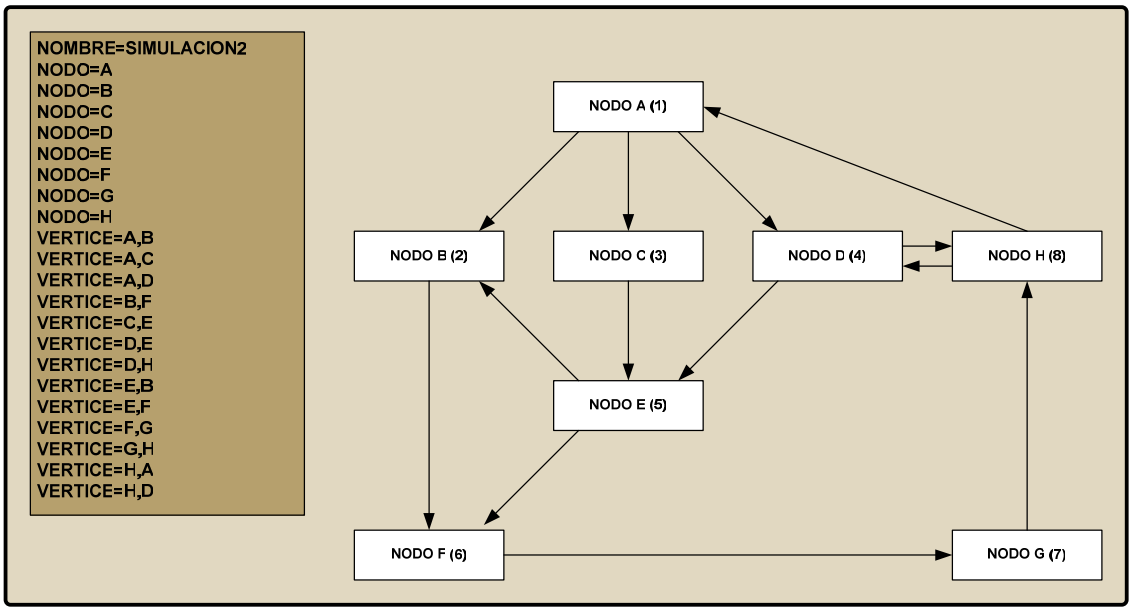


Ilustración 53 . Definición de red y su representación

Anexo 2. Parámetros de cliente y servidor

En la base de datos, encontraremos las tablas '*ParametrosServidor*' y '*ParametrosCliente*', en las que encontraremos los parámetros necesarios para configurar tanto diente como servidor.

Parámetros de Servidor	
ServerListenPort	Puerto de escucha del servidor
ClientListenPort	Puerto de escucha del cliente
LogLevel	Nivel de log por defecto [0,1,2,3]
LDAPServer	Servidor LDAP
LDAPPort	Puerto LDAP
LDAPContainer	Container donde buscar usuarios a autenticar
FreqSim	Pasos de simulación por segundo
FreqMsg	Tiempo medio entre mensajes generados automáticamente
NoLDAPPassword	Password a usar en caso de que la autenticación por LDAP esté desactivada

Parámetros de Cliente	
LDAPServer	Servidor LDAP
LDAPPort	Puerto LDAP
LDAPContainer	Container donde autenticar a los alumnos

Anexo 3. Opciones de línea de comando

Al iniciar el servidor, podemos pasarle las siguientes opciones por la línea de comando:

NOLDAP: En caso de que en la red en la que nos encontremos no esté disponible un servidor LDAP para autenticar a los profesores o bien este no esté en funcionamiento, evitaremos validar las credenciales en un servidor LDAP. Pero el password que deberán usar todos los profesores, será el mismo que se haya configurado en los parámetros de servidor bajo el nombre 'NoLDAPPassword'.

LocalServer: Usaremos este parámetro en caso de trabajar en una estación aislada de la red. En la misma computadora residirán servidor y cliente y tendrá instalado el protocolo TCP/IP.

Anexo 4. MiniCliente C++

Durante el desarrollo de 'Géminis Server' se ha ido desarrollando en paralelo un software en el que se han ido implementando las funcionalidades mínimas del cliente según las conversaciones mantenidas con el director del proyecto y el proyectista encargado del desarrollo de 'Géminis Client'.

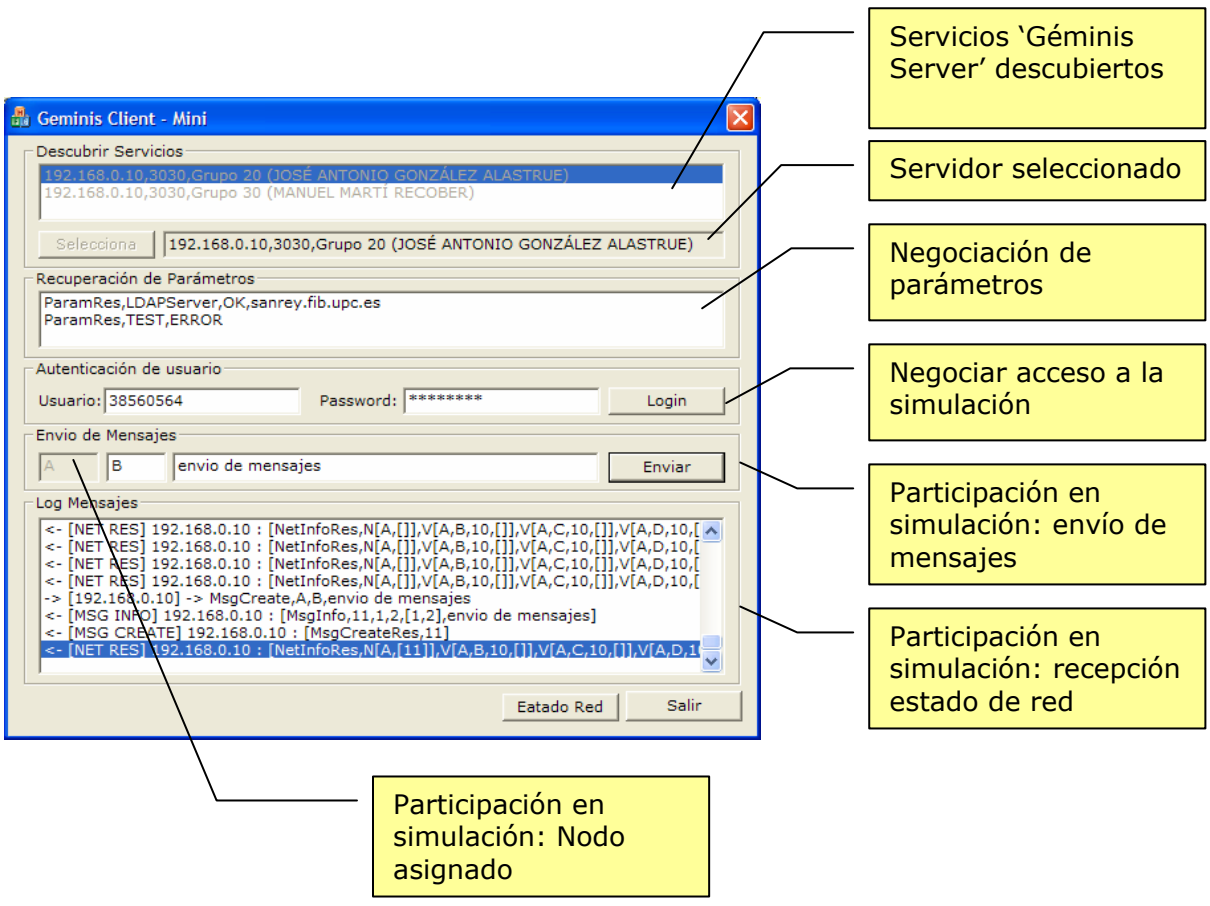
Los pasos que debe seguir un cliente hasta llegar a participar de la simulación son los siguientes:

- **Descubrimiento del servicio 'Géminis Server'**. Descubrir los posibles servicios de servidores 'Géminis Server' mostrar cada uno de los encontrados al alumno y dejar que sea este el que decida con cual quiere negociar su acceso.
- **Negociación de parámetros**. Una vez seleccionado un servidor le pedirá los parámetros necesarios para configurarse. Estos parámetros que se envían al cliente, se encuentran en la tabla 'ParámetrosCliente' de la base de datos. En nuestro caso, nos interesa pedirle al servidor cual es el servicio LDAP que debemos usar para autenticar las credenciales del alumno.
- **Autenticación LDAP**. Con las credenciales del usuario y la información de la localización del directorio LDAP, procederemos a validar el identificador y contraseña del alumno.
- **Negociar acceso a la simulación**. El cliente pasará al servidor el identificador del usuario, se verificará que pertenezca al grupo que ha iniciado la sesión y en caso de que no esta presente en la sesión o no

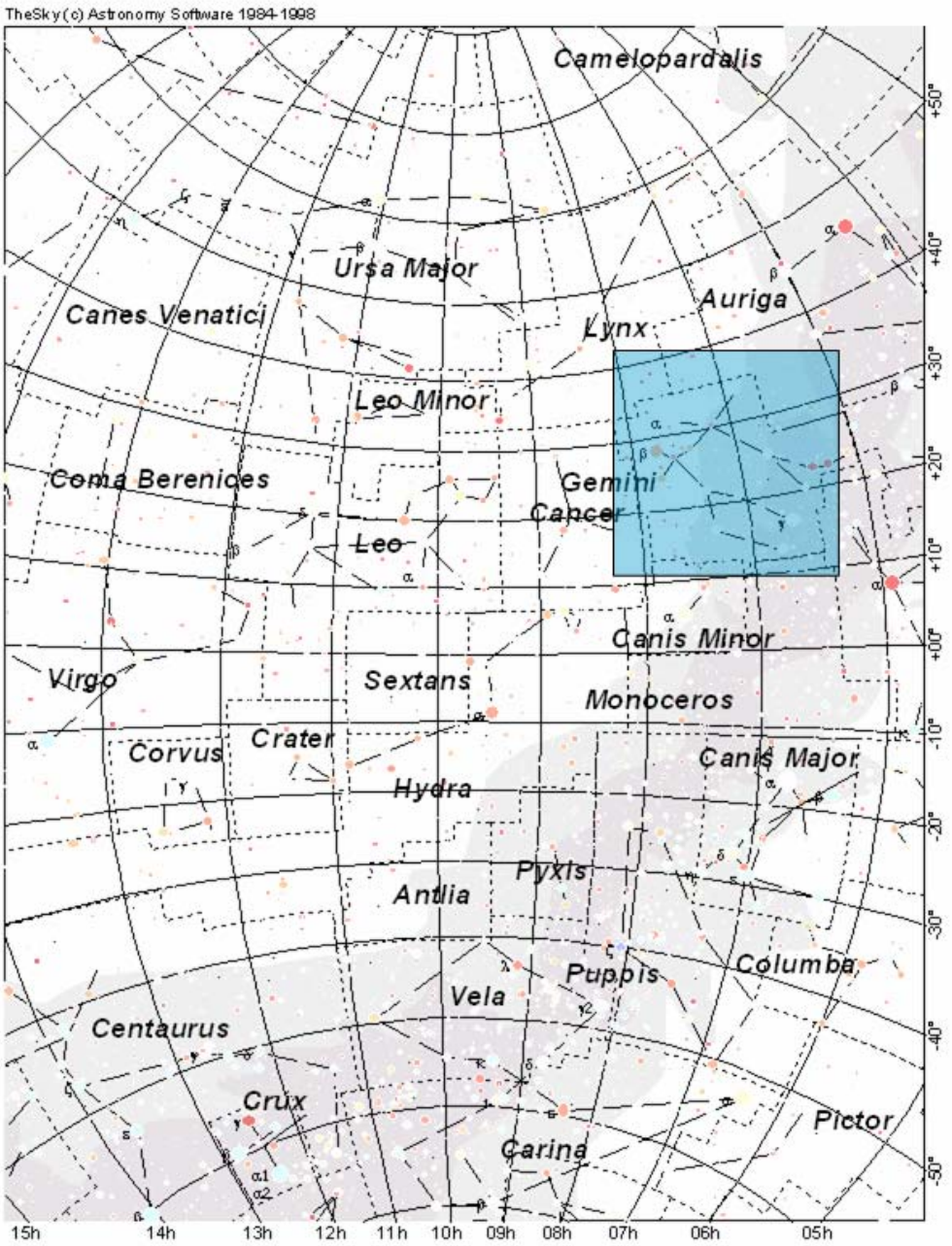
haya nodos libres, se le devolverá la información del nodo que le ha estado asignado.

- **Participación en la simulación.** A partir de este momento, el cliente, está autorizado a realizar eventos sobre la simulación (actualmente enviar mensajes, aunque se podría aumentar en un futuro el número de eventos) y a recibir el estado de esta.
- **Salida de la simulación.** Una vez haya terminado de recoger datos, el alumno cerrara el cliente, con lo que se liberará el nodo que tenia asociado en la simulación y se registrará información sobre el tiempo durante el que ha usado el simulador.

La funcionalidad descrita, se puede observar en el diseño de la pantalla del 'MiniClient':



Anexo 5. Sobre Géminis



Se trata de una de las constelaciones zodiacales más significativas, debido al brillo de sus dos estrellas principales. Su origen se remonta a Mesopotamia, aunque aparece en la literatura occidental con Arato (ver Leo), aunque éste no nos cuenta nada sobre el origen de esta constelación. Posteriormente, Eratóstenes identificaría los gemelos con Cástor y Pólux, mito que se ha conservado hasta la actualidad:

> Dicen que son los Dioscuros, que nacieron y se criaron en la región de Laconia, superando a todo el mundo en su amor fraternal, pues jamás disputaron entre sí ni por el mando ni por ningún otro motivo. Zeus quiso recompensar este estupendo testimonio de fraternidad, los denominó Géminis y los ubicó a ambos en el firmamento. El que se encuentra a continuación de Cáncer tiene una estrella brillante sobre la cabeza y otra también muy luminosa sobre cada hombro; otra sobre el codo derecho y una más en la mano derecha, una en cada rodilla y una más en cada pie. Suman un total de nueve. Su hermano, que está junto, tiene una estrella brillante sobre la cabeza, otra también de intenso brillo sobre el hombro izquierdo, otra en cada tetilla, una sobre el codo izquierdo, otra en el extremo de la mano, una sobre la rodilla izquierda, una en cada pie y otra más debajo del pie izquierdo, que se llama Antepié. Suman en total diez.
>
> Catasterismos

Los Dioscuros (Διόσκουροι, "hijos de Zeus") eran Cástor y Pólux, hijos de la unión de Zeus con Leda (hija de Testio, rey de Etolia), para lo cual el dios se transformó en cisne. Leda puso dos huevos, de uno de los cuales nacieron Pólux y Helena, hijos de Zeus y por lo tanto inmortales, mientras del otro, fruto de la unión con su marido Tindáreo (de ahí que a veces se les denominase Tindárides), nacieron Cástor y Clitemestra, mortales, aunque otras versiones señalaban a ambos como hijos de Zeus. Los dos hermanos

se criaron en Esparta, destacando Cástor en el manejo de las armas y la doma de caballos, mientras que Pólux lo hacía en el pugilato. Como héroes dóricos por excelencia participan en numerosas aventuras, rescatando a su hermana Helena tras su rapto por Teseo y uniéndose a Jasón en la expedición de los Argonautas, enfrentándose al rey de los bébrices. Posteriormente, una lucha homicida contra dos primos suyos, ocasionada según unas versiones ocasionada debido a una lucha por sus prometidas, las Leucípides, hijas de su tío Leucipo (hermano de Tindáreo), y según otras, por una disputa de ganado, provocó la muerte del mortal Cástor. Pólux fue llevado por Zeus al Olimpo, pero se negó a permanecer allí mientras su hermano estuviese en los infiernos, por lo que el padre de todos los dioses les ofreció que se turnasen un día de cada dos en el Olimpo. Posteriormente fueron considerados protectores de los marinos debido a su participación en la expedición de los argonautas (en la antigüedad se identificaba al fuego de San Telmo, que aparece en los mástiles de los buques, con los Gemelos). En Grecia fueron reverenciados especialmente en Esparta, pero alcanzaron su mayor fama en Roma, donde fueron considerados los protectores de la Ciudad Eterna y símbolos del amor fraternal, sin duda haciendo referencia a los hermanos fundadores de Roma.

Esta constelación tiene un origen claramente mesopotámico, pues en las tablas Mul-Apin aparece como Mas-tab-ba-gal-gal, en sumerio "Los Grandes Gemelos". Esta denominación parece proceder sin duda del nombre dado a las dos grandes montañas (o bien una grande con dos cimas) situadas en el límite del mundo conocido, según la cosmografía sumeria, "Las Grandes Montañas" o "Los Grandes Gemelos". Estos montes estaban defendidos por los legendarios Hombres-Escorpión, y formaban un largo desfiladero por el que el sol salía todos los días de debajo de la Tierra desde un lugar situado más al Este. Tienen un papel destacado en la Epopeya de Gilgamesh, la obra literaria más antigua de la humanidad, ya

que el héroe tiene que atravesarlos en su busca de la inmortalidad, que le llevará a visitar a Utanapishtin, el Noé sumerio:

El nombre de esta Montaña
Era Los Gemelos.
Cuando llegó
A los Montes Gemelos,
Que protegen cada día
El itinerario del Sol,
Cuyas cimas
Tocan la bóveda celeste
Y cuyos pies, abajo,
Alcanzan al Infierno,
Su entrada la defendían
Unos Hombres-Escorpión.
Inspiraban ellos un
imponente terror.
Su sola visión era la Muerte
Su espantoso brillo
sobrenatural
Cubría estas Montañas;
El itinerario del Sol.

Epopeya de Gilgamesh (traducción de J. Bottéro), Versión Ninivita, Tablilla IX, II, 1-9

Otros autores han querido ver en estos grandes gemelos una catasterización de Gilgamesh y su amigo Enkidu, aunque no parece que haya mucha base para esta afirmación. Otras fuentes babilónicas posteriores identificaban a los grandes gemelos con los dioses Lugalgirra y Meslamtea.

En la época neobabilónica recibiría el nombre de Simanu, junto con Orión. Esta claro que los griegos adaptaron esta constelación zodiacal de Mesopotamia, pues como podemos ver en los Fenómenos, no tenían ningún mito asociado a ésta. Posteriormente, identificarían a esta constelación con los gemelos más famosos de la mitología griega, aunque no sería la única identificación, pues Ptolomeo veía en esta constelación a Apolo (junto a Tauro)

Posteriormente, los árabes denominarían a esta constelación Al taw'aman, "Los Gemelos" o Al Burj al Jauza, "la Constelación de los Gemelos". De todas formas, aparecen en algunos mapas árabes medievales como dos pavos reales.

Nombres de las estrellas:

- α Gem (1.59m): Cástor o Castor, "el Jinete" en griego. Fue denominada por los árabes Al Ra's al Taw'am al Mukaddim, "la Cabeza del Gemelo más Próximo". También se le llamó Apolo.
- β Gem (1.16m): Pólux, Pollux o Polideuces, del griego Polluces o Polledeuces, "el Boxeador" o "el Pugilista". También Hércules.
- γ Gem (1.93m): Alhena o Almeisam. Aparentemente proviene del nombre árabe al-Han'ah dado a la marca que se hace a los camellos, pues γ, μ, ν, η y ξ se veían como un camello.
- δ Gem (3.51m): Wasat, "medio", de Wasat as-Sama', "la mitad del cielo". También Melucta, Melouda.
- ε Gem (2.98): Mebsuta, de Al-Mabsutah, la pata "estirada".
- ζ Gem (4.4-5.2m,v): Mekbuda, de Al-Meqbudah, "retraída", de "pezuña retraída del león"..
- η Gem (3.33m,v): Propus o Tejat Prior; Tejat: "estrella lluviosa".
- μ Gem (2.86m): Tejat Posterior.También Calx.