Anales PANEL'81/12 JAIIO
Sociedad Argentina de informática
e Investigación Operativa. Buenos Aires, 1981

DISEÑO DE UN SISTEMA DE ARCHIVOS DISTRIBUIDO PARA LA RED LOCAL UNIANDES

Edilberto Sánchez
Edgar Ruiz
Roberto Pardo

Departamento de Ingeniería de Sistemas
Universidad de Los Andes
Apartado Aéreo 4976, Bogotá - Colombia.

O B J E T I V O

El objetivo de este artículo es describir las principales decisiones de diseño de un sistema de archivos distribuído para la "Red Uniandes". En el diseño se integran una serie de soluciones a problemas relacionados con este tipo de sistemas. El artículo describe la arquitectura del sistema, estructura de nombres, directorio, lenguaje de trabajo y "software" (1) de control de concurrencia.

PALABRAS CLAVES: procesamiento distribuído, sistema de archivos distribuido, datos replicados, transacción, atomicidad.

(1) En el artículo se utilizan los términos del inglés "software", "hardware", "lock", "host", "hashing", "livelock","deadlock", "buffer", por no tener una traducción ampliamente utilizada en español.

1. INTRODUCCION

En este artículo se describen las principales decisiones de diseño de un sistema de archivos distribuído para la "Red Uniandes". Otros artículos (ARAU80) y (GAIT80), describen los aspectos de Software y Hardware de comunicación de la red.

La Red Uniandes es una red experimental -actualmente en fase de diseño- que nació como un esfuerzo conjunto entre los departamentos de Ingeniería Eléctrica y Sistemas de la Universidad de los Andes. Uno de los objetivos del proyecto es crear una infraestructura de bajo costo para desarrollo experimental de software distribuído. El esquema general de la red es el de un conjunto de nodos heterogéneos que a través de interfases se interconectan al compartir un medio de transmisión común.

Tradicionalmente los sistemas de archivos se han utilizado para manipular y almacenar información en una sóla máquina, es decir han sido centralizados. Aún bajo este esquema existen problemas de diseño interesantes como: decidir sobre la estructura de nombres y directorios, incluír mecanismos de confiabilidad, definir las primitivas de la interfase, etc.

Un posible esquema de funcionamiento de un sistema de archivos en una red es presentarle al usuario tantos sistemas de archivos (posiblemente diferentes) como nodos haya en la red. Este esquema sin embargo, es muy inflexible. El problema que se ataca en este artículo, es por el contrario, el de diseño de un sistema de archivos homogéneo a través de la red, donde tanto los programas para manipular archivos como los archivos mismos están repartidos en varios nodos, y es el sistema de archivos el que internamente maneja los problemas asociados con la dispersión.

En la literatura del tema se encuentran unas pocas propuestas de sistemas de archivos distribuídos y sobre todo gran cantidad de algoritmos para resolver problemas que pueden o no atacarse en este contexto, como control de concurrencia, atomicidad, confiabilidad, etc. Los objetivos principales del sistema de archivos distribuído aquí descrito son: 1) integrar en un sólo sistema una serie de soluciones a problemas varios relacionados con este tipo de sistemas, y 2) profundizar en detalles de soluciones a problemas que poco se mencionan al describir estos sitemas.

El artículo está dividido en ocho secciones. En las dos primeras secciones se aclaran algunos conceptos básicos y se mencionan algunos sistemas relacionados. Las otras seis secciones discuten el diseño del sistema de archivos para la red. La sección 4 discute el medio ambiente e introduce ejemplos para motivar el uso del sistema y para ilustrar sus funciones. La siguiente sección -la 5- describe el sistema propuesto partiendo de una arquitectura de niveles de donde se deducen los módulos internos (y sus relaciones) para la realización de las funciones. En la sección 6 se explican brevemente los mecanismos de control de concurrencia usados cuando se desea proveer atomicidad (confiabilidad) de transacciones y replicación de datos. Finalmente en la sección 7 se dan recomendaciones de implantación y en la sección 8 se mencionan algunas conclusiones. Más detalles sobre el diseño se pueden encontrar en (SANC80).

2. CONCEPTOS BASICOS

A continuación se definen una serie de términos y conceptos que se usarán a través del artículo.

Sistema de Archivos: Es el conjunto de Software del sistema operacional que se ocupa del manejo y organización tanto lógica como física de la información. Este sistema como mínimo crea un mecanismo para nombrar los archivos, genera y usa información sobre los archivos mismos (directorios), y provee ciertas primitivas para que los usuarios y/o aplicaciones manipulen los archivos. Adicionalmente el sistema puede soportar múltiples organizaciones de archivos, mecanismos de protección y confiabilidad, etc.

Sistema de Archivos Centralizado: Es un sistema de archivos en el que tanto el software para manejo de archivos como los archivos mismos están en un sólo nodo. Estos sistemas pueden ser o no ser usados en ambiente de red.

Sistema de Archivos Descentralizado: Es un sistema de archivos en el que la información puede estar ubicada en diferentes nodos y manipulada independientemente. Sin embargo, el usuario debe coordinar el acceso a cada uno de los sistemas de archivos locales que necesite. Es similar a un conjunto de sistemas de archivos centralizados repartidos sobre diferentes nodos de una red.

Sistema de Archivos Distribuído: Es un sistema de archivos que tiene las mismas características de un sistema de archivos descentralizado pero que realiza sus funciones por medio de la cooperación e interacción entre los diferentes sistemas de archivos en la red. Este sistema coordina automáticamente las referencias a diferentes nodos. Este tipo de sistema, con respecto al descentralizado, es más fácil de usar y evita el mal uso que el usuario podría hacer sobre él. Con respecto al centralizado este sistema puede ser más confiable, más eficiente (en términos de retrasos de comunicación y throughput) y más natural ya que la información puede estar cerca de donde se genera y usa.

Directorio: Es un archivo (o un conjunto de archivos) que contiene información sobre los archivos mismos. Por ejemplo contiene información sobre los dispositivos donde están los archivos, sobre su organización, sobre sus características físicas, etc. El sistema descrito en este artículo tiene un directorio que adicionalmente contiene información sobre los nodos de la red donde residen los archivos.

Acción: Es una llamada directa de acceso (lectura, escritura) a un archivo que cuando es local, o sea cuando se ejecuta sobre un archivo del mismo nodo, satisface condiciones de serialización y atomicidad.

Transacción: Es la unidad lógica de trabajo que consta de una o más acciones; es la llamada externa de un usuario al sistema de archivos. Si todas las acciones de la transacción son locales, la transacción es centralizada. Si una o más acciones de la transacción se refieren a archivos en diferentes nodos, la transacción es distribuída.

Propiedad de Atomicidad: Garantiza que todas las escrituras se hacen o ninguna se hace, independientemente de fallas. A nivel de acción, la atomicidad se refiere a acciones de escritura (o se hace o no se hace). A nivel de transacción, la atomicidad se refiere a un conjunto de acciones de escritura locales o remotas (o se hacen todas o ninguna se hace).

Propiedad de Consistencia: Garantiza que la concurrencia se efectúe correctamente. O sea, si se están ejecutando acciones intercaladas de diferentes transacciones (es decir transacciones concurrentes), cada transacción ve un panorama consistente de los datos como si las transacciones fueran ejecutadas una a la vez (serialmente). Cuando hay copias de archivos existen dos tipos de consistencia: a) Consistencia Interna: el resultado de acciones intercaladas es equivalente a un despacho serial de las transacciones, b) Consistencia Mutua: al suspender las actualizaciones las copias eventualmente convergen a valores idénticos.

Transacción Comprometida: Es un estado del progreso de ejecución de una transacción en el cual se decide que los efectos de su ejecución eventualmente se harán permanentes.

Propiedad de Confiabilidad: Garantiza que aunque ocurran fallas las transacciones comprometidas se lleven a cabo.

Propiedad de Recuperación: Garantiza que los efectos de las transacciones comprometidas se lleven a cabo y los de las otras (las no comprometidas) se deshagan luego de la recuperación de una falla.

3. TRABAJOS RELACIONADOS

En esta sección se describen algunos sistemas relacionados como RSEXEC - (THOM79) y WFS (SWIN79) .

RSEXEC (The Resource Sharing Executive)

Es un sistema de archivos distribuído que crea un medio ambiente el cual facilita compartir recursos de computación y almacenar información entre computadores ("host") de la red ARPA (DAVI79), de tal forma que los grandes computadores "aumentan" los recursos de los pequeños. Entre los principales comandos que provee se encuentran: WHERE y LINK que son usados para establecer diálogo en línea con otro usuario, comandos de configuración BIND, LIST y APPEND. (Por ejemplo: BIND (dispositivo) IMPRESORA (a nodo) SISTEMAS).

El modelo tomado por el RSEXEC mantiene información acerca de los archivos no locales más referenciados por un usuario y adquiere información de estos por medio de programas servidores remotos cuando es necesario. RSEXEC tiene una estructura de nombres de tipo jerárquico, por lo tanto para obtener una configuración determinada es necesario fijar los nombres de archivo, dispositivos y host los cuales son transformados en direcciones físicas. Esta función se realiza por medio de programas servidores. El usuario tiene la facilitad de manejar los archivos en forma compacta debido a las características de los comandos. No tiene problemas con el olvido de nombres de los archivos ya que los puede ubicar en su "contorno".

WFS (A Simple Shared File System for a Distributed Environment)

Es un servicio de archivos compartidos que opera en un medio ambiente local de computación distribuída implantado en ETHERNET (METC76). El sistema provee un conjunto conciso de operaciones, entre las cuales hay operaciones para :a) leer y escribir sobre páginas de archivos; b) asignar y liberar páginas de archivos; c) obtener y modificar propiedades de archi-

vos y d) realizar actividades de mantenimiento al sistema. Las operaciones que más se usan son las de lectura y escritura de páginas, las cuales pasan como parámetro la identificación del archivo (FID) y el número de página; por ejemplo: Readpage (fid, pagenum). Las operaciones están diseñadas de tal forma que si se repite una acción antes de confirmarla, tendrá el mismo efecto que una sola.

El directorio de WFS es implantado como una tabla de "hashing" de tamaño fijo, contigua y en una dirección de disco conocida; adicionalmente está duplicado en los servidores los cuales son computadores con gran capacidad de almacenamiento. La estructura de nombres es de tipo planar compuesta de: a) identificadores de los archivos que existen en la red; b) páginas de transformación de identificador lógico a dirección física de cada archivo, y c) archivos divididos en páginas de tamaño fijo. Esta estructura facilita la adición de un mecanismo de recuperación y de control de transacciones debido a la fragmentación del archivo en páginas.

4. MEDIO AMBIENTE

SAD (se usa esta abreviación en adelante para referenciar el sistema propuesto) utiliza las facilidades que le provee el protocolo para comunicación entre procesos para establecer canales virtuales entre dos procesos remotos por medio del cual se envían mensajes en forma confiable (GAIT80). El protocolo de transferencia de archivos permite expandir operaciones sobre un archivo local a nivel de red, como crear, borrar, transferir, etc.

El usuario del sistema de archivos distribuído invoca las transacciones que se encuentran almacenadas en una librería de transacciones. Otro tipo de usuario más familiarizado con el sistema elabora las transacciones mismas ya sea mediante un lenguaje específico o haciendo uso directo de las primitivas que le provee el sistema.

El siguiente ejemplo ilustra el uso de transacciones y algunos problemas que debe solucionar SAD

```
/* Se otorga un préstamo en base al valor e ingresos */
/* del empleado                                      */

Trans: PRESTAMO_A_EMPLEADO (#_EMPLEADO, VALOR, OTORGADO)
       P1: Lea sueldo, bonificaciones, deducciones de
           #_EMPLEADO.
           Calcule ingresos
           if ingresos >= 20% de VALOR then   OTORGADO = true
                                      else    OTORGADO = false.

end trans

/* Se elabora un transpaso de fondos cumpliendo con */
/* una condición de mínimo ingreso predeterminada   */

Trans: ABONO_A_FONDO_DE_EMPLEADOS (#_EMPLEADO, CANTIDAD)
       D1:  Lea ingresos de # EMPLEADO. Retire CANTIDAD a #_EMPLEADO
```

```
D2: Lea ingresos de fondo.
    Abone CANTIDAD a ingresos de fondo.
    if ingresos de #_EMPLEADO< mínimo_ingreso
       then escriba viejo_ingreso de #_EMPLEADO
                    viejo_ingreso de fondo.
end trans
```

En el * ejemplo se supone que los archivos usados se encuentran en nodos diferentes al nodo donde se lleva a cabo la transacción. Cuando las transacciones se ejecutan se pueden presentar varios problemas. Por ejemplo, al ejecutarlas en forma serial si ocurre una falla en el nodo donde están los items del fondo de empleados luego de ejecutar la acción D1, se haría un retiro inválido al empleado al no realizarse D2. Al ejecutarlas en forma concurrente si P1 se efectúa después de D1 y el resultado de la acción D3 es dejar los viejos valores, la transacción Préstamo_A_Empleado ejecuta con un dato falso. El prevenir este tipo de problemas es un objetivo de SAD.

5. DESCRIPCION DEL SISTEMA

5.1 Características

Las principales características de SAD son:

-Los archivos están divididos en páginas para facilitar el manejo de buffers, apuntadores y registros (relativos a páginas).
-Los servidores coordinan las transacciones del nodo, dan respuesta a los pedidos externos de otros servidores y se encargan de la interfase con el sistema de archivos local.
-Las operaciones sobre los archivos son agrupadas en transacciones.
-El sistema provee un conjunto definido de primitivas adecuadas para la creación de transacciones.
-Existe un mecanismo de control de concurrencia por medio de "locks" a nivel de página y archivo. Los locks tienen asociado un tiempo de inactividad sobre estos items; cuando el tiempo de inactividad se completa, automáticamente se deshacen los locks hechos por la transacción.
-Se provee un mecanismo para mantener consistencia entre las copias remotas de los archivos replicados.
-Los mecanismos de control entre servidores garantizan las propiedades de consistencia, atomicidad, confiabilidad y recuperación.

5.2 Arquitectura

El sistema está estructurado en cuatro niveles. El nivel 0 provee las facilidades básicas para comunicación entre procesos, transferencia de archivos y la interfase con el sistema de archivo local. El nivel 1 borra los límites físicos en cuanto al uso de los archivos, controlando concurrencia, manteniendo atomicidad y asegurando confiabilidad. El nivel 2 corresponde al despachador de transacciones que por medio de la interfase (primitivas) con el nivel 1, coloca en forma ejecutable las transacciones en el contexto de los mecanismos de control. El nivel 3 comprende la aplica-

ción y la interfase con el usuario. Para una mayor comprensión de la arquitectura el lector puede referirse a la figura 1.

Nivel 3

Existen dos tipos de usuarios del sistema: unos como por ejemplo el cajero del Fondo de Empleados que invoca la transacción ABONO_A_FONDO_DE_EMPLEADOS o el jefe de este Fondo que invoca la transacción PRESTAMO_A_EMPLEADO (ver sección 4), quienes solo conocen su efecto o sea solo invocan transacciones; y otros, más familiarizados con el sistema y con la aplicación, quienes elaboran las transacciones.

Nivel 2

Este nivel está conformado por los siguientes módulos: a) Configura Directorio de la Transacción. Provee al usuario que crea la transacción un medio para conocer los archivos disponibles para la elaboración de la transacción y crea un directorio lógico cuando se crean instancias de las transacciones en base a los parámetros provistos por quienes hacen uso de la transacción. b)Librería de Transacciones. Usado para almacenar las transacciones elaboradas, guardando el directorio y las acciones de la transacción (escritas en un lenguaje específico), como P1 y P2 de PRESTAMO_A_EMPLEADO. c) Contexto de la Instancia. Al recibir el pedido de ejecución por parte del usuario, se encarga de crear una instancia de la transacción en base a los parámetros pasados y asociar un descriptor de la instancia que contiene además de otra información un apuntador al directorio lógico de la instancia, conformado por el módulo 2. Extrae las acciones de la transacción de la librería y las pasa al módulo traductor con los cambios involucrados por los parámetros dados por el usuario. d) Traductor, se encarga de traducir a primitivas del sistema las acciones de las transacciones pasadas por el módulo anterior, haciendo uso del directorio lógico de la instancia.

Nivel 1

Este nivel corresponde a los mecanismos de control y se encarga de resolver los problemas de concurrencia, atomicidad (confiabilidad) de transacciones y consistencia de datos replicados cuando se ejecutan transacciones concurrentemente en un medio de computación distribuído.

Se hace uso de archivos de datos e intenciones, que están divididos en páginas. Un descriptor (mapa de páginas) se utiliza para cambiar los datos del archivo. El archivo de intenciones se usa para completar las transacciones suspendidas cuando ha habido una falla. Adicionalmente provee un mecanismo de control de inactividad sobre los items, asociando a cada uno un tiempo máximo. Las anteriores funciones las realiza el servidor de cada nodo, el cual coordina las acciones de las primitivas y los pedidos externos de los servidores remotos.

Nivel 0

Provee las facilidades básicas para comunicación entre procesos, transferencia de archivos y la interfase con el sistema de archivos local. Estas facilidades son operadas por un conjunto definido de primitivas que permiten

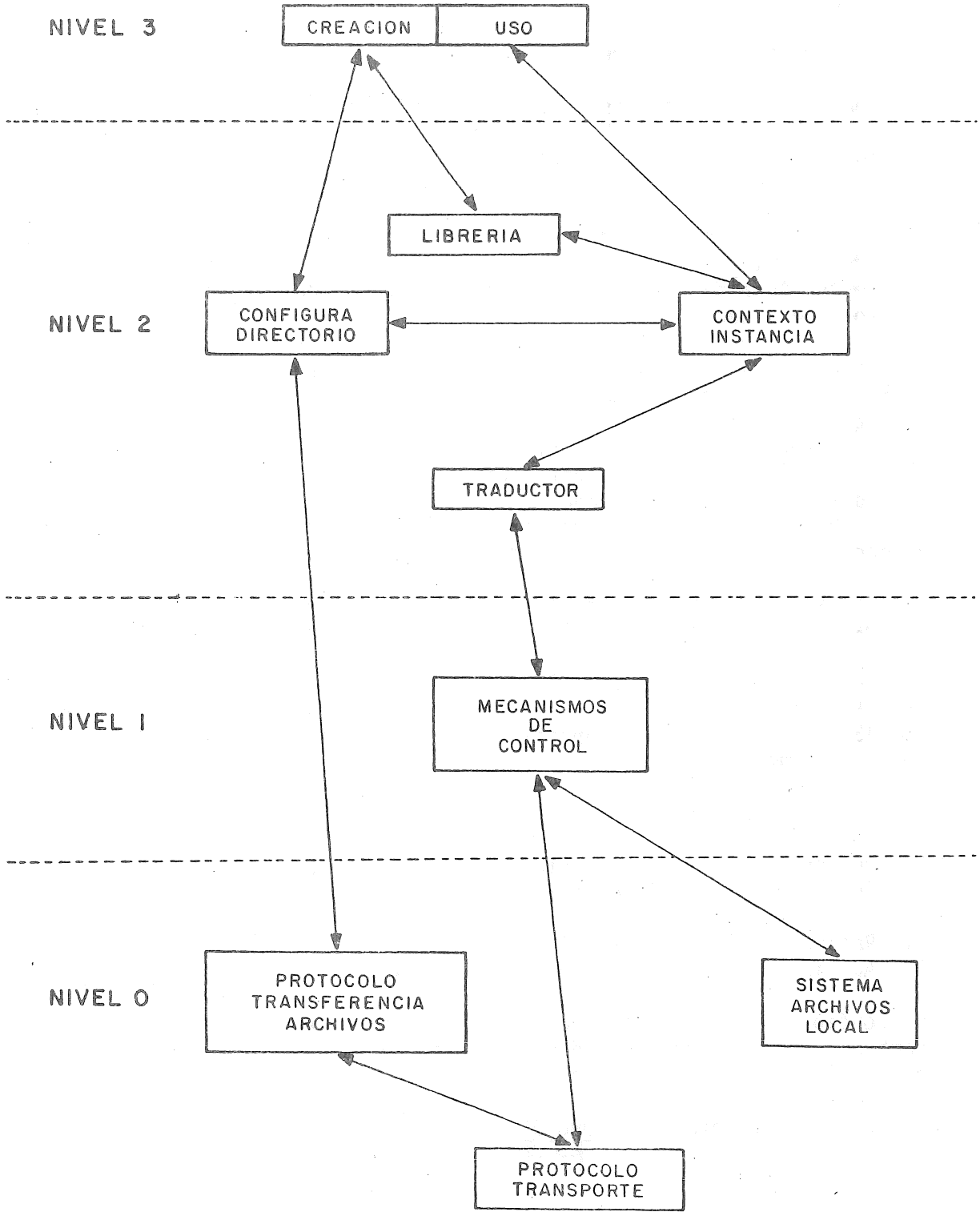


FIGURA 1. ARQUITECTURA DE SAD

establecer comunicación con procesos remotos, intercambiar mensajes confiablemente, transferir archivos en una forma segura y hacer acceso a la información local.

5.3 Estructura de Nombres

Un problema clásico de diseño ya sea en Sistemas de Archivos Centralizados, descentralizados o distribuídos es decidir cual es la estructura de nombres más adecuada, de tal manera que se pueda manipular fácilmente y opere en forma conveniente.

La estructura de nombres de SAD es de tipo jerárquico basada en la expansión de la estructura de árbol que proveen algunos sistemas operacionales. Esta estructura fue escogida para darle modularidad al sistema y permitir la expandibilidad.

La estructura se divide en cuatro niveles de la siguiente forma: a) En el primer nivel se identifican los nodos (computadores) de la red que han sido ligados por medio de la configuración establecida para la transacción. b) En el segundo nivel se identifican los dispositivos de cada computador. c) En el tercer nivel se identifican los archivos que se encuentran en cada dispositivo seleccionado. d) En el cuarto nivel se identifican los mapas de páginas de los archivos de datos que transforman páginas lógicas en direcciones físicas de disco e identifican páginas sin asignar.

La figura 2 describe claramente este tipo de estructura. Un ejemplo que ilustra el funcionamiento de la estructura de nombres es el siguiente:

Un usuario identificado con el código 001 por medio de la transacción Trans: TRASPASO-ESTUDIANTE cambia de facultad al estudiante cuya identificación es 7611432, de Sistemas de Economía. Para ello invoca la transacción en la siguiente forma:

Trans : TRASPASO-ESTUDIANTE (001, 7611432, SISTEMAS, ECONOMIA)

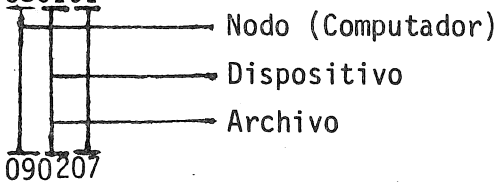
Al crearse la instancia de esta transacción las acciones quedan conformadas sintácticamente de la siguiente manera:

A1: Retire (SISTEMAS, ESTUDIANTES, 7611432)
A2: Adicione (ECONOMIA, ESTUDIANTES, 7611432)

El configurador de contexto en este punto opera en base a la estructura de nombres para conformar la identificación de los archivos, tomando los datos del directorio de la instancia así:

Sistemas.Estudiantes = 080101

Economía.Estudiantes = 090207



Como se observa hasta ahora se tienen tres niveles de la estructura de nombres: computador, dispositivo y archivo. La página no se ha ubicado debido a que falta establecer si el archivo es local o remoto. Sí se supone que la transacción se coordina en el nodo de la facultad de sistemas, por medio de los mecanismos de control se obtiene que la primera acción se debe ejecutar localmente mientras que para la segunda se debe realizar un pedido a un nodo remoto (Facultad de Economía); luego para la primera acción se tendría definido toda la estructura de nombres al invocarse el sistema de archivos local por medio de la primitiva correspondiente, mientras que la estructura de la segunda es completada por el sistema de archivos del nodo remoto.

5.4 Directorio

Otro problema interesante de diseño es el de directorio. En un sistema distribuído el directorio debe tener una localización transparente para el usuario, debe recopilar toda la información necesario para llevar a cabo las transacciones operadas sobre la red, y debe ser eficiente (tener un buen tiempo de respuesta) y tener un mecanismo sencillo de mantenimiento.

Teniendo en cuenta que el medio en que se va a implantar este sistema consta de pocos nodos y tratando de cumplir con las condiciones anteriormente expuestas, se unifican los directorios de cada nodo conformando un directorio global el cual contiene información sobre: los nodos (Computadores) los dispositivos de almacenamiento y los archivos en cada unidad.

También el directorio global contiene algunas características de los archivos, como por ejemplo, si están replicados o no, el código de protección, etc. Este directorio es replicado en los nodos que puedan mantenerlo ya sea por capacidad disponibilidad o por decisiones administrativas. Aquellos nodos que por una u otra razón no tienen un replicado del directorio global, tienen una tabla que les permite ubicar cuáles nodos cercanos lo contienen.

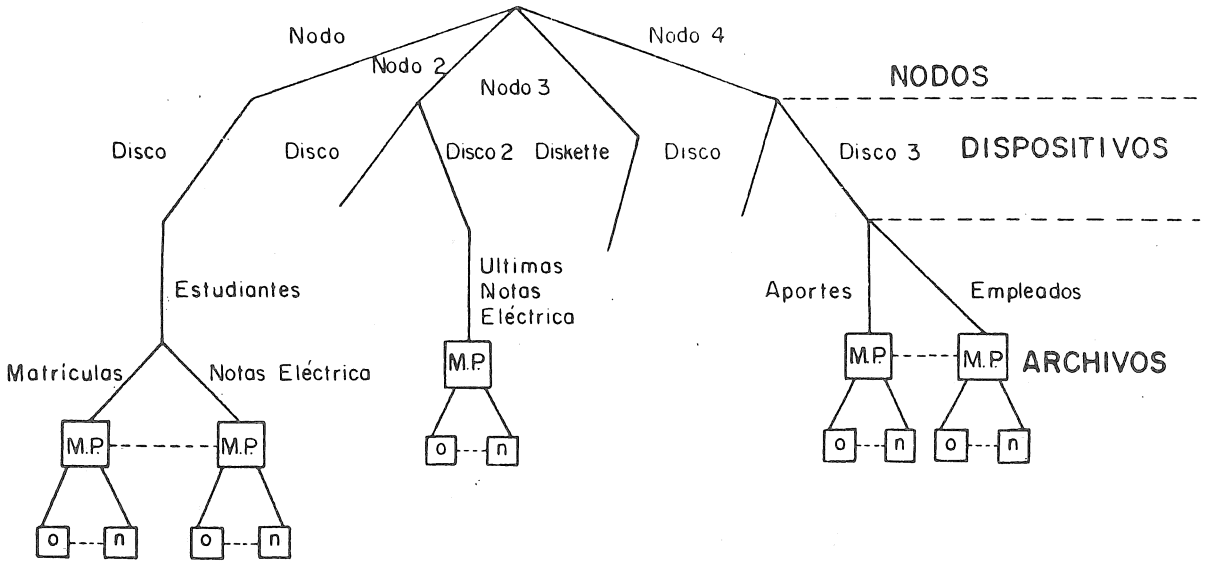
Como se observa en la figura 3 el directorio es de tipo jerárquico e implementado como listas con apuntadores. Este directorio es usado por el sistema en los siguientes casos: a) Cuando un usuario lo necesite para elaborar una nueva transacción, b) Cuando se va a crear el directorio lógico para la instancia de una transacción que se carga desde librería y cuyos archivos de trabajo son asociados de acuerdo a los parámetros del usuario.

Los siguientes ejemplos ilustran la diferencia entre el uso estático y dinámico de los archivos por medio del módulo Configura Directorio de Transacción.

Ejemplo 1: Supongamos que los empleados de una entidad están registrados con todos sus datos en un sólo archivo, este sería fijo en el directorio lógico de una transacción como:

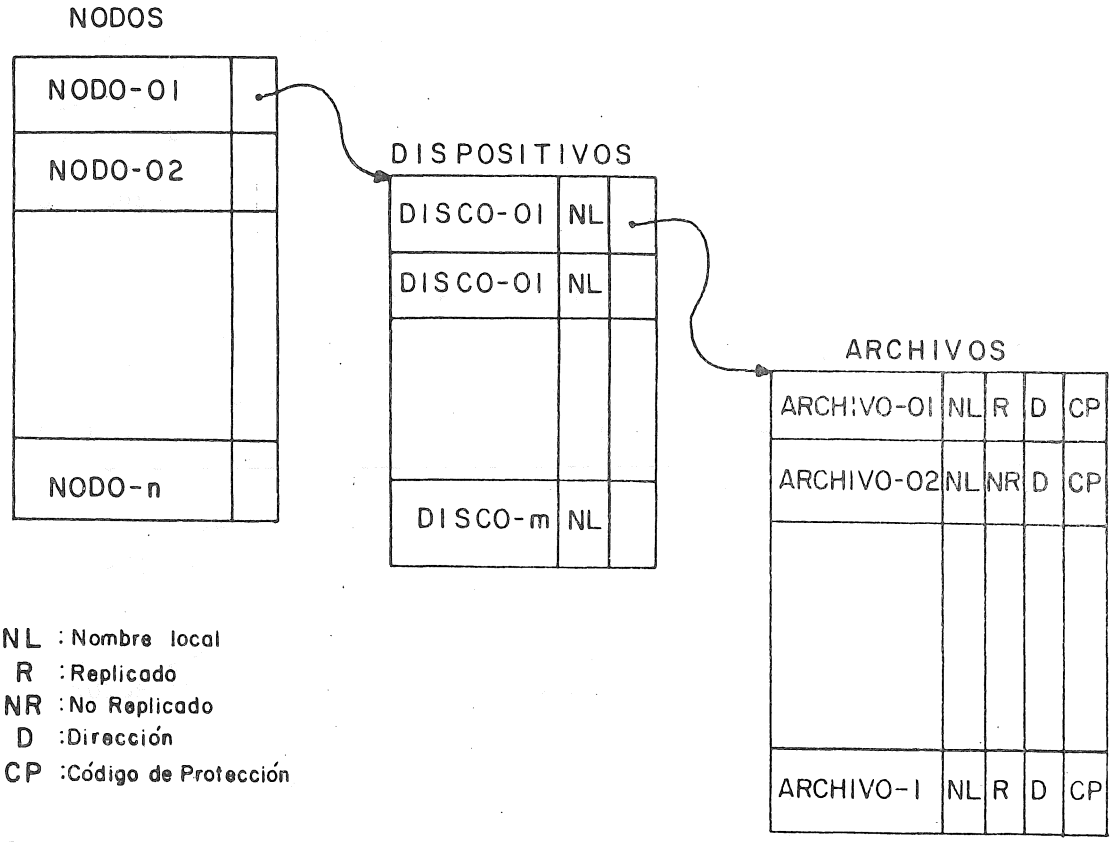
Trans: ABONO_A_FONDO_EMPLEADOS (#_EMPLEADO, CANTIDAD)

El descriptor asociado a esta transacción sería:



M.P Mapa de Páginas

Fig. 2. Estructura de Nombres de SAD



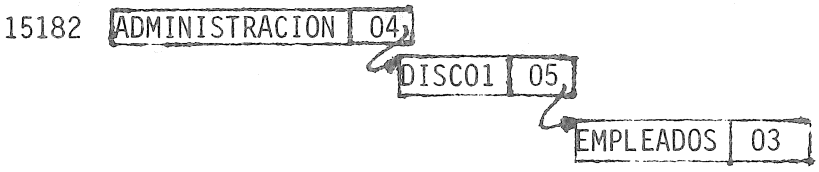
NL : Nombre local
R : Replicado
NR : No Replicado
D : Dirección
CP : Código de Protección

Fig. : 3 Componentes del Directorio Global de SAD

DESCRIPTOR

Nombre: ABONO_FONDO_EMPLEADOS
Número Transacción: _____
Prioridad Externa : 10
Código de Protección: 05
Entrada al Directorio: 15182

DIRECTORIO LOGICO ABONO_A_FONDO_EMPLEADOS



Ejemplo 2: Supongamos que en la Universidad cada facultad tiene un archivo de estudiantes, donde cada facultad corresponde a un nodo de la red. En una transacción de traspaso de estudiante los archivos pueden variar según el caso así:

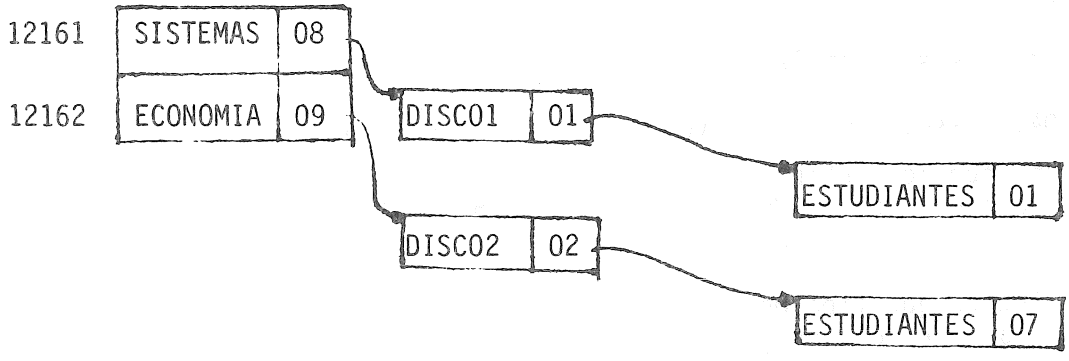
Trans: TRASPASO_ESTUDIANTE (001, 7611432, SISTEMAS, ECONOMIA)

El descriptor asociado a esta transacción sería:

DESCRIPTOR

Nombre: TRASPASO-ESTUDIANTE
Número Transacción: _____
Prioridad Externa: 15
Código de Protección: 01
Entrada al Directorio: 12161

DIRECTORIO LOGICO TRASPASO-ESTUDIANTE



El valor del número de la transacción se coloca cuando los mecanismos de control empiezan a operar sobre la transacción traducida. En el primer ejemplo el directorio lógico por ser estático es cargado directamente de librería, en tanto que, en el segundo ejemplo es conformado mediante la interacción con alguno de los nodos que contiene el directorio global.

5.5. Lenguaje de Trabajo

La interfase entre el nivel 2 y el nivel 1 se realiza por medio de un conjunto definido de primitivas, que para su operación envuelven un intercambio de mensajes entre servidores cuando el archivo referenciado es remoto, y una comunicación directa entre el servidor y el sistema de archivos local cuando recibe un pedido bien sea externo o interno a un archivo del nodo.

Para estandarizar el acceso a los archivos de la red se establece un identificador de archivo (IDA), siendo este un número entero compuesto de la concatenación del número del nodo, número de unidad y número de archivo en la unidad. Su longitud en bits depende de la cantidad de aquellos en la red.

Primitivas

Algunas de estas primitivas son:

LEA PAGINA (IDA, tipo, número, área, desplazamiento, #_transacción).

Esta primitiva tiene como función permitir el acceso a una página de un archivo. Los parámetros indican lo siguiente:

Tipo: Indica si es la llave de un registro o el número de una página

Número: Corresponde al valor de la llave o número de la página.

Area: Identifica el número de buffer donde se almacena la página leída. Si el Tipo es una llave se trae toda la página donde se encuentra el registro referenciado.

Desplazamiento: Localiza el registro dentro de la página.

#_Transacción. Identifica el número de transacción respecto al nodo en que se usa la primitiva.

ESCRIBA PAGINA (IDA, número, área, #_transacción)

Tiene como función escribir una página de un archivo.

Los parámetros son similares a los de LEA-PAGINA.

LOCK (IDA, clase, número, llave_acceso, #_transacción)

Tiene como función inhibir el acceso a otras transacciones hasta que no se realiceuna operación de UNLOCK o que se deshaga el LOCK por inactividad. Los parámetros indican lo siguiente:

Clase. Indica qué tipo de lock se va a realizar: si es a nivel de página o a nivel de archivo, y si va a ser lectura o escritura.

Número. Es un número de llave o página por medio del cual se identifica la página a inhibir.

Llave_Acceso. Es retornadà por el servidor para ser usada en operaciones subsecuentes (al utilizar las primitivas LEA o ESCRIBA).

UNLOCK (IDA, clase, llave_acceso, número #_transacción)

La descripción de los parámetros es similar a los de LOCK. Su función es desbloquear el Item para permitir el acceso a otras transacciones que lo necesiten.

ASIGNE PAGINA (IDA, #_ página)

Su función es retornar el número de una página libre identificada en el mapa de páginas .

RETIRE PAGINA (IDA, #_página)

Tiene como función colocar en la lista de libres la página especificada.

Se ilustra el uso de algunas de estas primitivas con el ejemplo de la transacción TRASPASO-ESTUDIANTE, El nódulo traductor forma un programa compuesto de instrucciones en lenguaje de trabajo y primitivas del sistema, partiendo de la instancia de la transacción que le provee el módulo que configura el contexto.
Así, la transacción lista para ejecutar es:

```
EMPIECE TRANSACCION
LOCK (080101,PAGINA,7611432, ____,_____)
LOCK (090207,PAGINA,7611432, ____,_____)
LEA-PAGINA(080101,PAGINA,7611432,BUFFER3,____)
ESCRIBA-PAGINA(080101,____,BUFER,_____)
LEA-PAGINA(090207,PAGINA,7611432,BUFER5,_____, _____)
        :
ESCRIBA-PAGINA(090207, _____,BUFER5,_____)
        :
UNLOCK(080101,PAGINA,____,____,_____)
        :
FIN TRANSACCION
```

Los parámetros que aparecen con guión (_____) toman su valor en la fase de ejecución cuando entran a operar los mecanismos de control. Los puntos se usan para indicar las operaciones de la transacción que no son traducidas a primitivas del sistema por no tener relación directa con el acceso a la información.

6. MECANISMOS DE CONTROL DE CONCURRENCIA

A continuación se describe en prosa un resúmen del algoritmo básico que implanta los mecanismos de control de concurrencia cuando hay replicación y se desea garantizar atomicidad (confiabilidad) de transacciones. La descripción algoritmica aparece en (SANC80).

6.1 Caracteristicas

Existen dos problemas que se presentan potencialmente en un medio ambiente donde se ejecutan procesos concurrentemente, estos son: "livelock", el cual

consiste en un número ilimitado de intentos para lograr un lock sobre un archivo o página del archivo; y "deadlock", el cual corresponde a una función en la que cada una de las transacciones de un conjunto S de dos o más transacciones están esperando hacer un lock sobre una página o archivo inhibido por alguna otra transacción en el conjunto S (ULLM80).

Una estrategia FIFO (first-input-first-output) elimina livelocks. Gran variedad de soluciones han sido propuestas por los diseñadores de sistemas operacionales para resolver problemas de deadlock (BRIN73), (DIJK68), por lo tanto no se discute en el artículo para hacer énfasis en los mecanismos de control de concurrencia citados anteriormente.

Facilidades de los Nodos

Se supone que los nodos proveen acceso aleatorio a un sólo sector (página) a la vez. El sistema propuesto maneja en una forma lógica los locks a nivel de página elaborados por las transacciones, liberándolos ya sea por mandato explícito o implícitamente en caso de sobrepasar un tiempo determinado de inactividad sobre la página o archivo.

Tipos de Archivos.

Se tienen dos tipos de archivos: archivos de datos y archivos de intenciones. Los archivos de datos contienen la información operada por las transacciones, están estructurados de tal forma que una modificación lo convierte en una nueva versión que puede tener muchas páginas cambiadas.

Se provee un nivel de transformación entre las páginas lógicas y las páginas físicas. La primera página física de cada archivo de datos es un descriptor del archivo que contiene un mapa de páginas indicando la página física donde la página lógica está almacenada. Adicional al mapa de páginas, existe una lista de páginas reales libres y una variable por cada página que indica la transacción, si hay alguna, en la que esté envuelta el archivo o página.

Los archivos de intenciones tienen una variable de estado, un número de transacción, una lista de cambios y una secuencia de los descriptores de los archivos modificados por la transacción. La transacción puede estar en alguno de los tres estados siguientes: iniciada, comprometida o completa. Cuando la transacción se encuentra en su estado de inicio, se la puede abortar; cuando está en el estado comprometida, eventualmente todas las escrituras serán realizadas aún en presencia de fallas hasta que llegue al estado completa. La lista de cambios contiene identificadores de los archivos de datos que han sido modificados en la transacción y apuntadores a copias de sus nuevos descriptores. Este archivo de intenciones está localizado en el nodo que coordina la transacción. Esta estructura es similar a la usada por -- (LAMP79).

6.2 Descripción

A continuación se describen las operaciones más relevantes que se ejecutan en el contexto de los mecanismos de control: Inicie Transacción, lock, lea,

escriba, aborte transacción y fin transacción. Cada una de estas operaciones básicas lleva asociada un procedimiento"Inicie Transacción" asigna un archivo de intenciones y retorna un número de transacción que es pasado como parámetro por los demás procedimientos.

Para visualizar mejor el algoritmo se describe primero suponiendo que no existen replicados como el algoritmo descrito por Paxton (PAXT79) y luego se explican los cambios necesarios.

Cuando el nodo que está coordinando la transacción contiene el archivo referenciado ejecuta directamente la acción de la primitiva, en caso contrario hace un pedido al programa servidor del nodo donde se encuentre el archivo referenciado.

INICIE TRANSACCION

Asigna un archivo de intenciones a la transacción y retorna un número de transacción relativo al nodo, el cual es pasado como parámetro adicional para los demás procedimientos . Este número sirve para identificar las primitivas de las diferentes transacciones cuyas invocaciones pueden venir entre mezcladas (concurrencia).

LOCK

Recibe el IDA y retorna la llave de acceso al archivo para ser usada en los accesos siguientes. Para ello estableci si el archivo es local o remoto, Si es local consulta su directorio y mira si la página o archivo está envuelto en alguna transacción. En tal caso la nueva transacción debe esperar y generar un nuevo pedido hasta que complete un número de intentos preestablecidos. En caso de ser un archivo remoto , en base a la primera componente del IDA se toma el número del nodo el cual se utiliza para establecer comunicación con el servidor de dicho nodo, por medio del Protocolo de Comunicación entre Procesos (PCP). Establecida la comunicación se ensambla un mensaje con el tipo de pedido requerido. El servidor remoto al recibir el pedido de lock verifica si el item referenciado está comprometido en alguna transacción (incompleta por una falla en ese nodo), en cuyo caso invoca el procedimiento de recuperación el cual se encarga de completar la transacción anterior. Cuando se otorga el lock sobre el item se retorna las propiedades de este. Al expirar el tiempo de inactividad sobre un item, automáticamente se deshacen los locks que la transacción había hecho sobre otros items, y se pasa al estado de la transacción a completa sólo si no estaba comprometida.

LEA

En base a las propiedades del item retornadas por el "lock" se verifica si el registro identificado por el parámetro (número) de la primitiva se encuentra en una página leída anteriormente y se retorna el desplazamiento dentro de la página. En caso contrario se genera un pedido de página bien sea el archivo local o remoto, transfiriéndose una nueva página.

ESCRIBA

En base al buffer especificado por la primitiva, se transfiere la página bien

sea al servidor local o remoto dependiendo del IDA; el servidor por medio de la primitiva Asigne_Página localiza una página libre donde se realiza la escritura, luego retorna el número de la página física donde quedó la página transferida para que el coordinador de la transacción actualice el mapa de páginas que le había sido transferido al invocar la primitiva lock.

ABORTE TRANSACCION

Los locks que había hecho la transacción sobre los items de trabajo son liberados, así como las estructuras de datos necesarias para la ejecución de la instancia de la transacción.

FIN TRANSACCION

Dependiendo del número de archivos modificados existen tres casos:

-Si no se modifica ningún archivo (lo cual se detecta mediante un bit de cambio en el descriptor local del archivo en la instancia), se liberan los archivos inhibidos y se cambia el estado de la transacción a completa. Otra posibilidad es dejar que los locks se rompan por tiempo de inactividad.
-Si se modifica un sólo archivo, se liberan los de sólo lectura, se cambia el descriptor del archivo remotor por el descriptor local (enviando la lista de páginas lógicas cambiadas) y se pasa el estado de la transacción a completa.
-Si se modifican más archivos se debe ir creando el archivo de intenciones para prevenir una falla. Se siguen los siguientes pasos:

1- Se liberan todos los items (páginas y archivos) de solo lectura y los que no se modificaron (para ello se utiliza el bit de cambio a nivel de página y archivo).
2- Se hace un pedido de lock local al archivo de intenciones.
3- Se marca cada item modificado con el número de la transacción y el IDA del respectivo archivo de intenciones para la instancia de la transacción.
4- Se copia en el archivo de intenciones la lista de cambios de cada archivo modificado, y se pasa el estado de la transacción a comprometida. El estado de la transacción se puede pasar a comprometida debido a que no hubo problemas con otras transacciones y el servidor remoto permitió marcar el archivo.
5-Se reescribe el nuevo descriptor del archivo (enviando la lista de cambios) y se retira la marca indicando que el item ya no está envuelto en ninguna transacción, luego se libera el lock sobre el item.
6- Se cambia el estado de la transacción a completa y se libera el lock del archivo de intenciones.

Procedimiento de Recuperación

Es invocado cuando un item referenciado por la primitiva Lock está involucrado en una transacción (marcado) o se rompe el lock del archivo por inactividad.

Al invocarse como resultado de la primitiva Lock, se libera el lock del item y se hace un pedido de lock al archivo de intenciones, para luego hacer un

nuevo pedido de lock al archivo de datos. Esto evita posibles deadlocks cuando un procedimiento tiene inhibido los items y otro el archivo de intenciones. Cuando se logra tener el archivo de intenciones sólo se libera hasta que se completa la transacción, aún esperando a que se liberen los locks de los items. Al invocarse como resultado de la inactividad sobre el item su primera acción es tratar de hacer un lock al archivo de intenciones. El archivo de intenciones se utiliza para ver el estado en que quedó la transacción antes de la falla y proceder a completarla en caso necesario.

Tratamiento para Replicados

Como se enunció anteriormente es necesario hacer algunos cambios para describir el algoritmo cuando existen replicados. En la ejecución de una transacción se opera con una de las copias disponible, efectuando los cambios requeridos en las páginas libres. En el procedimiento Fin_Transacción que es cuando se forma el archivo de intenciones, el servidor que coordina la transacción hace un pedido al servidor encargado del archivo para modificarlo, en caso de que existan replicados el servidor encargado espera confirmación de los otros nodos para efectuar los cambios requeridos dándose la posibilidad de abortar la transacción cuando exista conflicto con otras transacciones. Este algoritmo para actualizar copias es similar al descrito por Ellis. - (ELLI77).

Para la toma de decisión los servidores pueden estar en uno de estos estados: primero, que el servidor al que se le esté pidiendo el voto no haya llegado a la fase de formar el archivo de intenciones de la transacción que esté coordinando y segundo, que esté en la fase de formar el archivo de intenciones. La decisión en el primer estado en caso de conflicto es simple puesto que la transacción que hizo el pedido está más adelantada, abortándose la otra. En el segundo estado cuando ambos están formando los archivos de intenciones, se sugieren las siguientes alternativas de decisión en caso de conflicto: a) Se decide por el que haya recolectado un número mayor de confirmaciones según los archivos replicados utilizados (una transacción puede modificar varios archivos y algunos de estos estar replicados) pudiendo efectuar un promedio ponderado de los archivos marcados y no marcados (en esta fase todos los archivos que van a ser modificados quedan marcados); al hacerse el pedido se enviará información sobre los archivos ya marcados; b) Por registros de tiempo (cuenta eventos), para dar prioridad a la más antigua. Para el sistema se escogió la primera debido a que es más sencilla de implantar y provee más información actualizada. Se tiene en cuenta que el pedido para la actualización de un replicado conlleva un tiempo de inactividad sobre el archivo mucho mayor que el normal, debido a que el procedimiento para recolectar "listos" puede ser demorado, además se va formando el archivo de intenciones primero con los que no estén replicados para obtener un mejor rendimiento. Estos cambios se tienen también en cuenta en el procedimiento de recuperación.

7. NOTAS DE IMPLANTACION

En esta sección se describe un modelo para la implantación SAD. El modelo está formado por una colección de programas servidores que corren en algunos nodos de la red escogidos a priori.

Servidor

Para el soporte de programas modulares (transacciones) se utiliza una construcción llamada servidor. El servidor está compuesto de objetos (enteros, arreglos, colas, archivos, páginas de archivos, procedimientos) y procesos. Los procesos de un servidor pueden compartir objetos, comunicarse con otros procesos del mismo servidor por medio de objetos compartidos y comunicarse con procesos en otros servidores únicamente por envio de mensajes. Durante la ejecución de una trasacción la población de servidores serán creados y servidores existentes pueden autodestruirse cuando no tengan más funciones que realizar. Un servidor se crea como consecuencia de un EMPIECE_TRANSACCION o de un pedido hecho por un servidor remoto (Protocolo para iniciar conexión). El servidor es responsable de su espacio direcccionable, el manejo de almacenamiento y los procesos de recuperación que son iniciados después de una falla (descritos en la sección 6). Ver figura 4. La estructura del servidor utilizada en este modelo es similar a la usada para guardianes por Barbara Liskov (LISK79).

Organización Servidor

La comunicación del servidor con su medio ambiente es por medio de pedidos generados en base a las primitivas del sistema. Como respuesta a un pedido crea un proceso para manejarlo (Figura 5).

El proceso creado se sincroniza con los otros por medio de un MONITOR para asegurar que sólo uno manipule los datos compartidos (table de locks, cola de mensajes, tabla de estado de los servidores, directorio) (HOAR74).

Procesos

Una de las funciones más importantes del servidor es satisfacer los pedidos de las transacciones invocadas por el usuario. Estos pueden ser satisfechos directamente o por medio de la interacción con los servidores remotos. El servidor lleva a cabo sus funciones por medio de los siguientes procesos:

1- El Proceso ESER (Estado del Servidor) se encarga de mantener la información de estado acerca de los programas servidores remotos. Este proceso mantiene actualizados los registros de estado de los programas servidores de otros nodos por intercambio de información de los procesos ESER de cada nodo. Cada proceso ESER reliza periódicamente un "broadcast" de su propia información de estado.
2- Por cada pedido (interno o externo) que tiene asociada la ejecución de una primitiva se genera un proceso PRIM que es el encargado de llevarla a cabo. Los procesos de este tipo fueron descritos en la sección 6, donde las estructuras de datos como archivos de intenciones y de datos son compartidos.
3- Los seudo-operandos EMPIECE-TRANSACCION, ABORTE-TRANSACCION YFIN-TRANSACCION general procesos de control de las transacciones como se describió en la sección anterior.

Monitor

El monitor es el encargado de: sincronizar la ejecución de los procesos que conforman parte del servidor, mantener actualizada la tabla de locks (a ni-

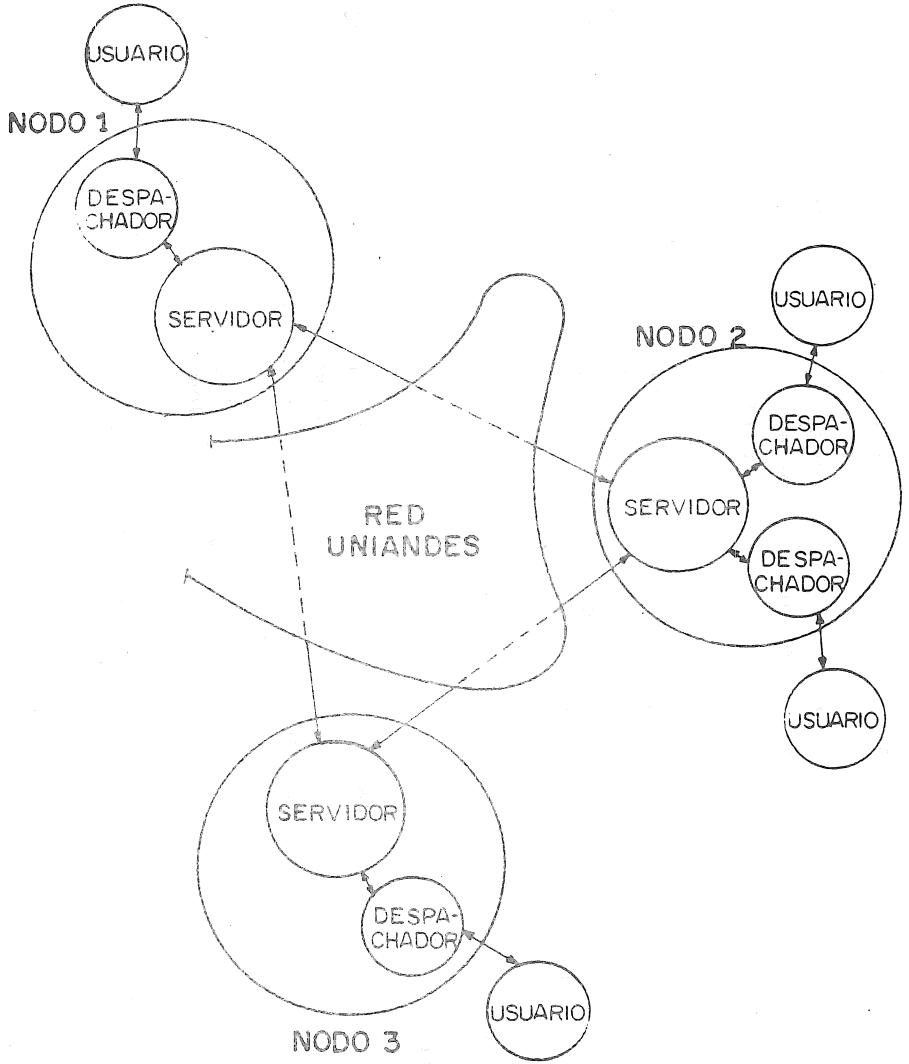
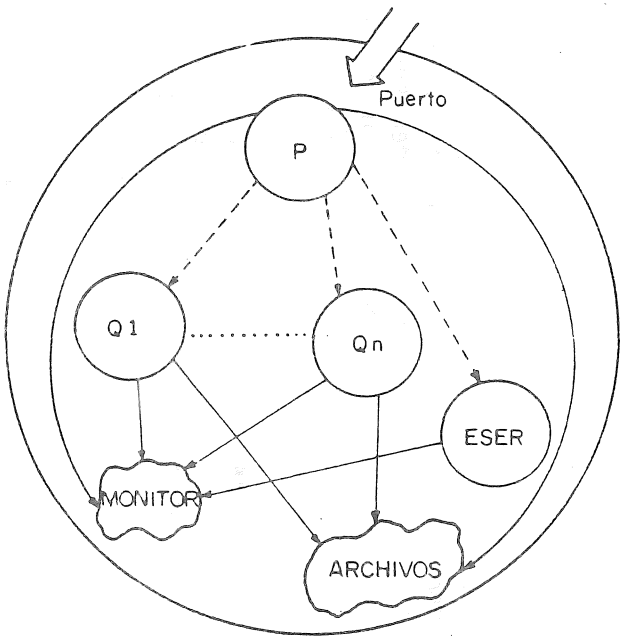


Fig. : 4. Estructura de Operación de SAD



P : Recibo de Pedidos

Qj : Proceso PRIMó Generado por Pseudo - operando

ESER : Estado - Servidores

Fig.: 5. Estructura Interna del Servidor

vel de página y archivo), manipular la cola de mensajes de los procesos en ejecución, asociar time-outs a cada mensaje y generar el proceso de recuperación.

Puertos y Mensajes

Un puerto es un punto de entrada a un servidor. Los puertos son las únicas entidades que tienen nombres globales para que cada proceso maneje directamente sus mensajes; los nombres de los puertos pueden ser enviados en mensajes, esto implica que se pueden recibir mensajes en un mismo puerto desde diferentes fuentes. La asignación dinámica entre buffer's y puertos es realizada por el protocolo de transporte.

Un mensaje consiste de un comando identificador y los argumentos necesarios. En el envio de mensajes para pedir un servicio el comando identificador corresponde al nombre de una operación que ha de ser invocada.

8. CONCLUSIONES

El diseño del sistema de archivos distribuído (SAD) para la Red Uniandes refleja las facilidades fundamentales que se requieren para dar soporte a una aplicación distribuída. La arquitectura de SAD abstrae niveles independientes lo cual reduce la complejidad y permite expandibilidad.

En el diseño de un sistema de archivos distribuído la arquitectura de nombres, directorio y primitivas deben ser bien definidas. La noción de transacción es fundamental para el control de concurrencia. Este diseño puede servir de base para la construcción de un sistema más sofisticado como una base de datos distribuída en la cual unproblema interesante de diseño es el modelo externo.

Para la implantación de un sistema de archivos distribuído se ve la necesidad de una herramienta que permita procesos integrados, monitores, variables de condición y otras facilidades, como los sitemas de lenguajes de Programación: PLITS (FELD79), MESA (MITC78) que son de considerable potencia y elegancia.

REFERENCIAS

(ARAU80) Araujo, L. y Fernándes, J. , "Diseño de la Interfase para la Red Local Uniandes", publicaciones del Departamento de Ingeniería de Sistemas y Computación de la Universidad de los Andes. 1980.

(BRIN73) Brinch Hansen, P., Operating System Principles Prentice Hall, Inc. Englewood Cliffs, New Jersey, 1973.

(DAVI79) Davies, D., et. al., Computer Networks and their Protocols, John Wiley & Sons, 1979.

(DIJK68) Dijkstra, E.W., "Cooperating Sequential Processes", Programing Languages. (F. Genuys, ed.), Academic Press, N.Y., 1968

(ELLI77) Ellis, C.A., "A Robust Algorithm for Updating Duplicated Databases", Proc. Second Berkeley Workshop on Distributed Data Management and Computer Networks,

(FELD79) Feldman, J.A., "High Level Programing for Distributed Computing", CACM, Vol 22, 1979

GAIT80) Gaitán, L. y Paredes. R., "Diseño de Protocolos de Alto Nivel para la Red Local Uniandes", Publicaciones del Departamento de Ingeniería de Sistemas y Computación de la Universidad de los Andes, 1980.

(HOAR74) Hoare, C.A., "Monitors: An Operating System Structuring Concept". CACM. Vol. 17, 1974. pp. 594-557.

(LAMP79) Lampson. B. y Sturgis, H., "Crash Recovery in a Distributed Data Storage System", Xerox PARC, Marzo 1979 (aparecerá en CACM)

(LISK79) Liskov, B. "Primitives for Distributed Computing", Proc. of 7th Symposium on Operating System Principles Diciembre 1979, pp. 33-42.

(METC76) Metcalfe, R. y Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks". CACM Vol. 21, 1976. pp. 395-403.

(MITC78) Mitchel, J.G., et. al., Mesa Language Manual, CSL Repor 79-3, Xerox PARC, Febrero, 1978.

(PAXT79) Paxton, W. , "Client-Based Transactiosn to Maintain Data Integrity" Proc. of 7th Symposium on Operating System Principles, 1979

(SANC80) Sánchez, E., Ruíz E., "Diseño de un sistema de Archivos Distribuído para la Red Local Uniandes", Proyecto de Grado, Departamento de Ingeniería de Sistemas y Computación de la Universidad de los Andes, 1980.

(SWIN79) Swinchart, G., et. al., "WFS: A Simple Centralized File System For a Distributed Environment", unpublished paper Xerox Palo Alto Reserarch Center 1979.

(THOM79) Thomas, R.H., "A resource sharing executive for the ARPANET", National Computer Conference, 1973

(ULLM80) Ullman, J.D., Principles of Database System, Computer Science Press, 1980