Taller de Sistemas Operativos Avanzados
Práctica #5: Modelo Cliente/Servidor
"Aspectos de Diseño"

Objetivo

Implementar el funcionamiento de los aspectos de diseño en el modelo Cliente/Servidor que soporta un Sistema Operativo Distribuido basado en Micro Núcleo.

Descripción

Los aspectos de diseño básicos del modelo Cliente/Servidor proveen los mecanismos para localizar procesos en la red de forma transparente a los programas de usuario, permiten optar por primitivas de ejecución secuencial o concurrente, proveen el almacenamiento de mensajes pendientes a ser entregados a los procesos y proporcionan los medios de confiabilidad necesarios para garantizar la entrega de mensajes a su destino siempre que dicho destino sea alcanzable.

Cada uno de los aspectos de diseño del modelo Cliente/Servidor y su adecuada implementación es fundamental para que un sistema operativo distribuido cumpla con los aspectos de diseño de los sistemas realmente distribuidos:

1. Transparencia: el direccionamiento de los procesos en la red, en cualquiera de sus variantes, debe ser transparente a los usuarios y los programas, de tal forma que estos no tengan idea de en que máquina se encuentra el proceso servidor
2. Flexibilidad: el mismo modelo Cliente/Servidor soportado por un Micro Núcleo permite que la mayor parte de los servicios del sistema sea provista por procesos a nivel usuario. Los aspectos de diseño del modelo Cliente/Servidor pueden ser administrados por *procesos de sistema*, los cuales se ejecutarían en modo usuario, con lo cual se lograría mantener un núcleo mínimo.
3. Confiabilidad: implementando el uso de reconocimientos a los mensajes recibidos, de no haber un reconocimiento a un mensaje y tras haber reintentado con la misma máquina servidor, se puede optar por llamar al mismo servicio, sólo que localizado en otra máquina, de modo que aquella se haga cargo del trabajo
4. Escalabilidad: durante la implementación de los aspectos de diseño para el sistema y los procesos de sistema correspondientes habrán de considerarse las características de los algoritmos descentralizados a modo de evitar el uso de componentes, tablas o algoritmos centralizados, es decir, evitar la dependencia a un solo procesador
5. Desempeño: la ejecución del sistema distribuido habrá de ser igual o mejor que en un procesador, indistintamente de si se trate de primitivas por bloqueo o sin bloqueo, sólo que el desempeño de los procesos servidores se ve mejorado si se usan primitivas sin bloqueo; el almacenamiento de mensajes temporales o en buzones permite evitar en cierta medida la retransmisión de mensajes por la red

Los aspectos de diseño a implementar por el núcleo (ó los procesos de sistema) consideran la existencia de los siguientes tipos de mensajes y sus efectos:

Tipo	Significa	Máquina Emisora	Máquina Receptora
REQ	Solicitud a petición del proceso cliente	Inicia cronómetro a espera por ACK ó REP, almacena REQ ó mantiene bloqueo de send, al terminar el cronómetro envía AYA	Intenta entregar el mensaje al proceso destinatario; ...si no lo logra envía AU ó TA ...si lo logra es optativo el envío de ACK; almacena status de recepción por si se pierde ACK enviado; inicia cronómetro para enviar REP; si al instante no envió ACK al terminar cronómetro envía ACK y reinicia cronómetro
REP	Respuesta a petición del servidor	Inhibe cronómetro para REP, inicia cronómetro para ACK, almacena REP ó mantiene bloqueo de send, al terminar cronómetro envía AYA	Inhibe cronómetro para REP, envía ACK, almacena status de recepción por si se pierde ACK enviado
AU	Dirección desconocida	El proceso destinatario del mensaje (REQ ó REP) recibido no se encuentra en la máquina local	Cliente: inhibe cronómetro para REP ó ACK según sea, busca en tabla de direccionamiento por otro servidor, de no haberlo envía LSA ó consulta al servidor de nombres según el caso Servidor: inhibe cronómetro para ACK, elimina REP ó desbloquea send según el caso
TA	Intenta más tarde	El proceso destinatario no ha recibido el mensaje por alguna razón: no ha invocado a receive; si cuenta con buzón éste se encuentra lleno; de haber almacenamiento temporal éste ha llegado al límite; la máquina nunca recibió el mensaje (REQ ó REP)	Cliente: inhibe cronómetro para ACK ó REP, inicia cronómetro para retransmisión, al terminar reenvía REQ Servidor: inhibe cronómetro para ACK, inicia cronómetro, al terminar reenvía REP
ACK	Reconocimiento	El mensaje previo REQ ó REP ha llegado correctamente	Cliente: inhibe cronómetro para ACK, elimina REQ ó desbloquea al proceso que invocó a send, se inicia cronómetro para REP Servidor: inhibe cronómetro para ACK, elimina REP y la información para localización del cliente

AYA	¿Estás vivo?	Terminó el cronómetro para recibir ACK; inicia cronómetro para IAA; al terminar el cronómetro se asume haber recibido AU Cliente: debido a que tampoco ha recibido REP	Si el proceso destinatario tiene status de haber recibido mensaje, envía IAA, de lo contrario envía TA; si el proceso destinatario no se encuentra envía AU
IAA	Sí estoy vivo	El proceso destinatario sí recibió el mensaje (REQ ó REP)	Inhibe cronómetro para IAA; Cliente: reinicia cronómetro para REP Servidor: mismo efecto que ACK
LSA	Búsqueda de dirección del servidor (look up)	Realiza una transmisión para buscar al servidor cuyo *dest* no se pudo resolver con la tabla de direccionamiento; inicia cronómetro para FSA; si termina el cronómetro y no se encontró el servidor se puede reintentar un número finito de veces; de no encontrarse servidor se anula send	Si el servidor buscado está registrado en la máquina local envía FSA de lo contrario no contesta
FSA	Dirección del servidor encontrada	Incluye en el paquete el identificador del proceso en la máquina local	Almacena en la tabla de direccionamiento la ubicación del servidor encontrado; de ser el primer FSA recibido para un LSA dado se puede realizar el envío del correspondiente REQ a partir de ese momento

Requerimientos Generales

La implementación de los aspectos de diseño deberá incluir al menos uno de los 4 aspectos básicos (direccionamiento, bloqueos, almacenamiento y confiabilidad), asignados al azar, en alguna de sus variantes:

Direccionamiento

A) Direccionamiento por direcciones ralas de procesos
B) Direccionamiento por servidor de nombres

Bloqueos

Primitivas por bloqueo (esta es la implementación básica presentada durante prácticas de laboratorio anteriores), no aplica elegir esta opción, ya que fue implementada en prácticas previas

C) Primitiva sin bloqueo con copia al núcleo

Primitivas almacenadas vs no almacenadas

D) Mantenimiento temporal de mensajes inesperados

E) Almacenamiento de mensajes en buzones

Confiabilidad

F) Solicitud-Reconocimiento-Respuesta-Reconocimiento

G) Solicitud-Respuesta-Reconocimiento

1. La implementación deberá cumplir con todos los requerimientos (generales, visuales, funcionales, complementarios) así como restricciones especificados para la práctica #4.

Requerimientos Visuales y Funcionales

2. Mostrar en la interfaz gráfica (IG) correspondiente al núcleo, sólo cada suceso significativo en que éste se encuentre involucrado según la opción de implementación particular asignada, así como mostrar en suceso de envío ó recepción de todo mensaje núcleo a núcleo

Requerimientos No Funcionales

3. El funcionamiento de los procesos, en cuanto a la gestión de solicitudes y respuestas, sea el mismo que el entregado para la práctica #2, esto es, sólo en el caso en que se mencione al respecto, los procesos no deben ser modificados para efecto de que la práctica funcione

4. En cuanto al código implementado hasta el momento para el Micro Núcleo, a excepción de las líneas de código necesarias para cubrir la identificación y tratamiento de los paquetes núcleo a núcleo necesarios, las modificaciones en los métodos send(dest,messagePtr), receive(addr,messagePtr) y el correspondiente al hilo de control para el núcleo, no deberán de ir más allá de la semántica especificada para la práctica #2; esto implica la sustitución (o agregado si es el caso) de instrucciones específicas por la invocación de métodos que cumplan con tales instrucciones incluyendo el valor agregado de la implementación particular para esta práctica. Ejemplo: para cuando no se conocía el par (Máquina, Proceso) correspondiente al parámetro *dest*, la búsqueda del servidor se simulaba tomando la IP y el ID de proceso de la interfaz gráfica, que para la opción A una ó dos líneas de código reemplazarían tal(es) instrucción(es)

5. Efectuar la construcción y envío de los mensajes núcleo a núcleo correspondientes según el aspecto de diseño asignado; los mensajes LSA y FSA se consideran con las mismas restricciones especificadas para los demás mensajes núcleo a núcleo

6. La implementación particular de la práctica consiste en una de las siguientes opciones:

A) Direccionamiento por direcciones ralas de procesos

a) El núcleo asigne a un proceso un ID de proceso único y al azar
b) El proceso servidor sea encontrado por transmisión (LSA) basándose en lo proporcionado por el parámetro *dest* de la llamada a send(dest,messagePtr) del emisor, donde el núcleo del servidor notifique (FSA) al núcleo cliente el ID de proceso del servidor
c) Sea implementada la estructura de datos a consultar por la primitiva send(dest,messagePtr) como una *tabla de direccionamiento*, la cual contenga al menos tres campos: Llave-dest, dirección-IP e ID-de-proceso
d) En dicha tabla sean guardadas las entradas correspondientes a búsquedas de procesos servidores (inciso b), una vez encontrados, para evitar búsquedas subsecuentes
e) Se permita eliminar entradas de la *tabla de direccionamiento*
f) Sean registradas las ubicaciones de cada uno de los procesos que se recibiera respuesta de la transmisión y se almacenen bajo una misma clave *dest* en la *tabla de direccionamiento*, de modo que cuando se busque un destinatario *dest* en la tabla, sea posible obtener todas las ubicaciones para un mismo *dest*, y si no se puede entregar la solicitud a uno de los destinatarios, se intente con otro de la lista obtenida
g) Que una entrada en la tabla (ingresada previamente según el inciso d) sea eliminada si se recibe un paquete AU para el correspondiente par (máquina, proceso)
h) En caso de no tener más entradas en la tabla para un *dest* determinado (por recibir paquetes AU, según el punto anterior), sea realizada nuevamente la transmisión de acuerdo al inciso b
i) Los procesos clientes no sean buscados por transmisión
j) El proceso cliente recupere el control sólo hasta que el proceso servidor reciba la solicitud
k) Si tras el envío de un LSA a los 5 segundos no llega el correspondiente FSA se envíe de nuevo otro LSA
l) Máximo se realicen 3 intentos de LSA para un dest determinado por cada llamada a send
m) Esta opción se abstenga de tomar de la interfaz gráfica los datos de IP e ID de proceso para el envío de mensajes como se manejó para la práctica 2
n) Sean impresos en la IG todos los sucesos relacionados a la búsqueda de procesos, su registro en la *tabla de direccionamiento*, direcciones IP e IDs de procesos encontrados y fallas en la localización de los procesos

B) Direccionamiento por servidor de nombres

a) Sea implementado un proceso servidor de nombres
b) Dicho servidor proporcione las operaciones para alta, baja y búsqueda de servidores
c) Los otros procesos servidores le envíen un mensaje al servidor de nombres para indicar su nombre de servidor, IP e ID de proceso, y sean dados de alta
d) El núcleo cliente localice un servidor particular comunicándose primero con el servidor de nombres y obteniendo de él el par (máquina, proceso) del servidor que requiere
e) Una vez que un proceso servidor se mande terminar, éste se dé de baja del servidor de nombres

f) Los procesos clientes hagan uso de una nueva primitiva del núcleo, llamada send(nombreServidor,messagePtr), para la cual, el parámetro *nombreServidor* sea de tipo String y represente el nombre del servicio
g) Los procesos servidores utilicen las primitivas send() y receive() de la práctica #2
h) La interfaz gráfica del servidor de nombres provea un área particular para mostrar el contenido de su tabla de servidores
i) Sean impresos en la interfaz gráfica todos los sucesos relacionados a la búsqueda de servidores, direcciones IP e IDs de procesos encontrados y fallas en la localización de procesos

C) Primitiva sin bloqueo con copia al núcleo (implementación para puntos extras)
a) Aquí se simulará para la primitiva send() no bloqueante, la copia de un mensaje del espacio del cliente al espacio del núcleo con un tiempo (adicional) de transferencia al núcleo, de un milisegundo por cada byte y la transmisión de bytes por la red a un byte por centésima de segundo, deberá ser posible observar al proceso cliente (o en su caso al servidor) realizar actividades en paralelo a la emisión del mensaje por la red.
b) Para la primitiva receive() no bloqueante, una vez que se copie el mensaje al espacio del proceso, éste último deberá darse cuenta de ello (no es necesario que inmediatamente) y procesar el mensaje recibido.
c) Las labores de las primitivas send() y receive() no bloqueantes deberán ser las mismas que lo especificado en la práctica #2 con las bloqueantes, a excepción de que el proceso deberá recuperar el control antes de que surta efecto la primitiva (antes de haber enviado el mensaje y antes de haber recibido el mensaje, según el caso)
d) Sean impresos en la interfaz gráfica todos los sucesos relacionados con la copia al núcleo, la transmisión del mensaje por la red y las actividades en paralelo

D) Mantenimiento temporal de mensajes inesperados
a) Si un núcleo recibe un mensaje dirigido a un proceso no iniciado, o que no ha invocado a receive() y carece de buzón de mensajes (o este está lleno), mantendrá el mensaje en espera durante un lapso de 10 segundos esperando a que el proceso destinatario tome el mensaje
b) Si al concluir el lapso de tiempo, el proceso destinatario no ha tomado el mensaje y el proceso no ha sido iniciado, el núcleo envíe al núcleo emisor un paquete AU
c) Si al concluir el lapso de tiempo, el proceso destinatario no ha tomado el mensaje y el proceso si existe en entre los que este sistema administra, el núcleo envíe al emisor un paquete TA
d) Tras la recepción de un paquete TA, el núcleo intente reenviar el mensaje requerido, de manera automática 5 segundos después de recibir esta notificación
e) Capacidad para mantener mensajes dirigidos a distintos procesos y sólo uno por proceso (requiriendo enviar un TA si ya se encuentra almacenando un mensaje)
f) Sean impresos en la interfaz gráfica todos los sucesos relacionados al almacenamiento temporal de mensajes, llegada de mensajes notificaciones entre núcleos e intentos

E) Almacenamiento de mensajes en buzones

a) Un proceso cualquiera pueda solicitar un buzón (como un buffer en espacio del núcleo)
b) El buzón sea implementado como un espacio de memoria contiguo en arreglo circular
c) El proceso servidor solicite un buzón antes de invocar a receive
d) El buzón tenga capacidad para 3 mensajes del tamaño máximo posible ó los que quepan
e) Los mensajes sean atendidos uno a la vez liberando espacio del buzón
f) Los mensajes tomen un tiempo de 5 segundos en ser atendidos por completo una vez que hayan sido entregados al proceso servidor
g) El núcleo del receptor, en el caso de buzón lleno, envíe un mensaje TA al núcleo emisor
g) Tras la recepción de un paquete TA, el núcleo reintente enviar el mensaje requerido, de manera automática 5 segundos después de recibir esta notificación
h) El núcleo tenga capacidad de manejar un buzón por cada proceso que lo solicite
i) Sean impresos en la interfaz gráfica todos los sucesos relacionados al almacenamiento en buzones llegada de mensajes, notificaciones entre núcleos e intentos

F) Confiabilidad Solicitud-Reconocimiento-Respuesta-Reconocimiento

a) El núcleo inicie un cronómetro para esperar el reconocimiento (ACK) a un mensaje enviado (REQ ó REP)
b) El núcleo envíe un paquete de reconocimiento (ACK) a un mensaje recibido solo sí el núcleo copia el mensaje a un espacio dedicado al proceso destinatario
c) Si el núcleo emisor no recibe en 5 segundos un paquete (de preferencia ACK) alusivo a su mensaje (REQ ó REP), envíe un paquete AYA al núcleo remoto
d) Tras la recepción de un paquete ACK liberar el búfer del proceso emisor
e) Si se recibe reconocimiento a la solicitud, iniciar el cronómetro para esperar la respuesta
f) Tras la recepción de un paquete IAA, asumir el comportamiento del punto “d” anterior
g) Tras la recepción de un AYA, si el núcleo remoto determina que el proceso a que se refiere el paquete está vivo y recibió el mensaje (según “b”), envíe al núcleo emisor un paquete IAA; de lo contrario, si el destinatario no se encuentra envíe un paquete AU ó si el destinatario simplemente no recibió el mensaje (REQ ó REP) envíe un TA
h) Un trabajo a realizar por un proceso servidor tome un tiempo aleatorio entre 1 y 20 segundos
i) El núcleo del cliente busque al servidor (paquetes AYA y IAA) si en 10 segundos no recibe respuesta a su solicitud; para verificar este punto además se simule (aleatoriamente, que unas veces funcione y otras no) la llegada del mensaje de respuesta; lo anterior pueda sea simulado igualmente para los reconocimientos
j) El núcleo emisor de un mensaje intente al menos 3 veces localizar al proceso destino (paquetes AYA y IAA) de no recibir el reconocimiento ó respuesta esperado
k) De no encontrar al proceso destinatario, sea notificada la falla en ambos núcleos
l) Sean impresos en la interfaz gráfica todos los sucesos relacionados a los reconocimientos, intentos de búsqueda de procesos, éxitos y fallos

G) Confiabilidad Solicitud-Respuesta-Reconocimiento (para puntos extras):

Cumplir con los requerimientos de la opción F salvo que...

a) El núcleo inicia cronómetro sólo para esperar ACK por un REP enviado
b) El núcleo envía reconocimiento sólo a un REP recibido
c) Se inicie cronómetro para recibir REP justo después de enviar REQ

Requerimientos y Restricciones Complementarios

7. Código fuente libre de líneas de instrucciones comentadas y que ya no son ejecutadas
8. Código fuente libre de comentarios describiendo a detalle el funcionamiento de esta y las prácticas anteriores
9. Comentar los métodos implementados solo en cuanto a sus parámetros y resultados a obtener de la invocación

Criterios de evaluación

? Revisión sólo si en la práctica #2 se logra que el proceso Cliente reciba su respuesta.
? Los establecidos en las "Reglas de Operación y Evaluación" del Taller de Sistemas Operativos Avanzados y los correspondientes "Periodos de Entrega" de la práctica.
? Código fuente indicando en la parte superior de cada archivo: nombre del(la) alumn@, sección y no. de práctica; entregado vía email en formato zip.
? Fecha de asignación de la práctica: 13 de Mayo de 2009
? Fecha límite para entrega del primer avance de la práctica: 25 de Mayo de 2009
? Fecha límite para entrega de la práctica completa: 01 de Junio de 2009
? Observación: Se posterga asignación de práctica por las semanas de restricción sanitaria reanudando clases el 11/Mayo
? Evaluación por cobertura de requerimientos y fecha de entrega.