

# Comparación de Dos Sistemas Distribuidos: Amoeba y Sprite

Hillary Caituiro Monge

Departamento de Ingeniería Eléctrica y Computadoras

hillarycm@hotmail.com

## Resumen

*En este ensayo se comparan Dos Sistemas Operativos Distribuidos: Amoeba y Sprite, para ello se toma como fuente principal el ensayo titulado "A Comparison of Two Distributed Systems: Amoeba and Sprite" [1]. Ambos sistemas tienen los mismos objetivos pero difieren en la filosofía de diseño, influyendo aspectos como la asignación de procesos a procesadores y la estructura del núcleo. Amoeba sigue la estrategia de la pila de procesadores, implementando un micro núcleo que incluye funcionalidad básica. Sprite se apoya en un modelo basado en estaciones de trabajo e implementa un núcleo monolítico al igual que UNIX. La metodología usada para realizar las comparaciones de desempeño del sistema de archivos se consideran casos especiales de cada uno de los sistemas, en uno de ellos es necesario habilitar y deshabilitar el uso de caché, y en otro caso se toman medidas de desempeño al servidor de archivos excluyendo e incluyendo al servidor de directorio.*

## 1. Introducción

Las computadoras se han convertido en una herramienta importante en las diferentes actividades del ser humano. Desde que comenzó la era de la computadora moderna en 1945 hasta la actualidad, estas han evolucionado impresionantemente. Una de las primeras computadoras costaba 10 millones de dólares y ejecutaba una instrucción por segundo, actualmente cuestan 1000 dólares y ejecutan 10 millones de instrucciones por segundo.

Las redes de área local (LAN: Local Area Network) permiten conectar docenas, e incluso cientos de computadoras, a velocidades de 10 a 100 mega bits por segundo. Las redes de área amplia (WAN: Wide Area Network) están conformadas por millones de computadoras interconectadas, distribuidas en toda la tierra, a velocidades de 64 kilo bites por segundo.

Con la tecnología existente es posible conformar sistemas distribuidos compuestos por un gran número de CPU, conectados mediante una red de alta velocidad, con

facilidad y con bajos costos. Los sistemas operativos distribuidos están apenas en una etapa de surgimiento y son muy distintos a los sistemas operativos tradicionales.

Existen diferentes aspectos que se consideran en el diseño de un sistema operativo distribuido. El objetivo principal es lograr que una colección de computadoras independientes se vean como una sola computadora, esto se conoce como **transparencia**, la **transparencia de localización** se refiere a que los usuarios no pueden identificar la ubicación de los recursos de hardware y software, como los CPU, impresoras y archivos, la **transparencia de migración** significa que los recursos no sufren modificaciones en su nombre al ser cambiados de posición, se dice que un sistema tiene **transparencia de réplica** cuando puede generar replicas de los recursos sin que el usuario lo note. La **flexibilidad** es el segundo aspecto más importante, actualmente existen dos tendencias en la estructura de los sistemas operativos, una sostiene que cada computadora debe ejecutar un núcleo monolítico que proporcione la mayor parte de los servicios, la otra sostiene que solo es necesario la ejecución de un micro núcleo que proporcione lo menos posible y los servicios deben correr independientemente de este, la segunda estrategia es la más flexible. La **confiabilidad**, quiere decir que cuando un miembro del sistema falla otro tome su lugar. El **desempeño** se ve fundamentalmente influenciado por la velocidad de las comunicaciones de las redes, estas no son lo suficientemente rápidas, los diseñadores de Sistemas Operativos Distribuidos para optimizar el sistema, con frecuencia minimizan el número de mensajes. La habilidad de que el número de usuarios crezca con un impacto tolerable en el desempeño, determina la **escalabilidad**, los algoritmos distribuidos contribuyen en lograr este objetivo.

En la publicación "A Comparison of Two Distributed Systems: Amoeba and Sprite"[1], los autores comparan dos sistemas operativos distribuidos, Amoeba y Sprite, dos sistemas operativos que comparten los mismos objetivos pero tienen diferentes filosofías de diseño. Tomando en cuenta aspectos de filosofía de diseño como el entorno de las aplicaciones y asignación de procesadores, y las consecuencias sobre el diseño de la

arquitectura del núcleo, los mecanismos de comunicación, el sistema de archivos y el manejo de procesos.

Los autores argumentan tres razones para comparar ambos sistemas. **Primero:** Amoeba y Sprite enfocan de diferentes maneras las aplicaciones de usuario en un sistema distribuido. Sprite está diseñado como una red de estaciones de trabajo, sobre la que se pueden ejecutar aplicaciones UNIX. Distribuye el sistema operativo mediante un sistema de archivos compartido, pero no da soporte especial para aplicaciones distribuidas. Amoeba como parte de su diseño da soporte a aplicaciones tradicionales, distribuidas y paralelas. Proporciona un lenguaje con facilidades para la programación paralela **Segundo:** La manera en que Amoeba y Sprite asignan recursos de procesamiento es diferente. Sprite asocia usuarios con estaciones de trabajo individuales, en cambio los usuarios de Amoeba comparten una pila de procesadores. **Tercero:** Los autores tienen amplia experiencia en ambos sistemas. Conocen el desarrollo histórico así como las debilidades y fortalezas de Amoeba y Sprite.

El resto de esta publicación está organizado como sigue: La sección 2 narra como los autores ven a los dos sistemas operativos desde la perspectiva de la filosofía de diseño y como los autores comparan diferentes aspectos: las aplicaciones frente a los servicios, la asignación de procesadores a procesos, estructura del núcleo y apoyo a la transparencia. La sección 3 explica la metodología que usó el autor para comparar el desempeño de ambos sistemas. La latencia en la comunicación a nivel de núcleo y a nivel de usuario. El desempeño del sistema de archivos y el desempeño de la administración de los procesos. La sección 4 ve las conclusiones a las que llegaron los autores en términos del desempeño de ambos sistemas. Finalmente, la sección 5 plasma las conclusiones arribadas, basadas en nuestro punto de vista pero influenciadas por los autores [1].

## 2. Filosofía de diseño

Los autores [1] toman dos aspectos de filosofía de diseño fundamentales para comparar ambos sistemas: el almacenamiento compartido y el poder de procesamiento compartido.

Ambos proyectos adoptan un modelo de computación diferente. El equipo de diseño de Sprite asume un modelo tradicional de computación, siguiendo la línea de UNIX, con estaciones de trabajo conectadas en red. Sostienen que la naturaleza distribuida del sistema debería mantenerse oculta en el núcleo. En cambio en Amoeba se apuesta por un futuro donde se supone que en los sistemas de redes existan más procesadores que usuarios y el software sería diseñado para aprovechar el paralelismo

masivo, para facilitar esto se han desarrollado herramientas que facilitan el desarrollo de aplicaciones distribuidas.

Al comparar la forma en que los procesadores se asocian a los procesos, de acuerdo a los autores en Amoeba los usuarios no deberían tener un procesador personal, el poder de cómputo debería ser compartido equitativamente mediante una pila de procesadores conteniendo un gran número de procesadores. En cambio, en Sprite cada usuario tiene su propia estación de trabajo donde ejecuta sus procesos, además provee un mecanismo de migración para trasladar los procesos a otras máquinas que no estén siendo usadas.

### 2.1. Como las aplicaciones ven los servicios de Amoeba y Sprite

[1] Los autores del ensayo, al comparar ambos sistemas en términos de como las aplicaciones ven sus servicios, toman en cuenta la compatibilidad con UNIX y las facilidades que brindan ambos sistemas para el desarrollo de nuevas aplicaciones distribuidas.

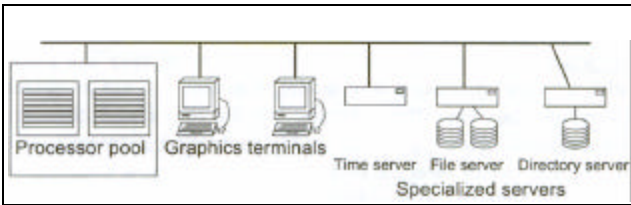
Sprite ofrece compatibilidad con las aplicaciones de UNIX, muchas aplicaciones de UNIX únicamente requieren ser re compiladas para ejecutarse adecuadamente. Para facilitar esta compatibilidad se brinda un sistema operativo de red que corre sobre estaciones de trabajo interconectadas, orientado a un sistema de archivos compartido de acceso consistente y alto desempeño, enfatizando la transparencia de localización en el acceso a archivos. Usa caché para los archivos en las estaciones de trabajo y también en el servidor de archivos, para lo que hace uso de espacio en su memoria principal, reduciendo las transferencias en la red y el uso del ancho de banda del disco, logrando un alto desempeño especialmente con aplicaciones que hacen uso intensivo de archivos. La comunicación entre procesos es pequeña, esta característica es propia de los procesos de UNIX. El método para la transparencia de localización a nivel de usuario es relativamente ineficiente. Aspectos como la migración de procesos y otros se han visto afectados y limitados por la compatibilidad con UNIX.

En Amoeba los procesos, archivos y otras entidades son tratadas como objetos, donde cada objeto está relacionado con una dirección lógica (puerto), que no es una dirección física del servidor que lo maneja, escondiendo así la ubicación del servidor frente a los objetos que interactúan con él. Es parcialmente compatible con UNIX, debido al nuevo y peculiar enfoque basado en objetos, en el diseño de Amoeba, lo que hace más difícil el uso de las aplicaciones existentes, y reduce enormemente el desempeño del emulador que corre para aplicaciones de UNIX. En cambio ofrece un lenguaje de programación,

llamado Orca, que simplifica la escritura de aplicaciones paralelas en un sistema distribuido.

## 2. 2. Como se da la asignación de procesos a procesadores

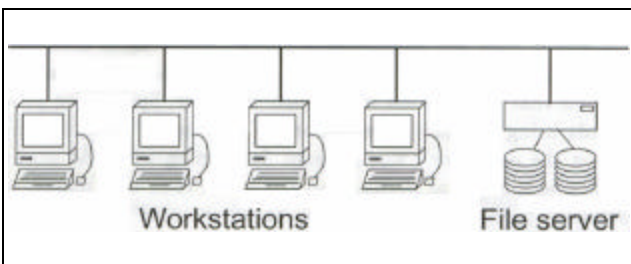
[1] Los autores mencionan que la diferencia entre la asignación de procesadores a procesos, se debe esencialmente a las arquitecturas disimilares que adoptan ambos sistemas. Amoeba sigue la estrategia de la pila de procesadores y Sprite adopta la estrategia del modelo de estaciones de trabajo.



**Figura 1:** “Un sistema Amoeba esta conformado por una pila de procesadores, servidores especializados, y terminales gráficas.” [1]

La arquitectura del sistema en Amoeba contempla una pila de procesadores, servidores especializados y terminales gráficas. Una pila de procesadores tiene una interfaz de red y memoria RAM. Es posible configurar un conjunto de máquinas como una pila de procesadores.

En Amoeba los procesos son dinámicamente asignados a los procesadores de la pila de procesadores, de forma transparente, basado en factores como la carga del procesador y el uso de memoria, independientemente de donde los usuarios los ejecuten, y los usuarios pueden ver el estado de sus procesos desde cualquier lugar en el sistema. “Amoeba asigna procesos al procesador más eficiente, logrando algo de balanceo dinámico de carga” [1]. Además, ejecuta procesos dedicados como el servidor de archivos, para evitar conflictos entre las funciones de los usuarios y del sistema. En las terminales, corren los procesos relacionados con la interfaz gráfica y de red, todos los demás procesos se ejecutan en la pila de procesadores.



**Figura 2:** “Un sistema Sprite esta conformado por estaciones de trabajo y servidores de archivos”. [1]

En Sprite la arquitectura contempla un conjunto de estaciones de trabajo y servidores de archivos. Es posible que una estación de trabajo tenga un conjunto de procesadores, sirviendo así como una pila de procesadores.

En Sprite los procesos se ejecutan en la estación de trabajo donde el usuario inicio la aplicación. Es posible ejecutar los procesos en otras estaciones de trabajo de manera explicita, utilizando una conexión remota a la estación de trabajo donde se desee que corran los procesos. Sprite, además, permite la ejecución automática y transparente de procesos en otras estaciones de trabajo, utilizando el procedimiento conocido como migración de procesos. Mediante este mecanismo, se aprovechan las estaciones de trabajo que no están siendo usadas, un proceso “daemon” centralizado llamado “mig keeps” rastrea los servidores desocupados y les asigna procesos cuando es necesario. Cuando el propietario de una estación de trabajo retorna ha hacer uso de ella, entonces el proceso es migrado otra vez, de nuevo a la estación de trabajo de origen o a otra que este desocupada.

## 2. 3. Comparación de la estructura del núcleo de ambos sistema.

Sprite y Amoeba, difieren enormemente en la arquitectura de la estructura de su núcleo. Amoeba implementa un micro núcleo con un mínimo de servicios. Sprite en cambio tiene un núcleo monolítico, donde incluye toda las funciones y servicios del sistema operativo. En este aspecto los autores [1] comparan ambos sistemas desde la perspectiva de cada estructura de núcleo, y además concluyen a partir de los resultados que no necesariamente el desempeño de los micro núcleos necesitan ser inferiores a los núcleos monolíticos.

Al igual que Sprite UNIX tiene un núcleo monolítico, esa es una de las razones por las que Sprite es compatible con sus aplicaciones. Cada estación de trabajo ejecuta su propia copia del sistema operativo y cuando requiere hacer uso de los servicios de este lo hace de forma local. El único servicio compartido a nivel de núcleo es el sistema de archivos. Un núcleo monolítico al integrar toda la funcionalidad del sistema operativo en una solo espacio de direcciones, tiene un mejor desempeño.

Amoeba implementa un micro núcleo con un conjunto mínimo de servicios que tienen cuatro funciones básicas: manejo de procesos e hilos, soporte del manejo de memoria de bajo nivel, soporte de la comunicación, manejo de E/S de bajo nivel. Todos los demás servicios son implementados como procesos que corren en el nivel de usuario. Por ejemplo un sencillo proceso como el de la

hora del día podría correr en un servidor dedicado. Un micro núcleo es teóricamente ideal, ofrece uniformidad, modularidad, y extensibilidad. Pero aquí surge la interrogante de cuanto afectaría la sobrecarga surgida cuando un proceso se ejecuta en un servidor dedicado, la comunicación entre el micro núcleo y el proceso tendrían que atravesar la red que los separa, esto es mucho mas lento que comunicarse utilizando la memoria. Los autores dan este ejemplo: "... el costo mínimo de una llamada de núcleo en Sprite sobre una estación de trabajo Sun 3/60 es alrededor de 70 microsegundos, mientras que el mínimo costo de una llamada de procedimiento remoto entre dos distintos procesos en un procesador Amoeba es de 500 microsegundos. Además, un servicio puede ser provisto por cada núcleo en Sprite pero por un simple servidor global en Amoeba. El acceder un servicio sobre el protocolo de red de área local 'Ethernet' en Amoeba toma al menos 1200 microsegundos" [1].

Ambos usan llamadas a procedimientos remotos para la comunicación entre núcleos con algunas pequeñas variaciones en la implementación. La Tabla 1 (a) muestra el desempeño de llamadas a procedimientos remotos de núcleo a núcleo en cada sistema.

## 2. 4. Comparación desde el punto de vista del apoyo a la transparencia.

Los sistemas operativos distribuidos tiene como uno de sus principales objetivos la transparencia, existen diferentes tipo de transparencia. Los autores mencionan la transparencia de localización en la comparación de los dos sistemas operativos tratados en el ensayo [1].

Ambos sistemas han enfatizado la transparencia de localización en el acceso de archivos, aunque lo implementan de formas diferentes. Sprite brinda acceso a archivos compartidos, cuando un archivo es requerido por una estación de trabajo, una copia del archivo se mantiene de forma local, utilizando para esto memoria caché, lo que le permite un mejor desempeño. Amoeba corre un servidor dedicado de archivos y directorios, igualmente el acceso a los archivos esta disponible de forma transparente independientemente de su ubicación. El servidor de directorio traduce nombres en capacidades (método de identificación de los objetos en Amoeba). La replicación de las entradas del directorio son creadas automáticamente, y los archivos son replicados asincrónicamente. La replicación de archivos es simple e inmutable, pero la replicación de entradas de directorio es más complicada.

Amoeba esta organizado de tal manera, que los procesos sean automáticamente asignados a los procesadores, sin que los usuarios intervengan, es decir tiene transparencia en la asignación. En Sprite mediante un

mecanismo transparente de migración de procesos, estos son asignados a los procesadores de estaciones de trabajo desocupadas.

## 3. Metodología que se usó para comparar el desempeño de ambos sistemas

Para medir el desempeño de ambos sistemas, los autores [1], toman en cuenta tres aspectos: latencia en la comunicación, desempeño del sistema de archivos, y desempeño de la administración de procesos.

Para compara la latencia en la comunicación, toman medidas a nivel del núcleo y a nivel usuario, transfiriendo 0 bytes, 16 Kbytes, y 30000 bytes, utilizando dos estaciones de trabajo Sun 3/60. La tabla 1 nos muestra los resultados y además explica aspectos particulares.

En cuanto se refiere al desempeño del sistema de archivos de Amoeba y Sprite se utiliza el método de Ousterhout. Se toman medidas de operaciones "apertura-cerrado", lectura, y creación-borrado. Los resultados se pueden apreciar en la tabla 2 y también se describe detalladamente el proceso aplicado y se interpretan los resultados.

Finalmente el desempeño del administrador de procesos es visto desde dos perspectivas: El modelo de procesos y la asignación de procesos. La tabla 3 muestra el desempeño de creación de un nuevo proceso a partir de una imagen ejecutable y se espera que termine. La tabla 4 muestra el costo de creación de un nuevo proceso para ejecutar un pequeño programa que inmediatamente termina.

### 3.1. Latencia de Comunicación

Size (Bytes)	Kernel-level Latency (msec)	
	Amoeba	Sprite
0	1.1	1.9
16384	20.0	19.5
30000	36.0	---

(a)

Size (Bytes)	User-level Latency (msec)	
	Amoeba	Sprite
0	1.2	7.9
16384	21.0	33.5
30000	36.0	62.8

(b)

**Tabla 1:** "Latencia de comunicación en Amoeba y Sprite. Las medidas fueron tomadas para unidades de transferencia de 0 bytes, 16 Kbytes (la transferencia más larga permitida para las llamadas a procedimientos remotos de núcleo a núcleo en Sprite), y 30000 bytes (la transferencia más larga permitida durante una sencilla llamada a procedimientos remotos en Amoeba). La parte (a) muestra el desempeño de las llamadas a procedimientos remotos de núcleo a núcleo. Amoeba provee una apreciable baja latencia para pequeñas llamadas a procedimientos remotos pero Sprite provee mejor desempeño con unidades de transferencia más grandes. La diferencia en el desempeño de transferencias grandes crece debido a que no se confirma la recepción (acknowledged) de los fragmentos

individuales en Sprite. La parte (b) muestra el desempeño de la comunicación entre procesos en el nivel de usuario. Las llamadas a procedimiento remoto en Amoeba son considerablemente más rápidas que las operaciones 'pseudo-device' operaciones para todos los tamaños de datos. Las medidas fueron tomadas en dos estaciones de trabajo Sun 3/60 conectados por un protocolo de red de área local 'Ethernet' de 10-Mbit" [1].

### 3.2. Sistema de Archivos

Los autores [1], comparan el desempeño del sistema de archivos de ambos sistemas, usando el estándar de comparación (benchmark) del análisis de desempeño de sistema operativo de Ousterhout. Es posible apreciar los resultados en la tabla 2.

Operation		Delay (msec)			
		Amoeba		Sprite	
open-close	foo	7.2		9.7	
	a/b/c/foo	7.6		10.4	
read				CACHE	NOCACHE
	10 Kbytes	14.0		2.8	18.6
	100 Kbytes	123.0		21.7	167.4
create-delete		BULLET	BULLET/DIR	CACHE	NOCACHE
	no data	33.0	288.0	50.9	50.9
	10 Kbytes	86.0	312.0	67.1	84.9
	100 Kbytes	367.0	617.0	101.4	411.1

**Tabla 2[1]:** Resultados de la comparación del sistema de archivos de Amoeba y Sprite. Estas medidas fueron hechas en una estación de trabajo Sun 3/60 conectado por un protocolo da red de área local de 10-Mbit "Erthernet".

En Sprite se mide el tiempo que toma el abrir y cerrar un archivo. En Amoeba, el tiempo que toma la búsqueda y ubicación de una entrada en el servidor de directorio. El nombre del archivo conteniendo un elemento, "foo" (nombre genérico usado para elementos indefinidos (por ejemplo: archivos de computadora, programas, etc)), y el tiempo para un nombre conteniendo cuatro elementos, a/b/c/foo/.

Se mide también, la lectura de 10 Kbytes y 100 Kbytes. En Sprite, se considera el caché de cliente activado y desactivado.

Se simula el uso de un archivo temporal sin datos, de 10 Kbytes, y 100 Kbytes, para medir el tiempo de creación de un archivo, escribir una cantidad fija de datos en este, y cerrarlo, entonces abrir el archivo, leer datos desde este, cerrarlo y finalmente borrarlo. En Amoeba se considera con y sin el registro en el directorio. En Sprite se activa y desactiva la caché de cliente.

### 3.3. Administración de Procesos

#### 3.3.1. Modelo de Procesos

Operation	Time (msec)	
	Amoeba	Sprite
Context switch	0.5	1.6
Thread creation	2.4	(12.5)
fork	(169.5)	13.6
Program invocation	58.0	71.6

**Tabla 3:** "Desempeño de la conmutación de contexto y creación de procesos en una estación de trabajo Sun 3/60. Los números entre paréntesis indican operaciones que no son ejecutadas bajo condiciones normales: las bifurcaciones de memoria compartida en Sprite y UNIX como las bifurcaciones en Amoeba. El estándar de medida 'conmutar contexto' mide el costo de comunicación de ida y vuelta (por ejemplo dos conmutaciones de contexto). Amoeba funciona mejor que Sprite en todas las áreas pero con una bifurcación como UNIX. El alto costo de creación de un proceso en Amoeba desde uno existente es atribuible a la sobrecarga relacionado con la compatibilidad con UNIX; normalmente, este costo es evitado debido a que los procesos en Amoeba invocan programas sin la intervención de una bifurcación" [1].

#### 3.3.2. Asignación de Procesos

Operation	Time (msec)	
	Amoeba	Sprite
Local	58	72
Remote (specified)	84	116
Remote (unspecified)	95	131

**Tabla 4:** "Desempeño de la invocación de programas. La invocación local es más rápido en Amoeba que en Sprite, como es una invocación remota si un nuevo procesador debe ser seleccionado. Sprite normalmente ejecuta localmente o rehúsa el mismo servidor múltiples veces para invocaciones remotas, con un costo mínimo de 72 y 116 milisegundos respectivamente. Amoeba normalmente selecciona un procesador cada vez que un programa es invocado, por un costo mínimo de 95 milisegundos. Las medidas fueron tomadas en estaciones de trabajo Sun 3/60 conectadas por un protocolo de red de área local 'Ethernet' a 10-Mbit" [1].

## 4. Conclusión de los autores sobre la comparación de ambos sistemas.

Los autores han tomado medidas en diferentes aspectos, en términos del desempeño de ambos sistemas, que les permitieron llegar a la conclusión de que la estrategia de un sistema híbrido, conteniendo estaciones de trabajo y pilas de procesadores es una opción que permitiría tomar las ventajas que ambos ofrecen. Sprite ha demostrado los beneficios del uso de caché en las estaciones de trabajo. La suposición de que el desempeño de los micro núcleos es inferior a la de los núcleos

monolíticos, es desmentida por las pruebas de comparación.

## 5. Conclusiones

En el ensayo [1] se comparan dos sistemas operativos distribuidos, que han cerrado sus arquitectura en dos extremos, la estrategia de Amoeba es la pila de procesadores y Sprite sigue el modelo de estaciones de trabajo. Eso ha influenciado significativamente la estructura de los núcleos y la asignación de procesos. Amoeba cuenta con un micro núcleo y Sprite con un núcleo monolítico. Los procesos en Amoeba son automáticamente distribuidos entre los procesadores de la pila de procesadores. En Sprite cuando se ejecuta un proceso, este se ejecuta en la estación de trabajo donde se inicio, el usuario tiene prioridad sobre sus recursos.

A pesar de que Sprite y Amoeba han seguido estrategias completamente opuestas, es posible acercarlos, debido a sus posibilidades de configuración. Sprite puede correr una estación de trabajo con una pila de procesadores con desempeño semejante a Amoeba. La pila de procesadores de Amoeba no tiene que ser necesariamente una unidad, un conjunto de computadoras pueden formar parte de la pila de procesadores.

En términos de transparencia, Amoeba ofrece más que Sprite. Amoeba es transparente en la localización mediante su sistema de archivos, la asignación de procesos a procesadores y la replicación lograda a través de su servidor de directorio. Sprite ofrece la transparencia de localización con su sistema compartido de archivos, igualmente es transparente en su mecanismo de migración de procesos.

Un sistema operativo híbrido, sería una opción más realista y tomaría las ventajas de ambas estrategias. Aunque seguramente esto conllevaría una pérdida en desempeño.

El equipo de diseño de Sprite, eligió la estrategia que le permitió mayor compatibilidad con aplicaciones de UNIX, logrando que se ejecuten sobre su entorno un gran número de aplicaciones, pero esto deterioró y limitó su funcionalidad.

En Amoeba se apostó por crear un nuevo enfoque, que les ha permitido, construir un sistema operativo más transparente y con una personalidad distribuida.

## 7. Referencias

[1] Fred Douglass, M. Frans Kaashoek, John K. Ousterhout, and Andrew S. Tanenbaum, "A Comparison of Two Distributed Systems: Amoeba and Sprite", *Journal*, Publisher, Location, Date, pp. 1-10.