# *Sistemas Operativos*

## Multiprocesadores - II

# *Sistemas Operativos*

- ☑ Versión: Junio 2017
- ☑ Palabras Claves: Multiprocesadores, SMP, Redes, Distribuidos, UMA, NUMA

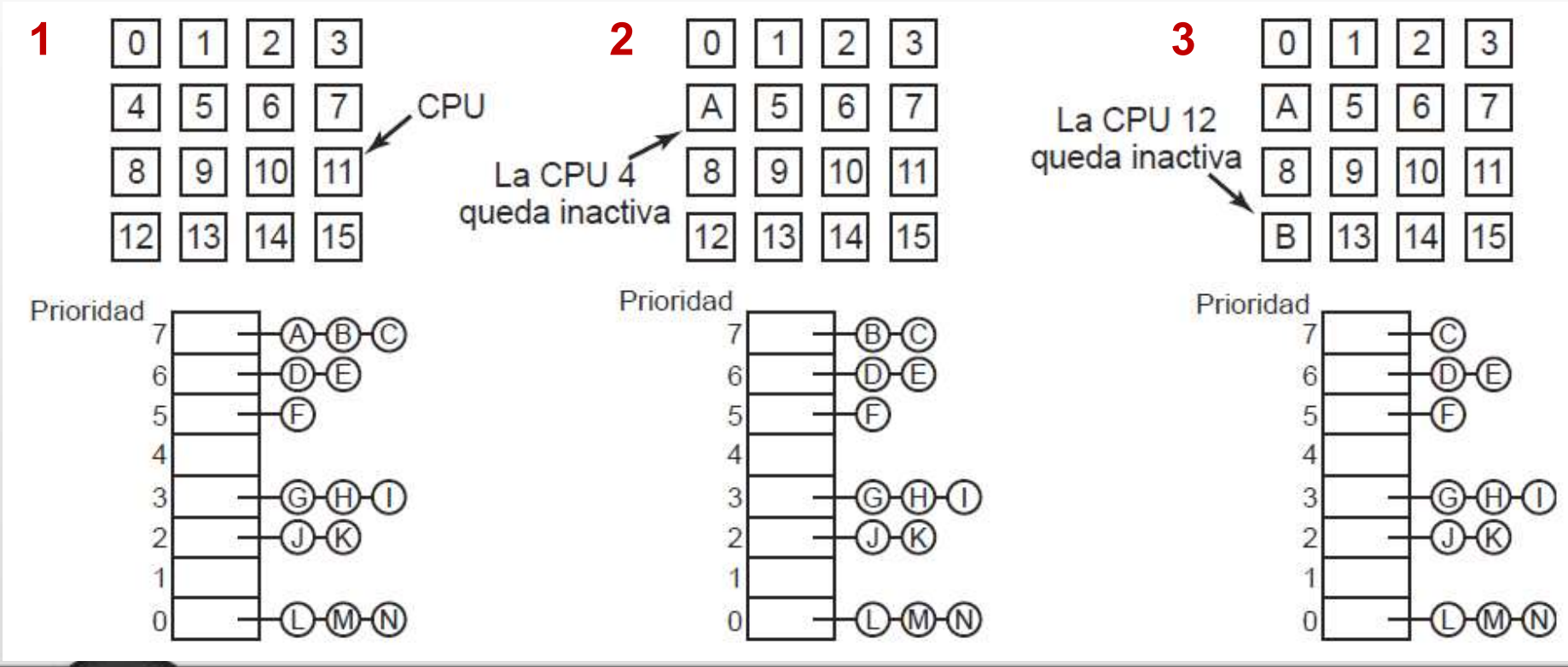
Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) , el de Silberschatz (Operating Systems Concepts)

# *Planificación de Multiprocesadores*

- ✓ Antes de analizar como se va a hacer la planificación, es necesario determinar que se va a planificar:
  - ✓ Uniprocesador : ¿Qué hilo (o proceso) se seleccionara?
  - ✓ Multiprocesador: Se suma ¿En que CPU se va a ejecutar?
  - ✓ Complejidad con hilos que trabajan "en conjunto":
    - ✓ Si son ULT, el planificador los planifica a nivel de proceso
    - ✓ Si son KLT, es posible tomar decisiones sobre su planificación ← **Este tipo de hilos vamos a analizar**

- **Planificación de hilos independientes:**
  - Esta situación se da generalmente en ambientes de tiempo compartido
  - Muchos usuarios ejecutando tareas que generalmente no tienen relación entre sí.
  - El esquema más sencillo para planificarlos es tener una única cola de listos para todos los hilos
  - Podríamos tener varias… Una para cada prioridad

- Planificación de hilos independientes:
  - Las 16 CPU están ocupadas y hay 14 hilos en la cola de listos ordenados por prioridad



- **Planificación de hilos independientes:**
  - Ventajas:
    - Mientras que los hilos no estén relacionados, esta planificación resulta razonable
    - Es simple de implementar
    - Es eficiente
    - Provee un balanceo de carga, ya que los trabajos se van repartiendo en todas las CPU
  - Desventaja:
    - El acceso a una estructura única, podría suponer un cuello de botella generando contención

- **Planificación de hilos independientes:**
  - Problema con la espera activa
    - Supongamos que un hilo tiene un bloqueo de espera activa y finaliza su Quantum antes de liberar el mutex.
    - Las otras CPUs que esperan a que se libere el bloqueo, están ociosas hasta que se libere el lock
  - Solución:
    - Utilizar un flag en cada proceso que indica que alguno de sus hilos tiene una espera activa (Zahorjan y colaboradores, 1991)
    - No expulsar procesos que tengan el flag activado
    - Cuando se libera el lock, se limpia el flag

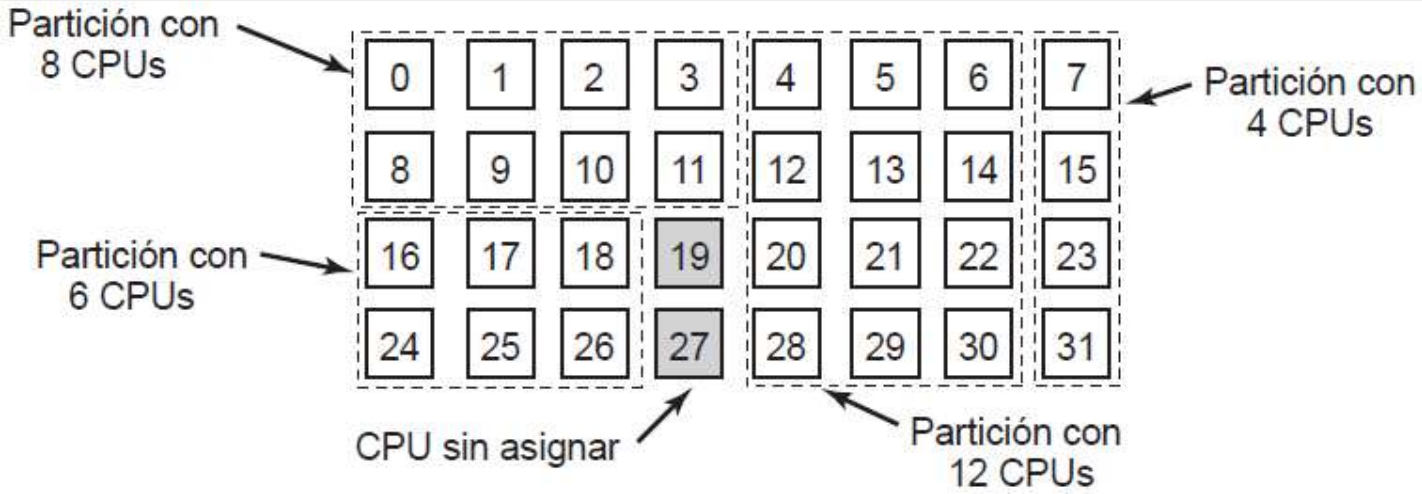
# *Multiprocesadores con M. Compartida (cont.)*

## ☑ Planificación de hilos independientes:

- ✓ Aprovechamiento de la Cache
  - Un hilo que se ejecuta en un CPU donde ya se ha ejecutado, tendrá mayor posibilidad de que sus datos aún sigan en la cache de dicha CPU
  - Se utiliza el algoritmo de Planificación de 2 niveles:
    - Cuando se crea un hilo, se asigna a una CPU
    - Cada CPU tiene su propia colección de hilos y los planifica por separado
    - Si queda una CPU ociosa, se reparten los hilos
    - Minimiza la contención de las estructuras de datos asociadas, ya que no hay solo una.

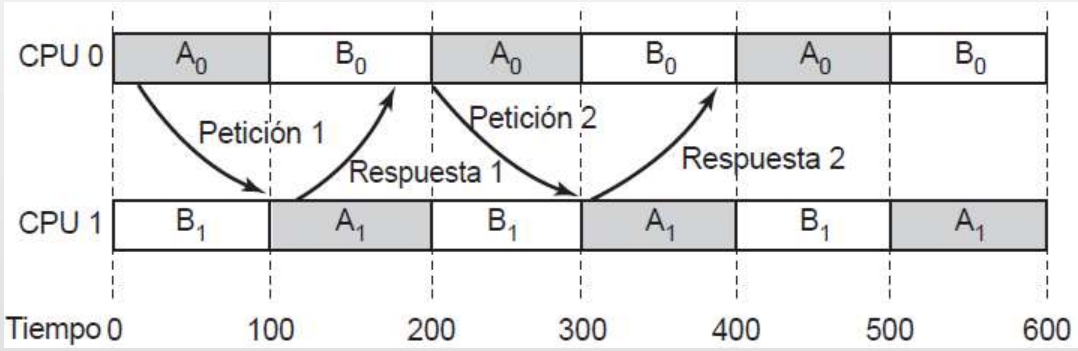
# *Multiprocesadores con M. Compartida (cont.)*

- **Planificación de hilos que trabajan en conjunto:**
  - Se pueden planificar en conjunto (en varias CPUs)
  - Mejora en trabajos en paralelo
  - El grupo se planifica si hay CPUs libres para cada hilo del grupo. Si no las hay, el grupo espera
  - No hay multiprogramación por CPU (cada CPU ejecuta solo 1 hilo) , baja la productividad



## ☑ Planificación de hilos que trabajan en conjunto:

- Se podrían multiprogramar las CPU, pero podría ocurrir que los hilos no se ejecuten sincrónicamente, ya que se planifican independientemente.
- Supongamos una situación de 2 hilos $A_0$ y $A_1$ que se ejecutan intercambiando mensajes compartiendo CPU con los hilos de un proceso B:




- Como $A_0$ se ejecuta en un intervalo distinto que $A_1$ ocurre que el lapso de ejecución de A es de 200 mseg, cuando podría haber sido de 100 si se hubieran ejecutado en el mismo intervalo.

# *Multiprocesadores con M. Compartida (cont.)*

- **Planificación de hilos que trabajan en conjunto:**
  - **Planificación por pandillas (una solución al problema anterior):**
    - Hilos relacionados se toman como una Pandilla
    - Todos los miembros de una pandilla se ejecutan simultáneamente en distintas CPUs multiprogramadas
    - Todos los miembros de la pandilla inician y terminan sus intervalos en conjunto.
    - Ejemplo:
      - Supongamos un multiprocesador con 6 CPU utilizadas por 5 procesos (A..E) y un total de 24 hilos

# *Multiprocesadores con M. Compartida (cont.)*

- Ejemplo (cont.)
  - En el instante de tiempo 0, se programan los hilos $A_0$ .. $A_5$. Luego se ejecuta $B_0$, $B_1$, $B_2$, $C_0$, $C_1$, $C_2$, etc…
  - Se continúa la ejecución cíclicamente

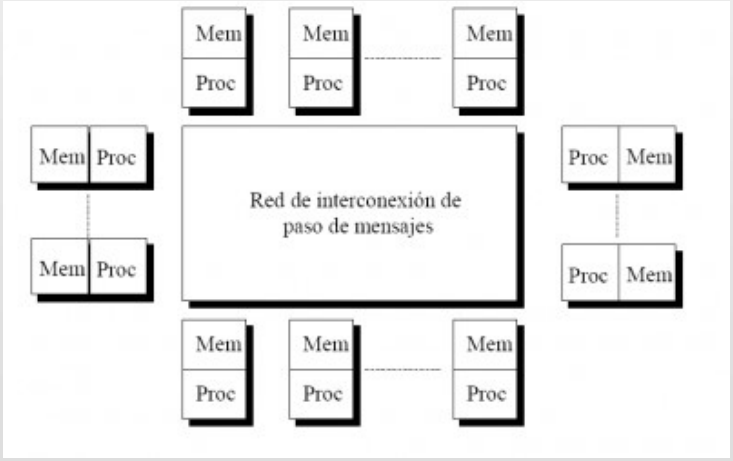
| Time slot | CPU 0 | CPU 1 | CPU 2 | CPU 3 | CPU 4 | CPU 5 |
|---|---|---|---|---|---|---|
| 0 | $A_0$ | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ |
| 1 | $B_0$ | $B_1$ | $B_2$ | $C_0$ | $C_1$ | $C_2$ |
| 2 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $E_0$ |
| 3 | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ |
| 4 | $A_0$ | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ |
| 5 | $B_0$ | $B_1$ | $B_2$ | $C_0$ | $C_1$ | $C_2$ |
| 6 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $E_0$ |
| 7 | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ |

  - La idea de ejecutar todos los hilos de un proceso juntos, permite que el pasaje de mensajes se realice de manera más rápida y eficiente

# Multicomputadoras

# *Multicomputadoras*

- ☑ También conocidos como cluster de computadoras
- ☑ CPUs con acoplamiento fuerte
- ☑ No se comparte memoria, cada CPU tiene la suya
- ☑ También conocidas como Clusters
- ☑ Multicomputadora:
  - ☑ PCs con una interfaz de red
  - ☑ Generalmente carecen de Placa de Video, Sonido y en algunos casos disco



# *Multicomputadoras*

- ☑ También conocidos como cluster de computadoras
- ☑ CPUs con acoplamiento fuerte
- ☑ No se comparte memoria, cada CPU tiene la suya
- ☑ También conocidas como Clusters
- ☑ Multicomputadora:
  - ☑ PCs con una interfaz de red
  - ☑ Generalmente carecen de Placa de Video, Sonido y en algunos casos disco

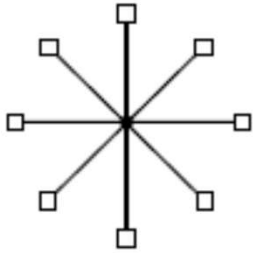
# *Multicomputadoras*

- Multicomputadora:
  - Poseen mucha CPU, memoria y placas de interconexión redundantes:
    - Red: Ethernet (10, 40, 100 Gbps), Infiniband (120 Gbps)
    - HBA: Host Bus Adapter para conexión Storage. Generalmente Fibra óptica y usa el protocolo FibreChannel

  - ✓ Es necesario diseñar correctamente la red

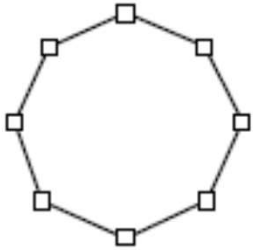
# *Multicomputadoras (cont.)*

Estrella



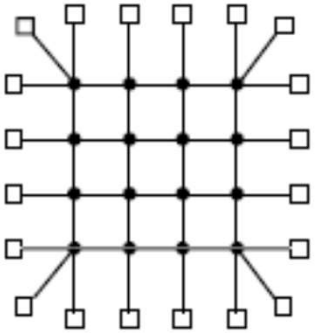
(a)

Anillo



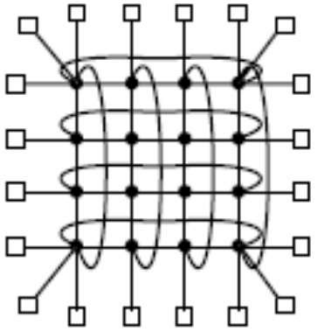
(b)

Malla o Mesh



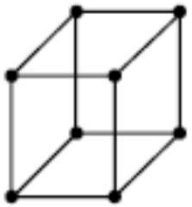
(c)

Doble Malla



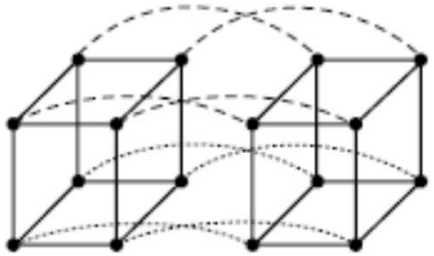
(d)

Cubo



(e)

Hipercubo

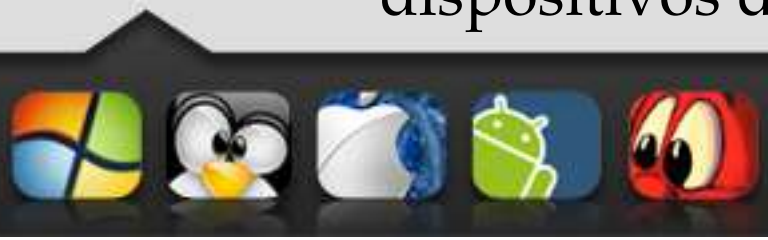


(f)

☑En la actualidad se suelen usar a, b, c y Full Mesh (todos contra todos)

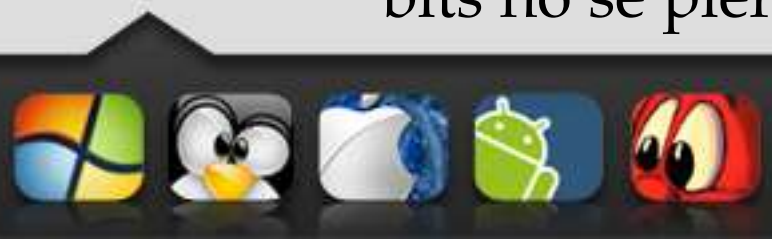
# *Multicomputadoras (cont.)*

- ☑ Transmisión de Datos:
  - ✓ Para la conmutación de datos entre nodos se utilizan 2 esquemas:
    - ✓ **Almacenamiento de conmutación de paquetes y retransmisión:**
      - ✓ Se almacenan los bits en un **buffer** hasta que se logra conformar un paquete, luego se transmite.
      - ✓ El paquete es conmutado entre switches.
      - ✓ Existen mecanismos de retransmisión para solucionar problemas de pérdidas
      - ✓ Inyecta latencia a medida que se atraviesan dispositivos de red



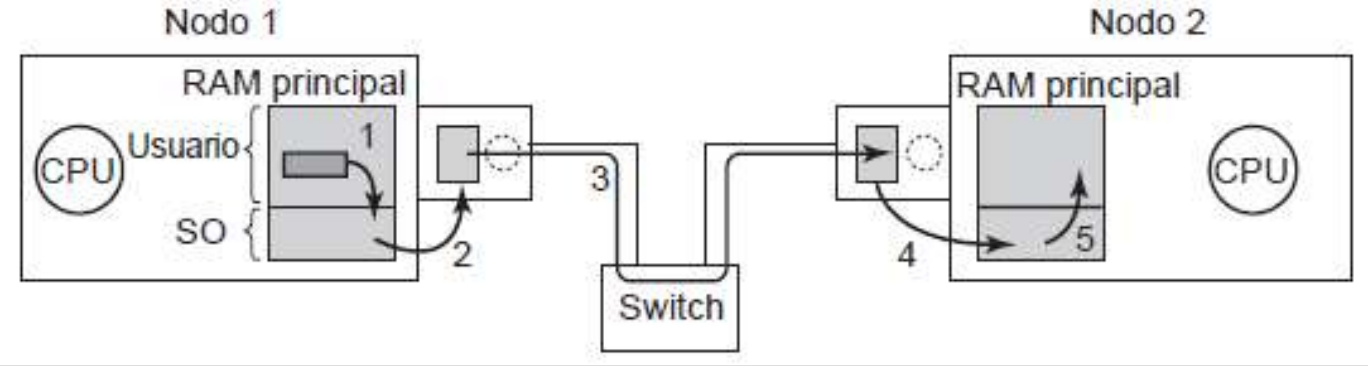
# *Multicomputadoras (cont.)*

- ☑ Transmisión de Datos:
  - ✓ Para la conmutación de datos entre nodos se utilizan 2 esquemas:
    - ✓ **Conmutación de Circuitos:**
      - ✓ El primer switch establece una ruta (circuito virtual) desde el origen hasta el destino.
      - ✓ Una vez establecida la ruta, los bits se inyectan **sin necesidad de almacenarlos en buffers**
      - ✓ Es mas rápida, pero requiere de configuración adicional
      - ✓ No hay control de flujo porque se asume que los bits no se pierden



# *Multicomputadoras (cont.)*

- Transmisión de Datos:
  - **Placas de Red:**
    - Proveen un mecanismo de conexión hacia la red utilizada
    - Poseen memoria (buffers) que permite almacenar los paquetes antes de copiarlos a la RAM.



# *Multicomputadoras (cont.)*

- **Problema en Multicomputadoras**
  - **Copiado de paquetes en exceso**
    - Los datos se tienen que copiar de la RAM principal a la RAM de la placa de red en el origen y viceversa en el destino
    - Adicionalmente si la placa de red está asignada al espacio de direcciones virtuales del kernel, para que un proceso de usuario pueda copiar datos, debe hacerlo a través de una System Call
    - Que los procesos de usuario generen llamadas al sistema continuamente para transmitir datos puede implicar un problema grande de performance (en el origen y el receptor!)

# *Multicomputadoras (cont.)*

- **Problema en Multicomputadoras**
  - **Solución al Copiado de paquetes en exceso**
    - Se puede minimizar asignando interfaz de red al espacio de usuario
    - De este modo el proceso de usuario copia los datos a la interfaz de red sin que el Kernel se involucre
    - Es mas performante
    - El problema es si varios procesos quieren utilizar la interfaz al mismo tiempo. ¿Quién copia/lee datos?
    - Este esquema es solo válido si hay un único proceso o si los procesos son colaborativos (sabemos que en entornos mixtos esto ultimo no ocurre)
    - Por otro lado el Kernel debería competir de eigual a igual con procesos de usuario??
    - Muchos sistemas utilizan 2 interfaces, asignando 1 al kernel  otra a los procesos de usuario (muy costoso)

# *Multicomputadoras (cont.)*

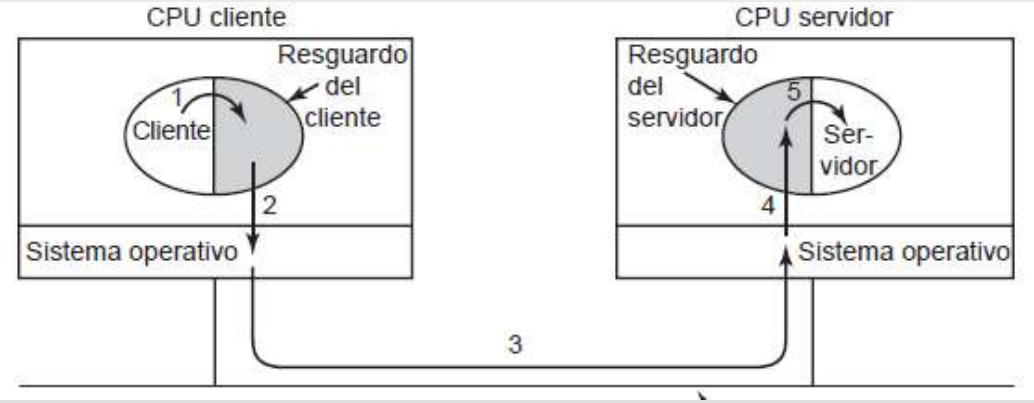
- **Software de Comunicación a nivel de usuario:**
  - Para comunicarse, los procesos en distintas CPUs en una multicomputadora se envían mensajes entre **sí.**
  - **Paso de mensajes:**

```
send(dest, &mptr);
receive(direc, &mptr);
```

    - Send y Receive con y sin bloqueo:
      - El SO Provee interfaces a los procesos de usuario para realizar la comunicación.
      - Envío con bloqueo (síncronas) (la CPU queda inactiva durante la transmisión del mensaje).
      - Envío sin bloqueo con copia (asincrónicas) (se desperdicia el tiempo de la CPU por la copia adicional).
      - Envío sin bloqueo con interrupción (dificulta la programación).

# *Multicomputadoras (cont.)*

- **Software de Comunicación a nivel de usuario:**
  - Otra alternativa: RPC (Remote Procesure Call) para obtener un mayor nivel de abstracción
    - Permite invocar a procedimientos que se ejecutan en otra CPU.
    - El proceso en una máquina invoca al procedimiento remoto y se bloquea hasta que llegue la respuesta
    - La comunicación es transparente al programador



# *Multicomputadoras (cont.)*

- ☑ Planificación
  - ✓ Cada nodo tiene su propio conjunto de procesos
  - ✓ Un nodo no toma procesos de otros para ejecutarlos, ya que seria bastante costoso
  - ✓ Es importante la asignación de procesos a los nodos
    - ◆ Balanceo de la carga
  - ✓ Se puede aplicar el concepto de planificación por pandillas
    - ◆ Sincronización entre los nodos en cada inicio de una ranura de tiempo

# *Multicomputadoras (cont.)*

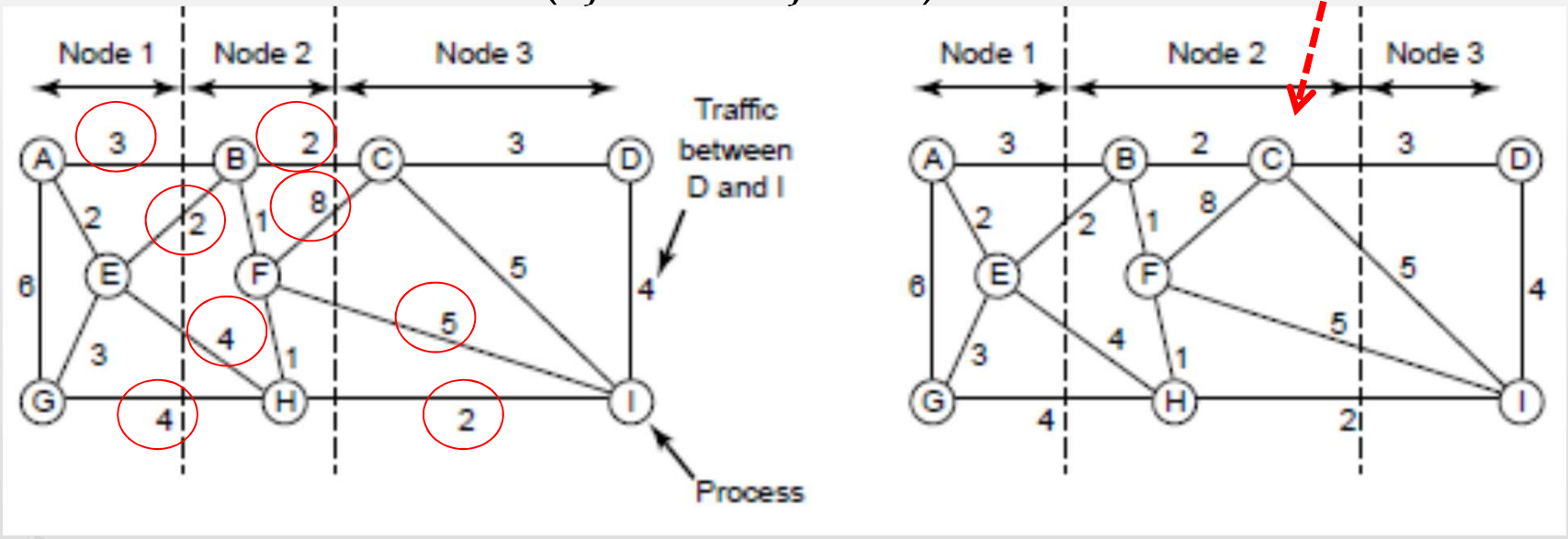
- Planificación – Balanceo de la carga
  - Se basa en lo que "suponen que saben" de cada proceso
    - Tiempo de CPU, memoria requerida, comunicación con otros procesos
  - Objetivos
    - Minimizar ciclos de CPU desperdiciados debido a que el nodo no tiene trabajo
    - Minimizar el uso del ancho de banda en la comunicación
    - Equidad para los procesos

# *Multicomputadoras (cont.)*

- Planificación – Balanceo de la carga (cont.)
  - Se representa al sistema como un grafo
  - Cada nodo es un proceso, y la comunicación entre ellos se representa a través de una arista
  - Se debe particionar el grafo en tantos subgrafos como nodos se tengan, teniendo en cuenta:
    - Requerimientos totales de CPU
    - Requerimientos totales de memoria
    - Minimizar la cantidad de aristas entre nodos de distintos subgrafos (tráfico de red)
    - Buscar clusters con acoplamiento fuerte (veamos el ejemplo)

# *Multicomputadoras (cont.)*

- Planificación – Balanceo de la carga (cont.)
  - Tenemos 9 procesos, sabiendo el costo de comunicación entre estos y tenemos 3 nodos
  - Trafico de la red, suma de los pesos entre enlaces de 2 nodos diferentes (ej1: 30 , ej2: 28)

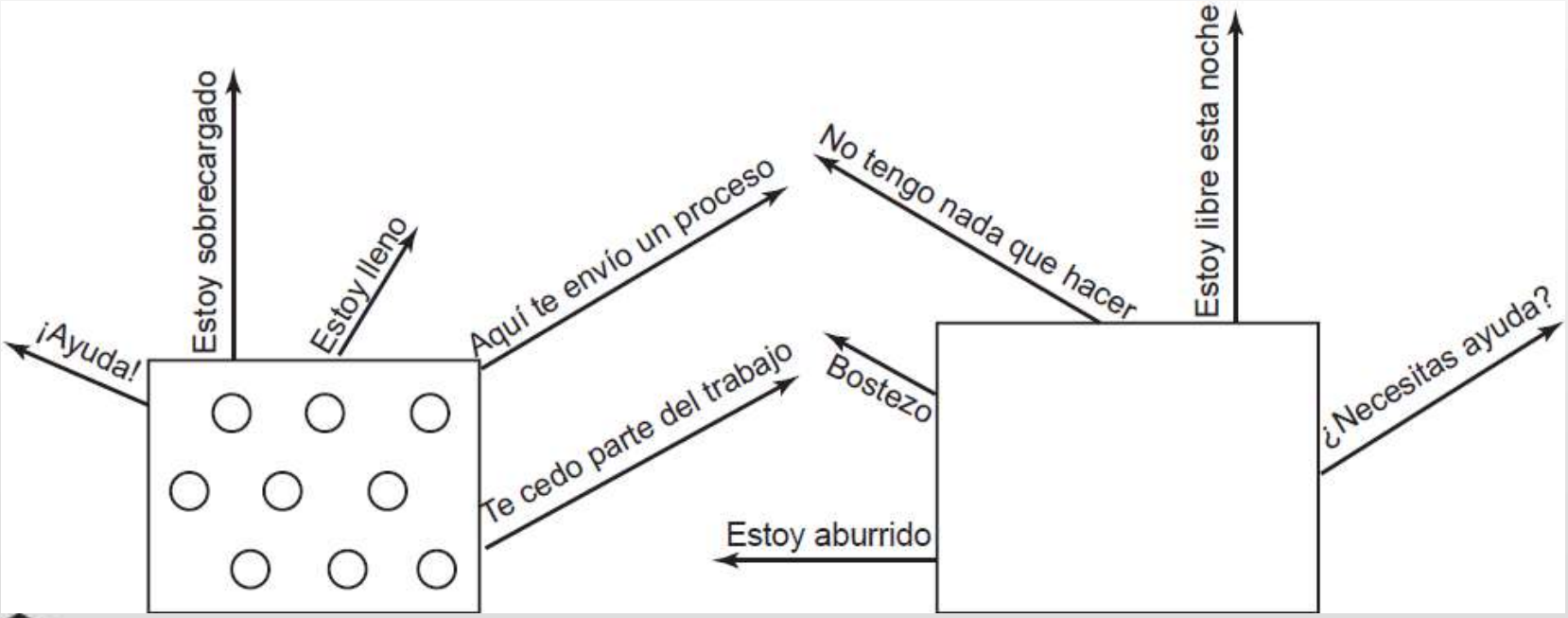


# *Multicomputadoras (cont.)*

- Planificación – Balanceo de la carga (cont.)
  - Otra forma
    - Usar Algoritmos distribuidos
    - El proceso se ejecuta en el nodo que lo creo al menos que el mismo este sobrecargado
      - Muchos procesos, trashing, etc.
    - El nodo sale a buscar un nodo no sobrecargado para "pasarle" el proceso
    - Problemas con sobrecarga del enlace si todos los nodos se encuentran sobrecargados. Mucho pasaje de mensajes!

# *Multicomputadoras (cont.)*

- ☑ Planificación – Balanceo de la carga (cont.)
  - ✓ Otra forma (cont.)
    - ◆ Es posible también que un nodo con poca carga informe la situación



# Sistemas Distribuidos

# *Definición*

- A distributed system is a collection of independent computers that appears to its users as a single coherent system (Tanembaum)
- A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages (Colouris)

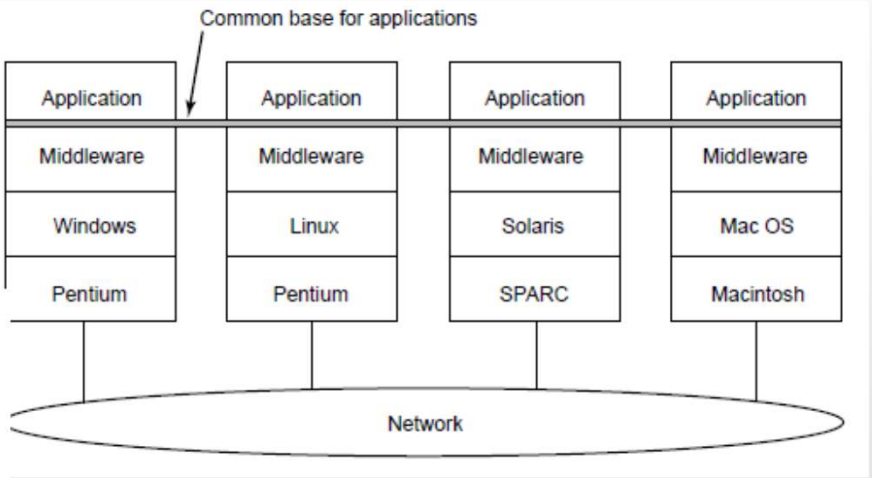
# *Sistemas Distribuidos*

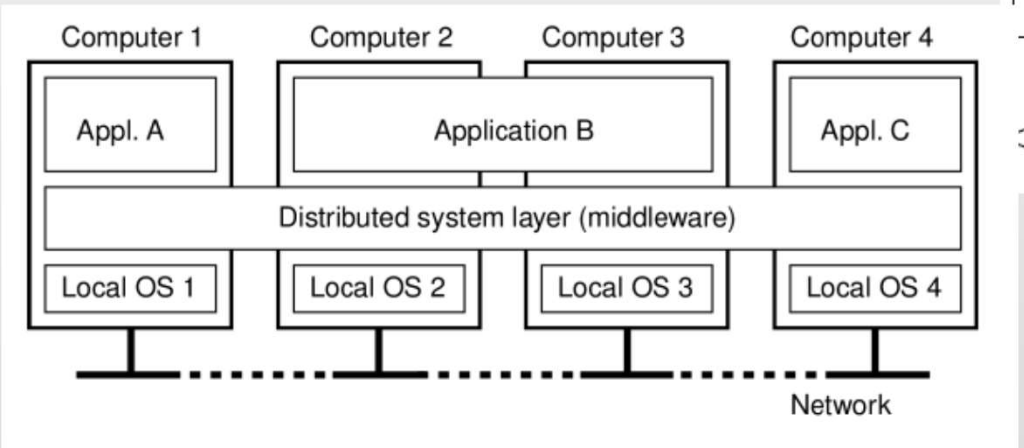
- ☑ Similar a las multicomputadoras
  - ✓ Cada nodo tiene su propia memoria privada
- ☑ Menor acoplamiento
  - ✓ Se pueden encontrar esparcidos por todo el mundo
- ☑ Cada nodo es una PC completa
  - ✓ Incluyendo dispositivos
- ☑ Cada nodo puede ejecutar un SO y Hardware diferente
  - ✓ Incluyendo su propio sistemas de archivos

# Sistemas Distribuidos (cont.)

## Middleware

- Capa de software por encima del SO que permite una uniformidad entre los distintos SOs.



# *Sistemas Distribuidos (cont.)*

- El **middleware** es una capa de Software fundamental en un Sistema distribuido:
  - Provee una interfaz común a todos los procesos
  - Soluciona los problemas de heterogeneidad,
  - Provee servicios a las capas superiores
  - Provee estructuras de datos y operaciones que permiten a los procesos y usuarios inter-operar, de manera consistente, entre máquinas remotas

# *Sistemas Distribuidos (cont.)*

## ☑Resumiendo

| Elemento | Multiprocesador | Multicomputadora | Sistema distribuido |
|---|---|---|---|
| Configuración de nodo | CPU | CPU, RAM, interfaz de red | Computadora completa |
| Periféricos de nodo | Todos compartidos | Compartidos, excepto tal vez el disco | Conjunto completo por nodo |
| Ubicación | Mismo bastidor | Mismo cuarto | Posiblemente a nivel mundial |
| Comunicación entre nodos | RAM compartida | Interconexión dedicada | Red tradicional |
| Sistemas operativos | Uno, compartido | Varios, igual | Posiblemente todos distintos |
| Sistemas de archivos | Uno, compartido | Uno, compartido | Cada nodo tiene el suyo |
| Administración | Una organización | Una organización | Muchas organizaciones |