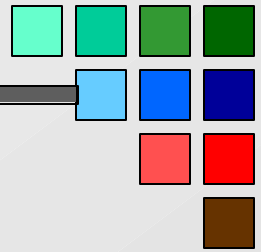
Sistemas Distribuídos

Ricardo Ribeiro dos Santos

ricrs@ec.ucdb.br

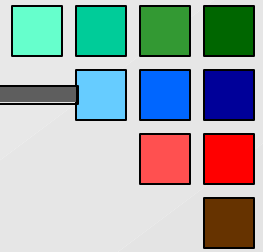
Curso de Engenharia de Computação – UCDB – Agosto/2003

Tópicos



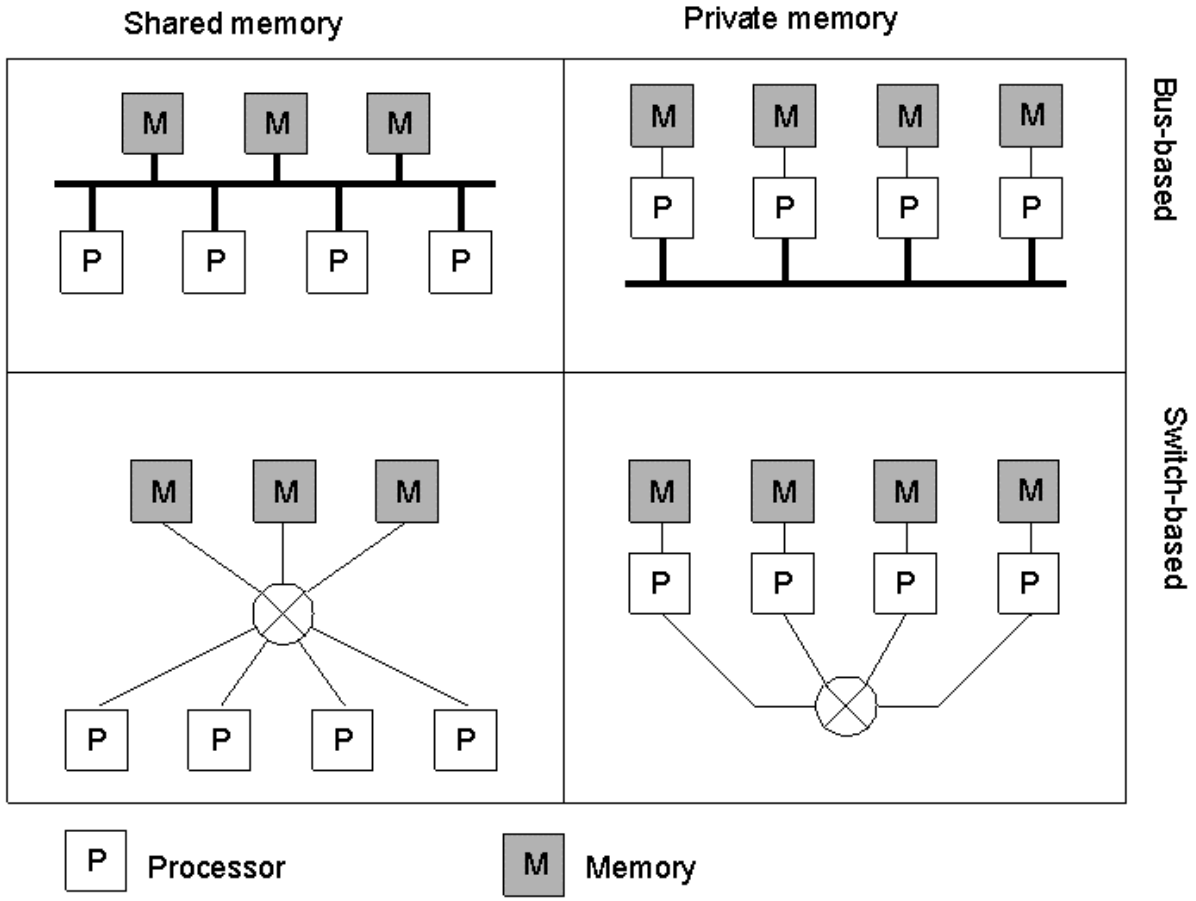
- Conceitos de HW em SD
  - Multiprocessadores e Multicomputadores
- Conceitos de SW em SD
  - DOS
  - NOS
  - Middleware
  - Modelo C/S

Conceitos de HW em SD



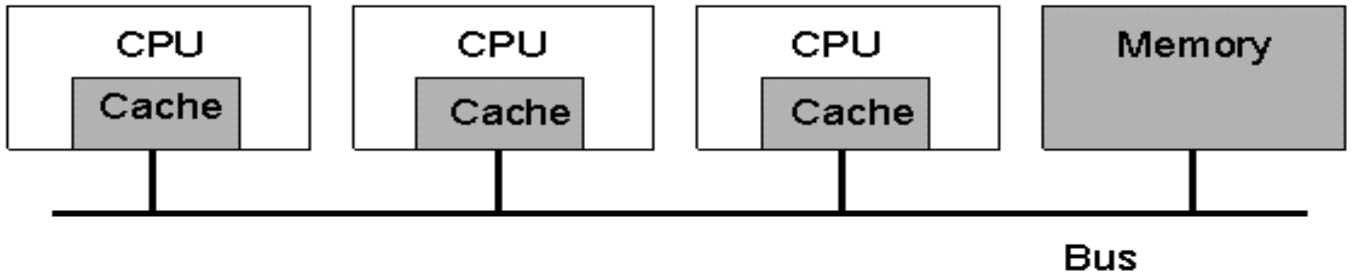
- Modelos arquiteturais indicam as possibilidades de organizar o HW disponível para fazer SDs!
  - ET/Servidores; Banco de Proc.; Minicomputadores
- SDs consistem de múltiplas CPUs!
- A partir dessas “múltiplas” CPUs...
  - Como são interconectadas?
  - Como a comunicação é realizada?
  - Homogeneidade X Heterogeneidade

Algumas organizações de HW para SD

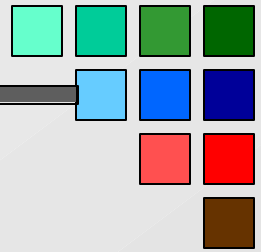


Conceitos de HW em SD

- Multiprocessadores
  - Acesso direito à memória compartilhada
  - Necessidade de controle de coerência para o acesso a essa memória
  - Introdução de caches
    - Novo problema! Como fazer com que os dados de um cache não fiquem incoerentes?



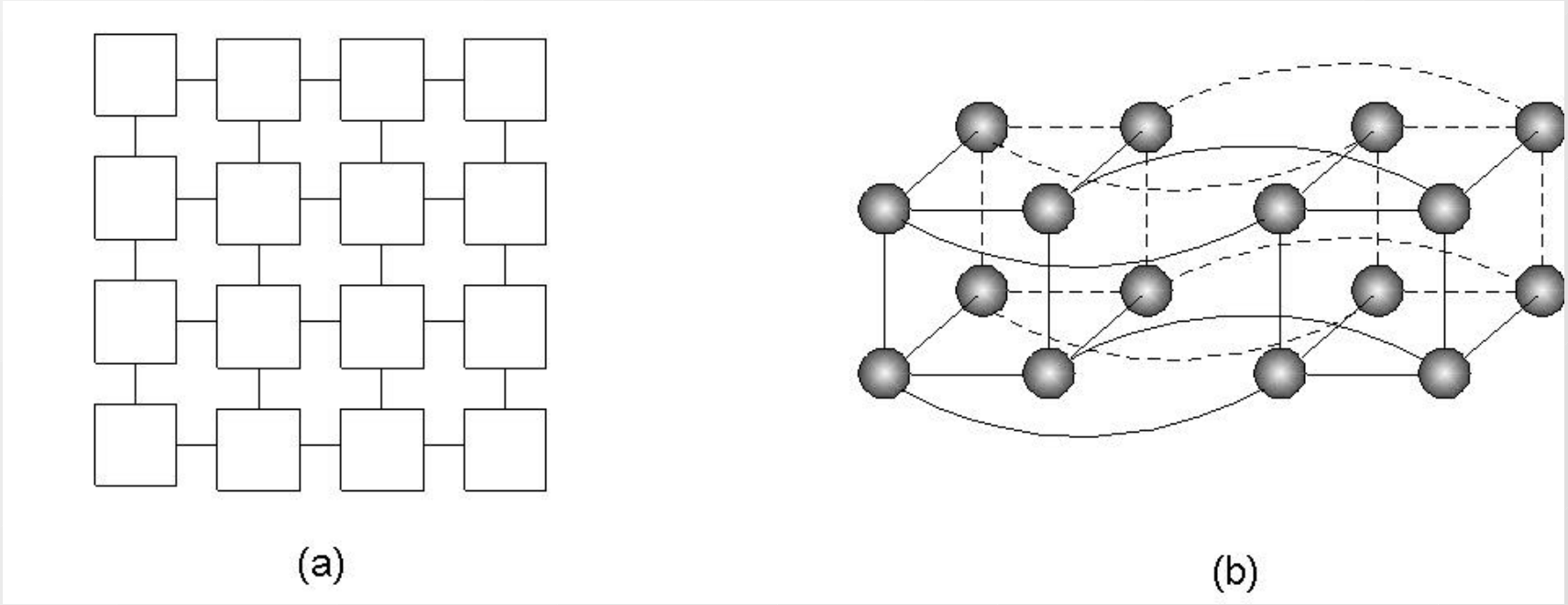
Conceitos de HW em SD



- Multicomputadores
  - Cada CPU possui sua própria memória
  - Em muitas situações, as CPUs são as mesmas utilizadas em PCs (em termos de capacidade de processamento)
  - O atrativo reside na rede de interconexão...

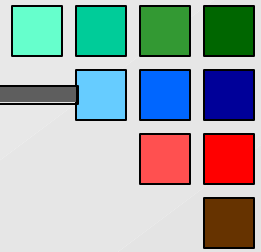
Conceitos de HW em SD

- Alguns exemplos de multicomputadores



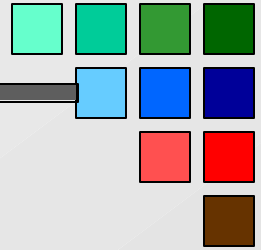
- (a) Grid
- (b) Hypercube

Conceitos de SW em SD



- Em SD, HW é importante mas é o SW que determina se é ou não um Sistema Distribuído!
- Enquanto que os conceitos de HW se preocupam em como organizar os **processadores...**
- Os conceitos de SW se preocupam em como organizar os **processos**
- Em termos de SO, não há mudança de objetivo com relação aos SOs tradicionais
  - De forma geral é um gerenciador de recursos

Conceitos de SW em SD

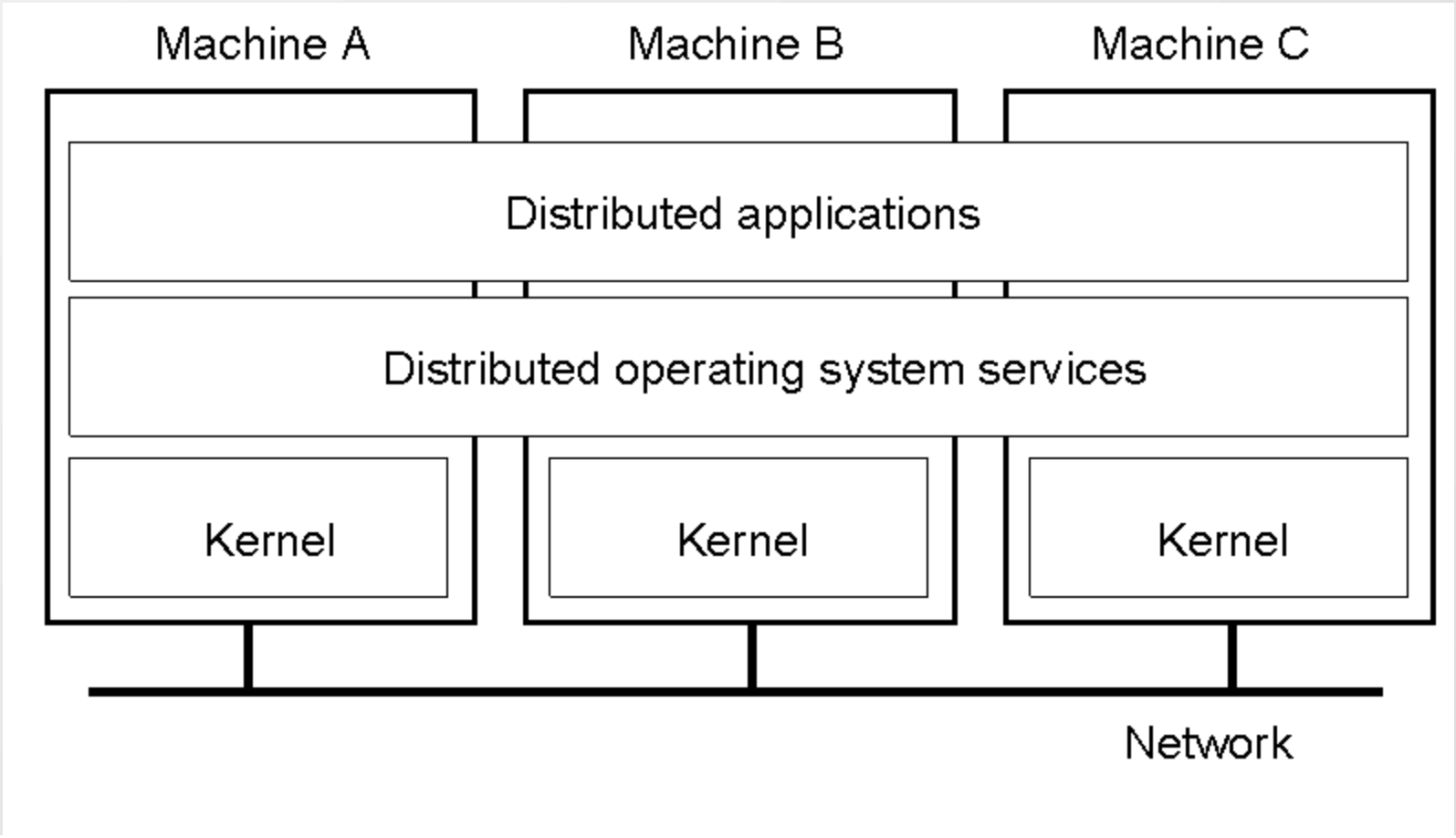


- Uma divisão dos SWs em SD consistem em DOS, NOS e Middleware

Sistema	Descrição	Principal Objetivo
DOS	SO fortemente acoplado para multiprocessadores e multicomputadores homogêneos	“Esconder” e gerenciar recursos de HW
NOS	SO fracamente acoplado para multicomputadores heterogêneos (LAN e WAN)	Oferecer serviços para clientes remotos
Middleware	Camada adicional sobre NOS que implementa serviços de propósito geral	Fornece transparência

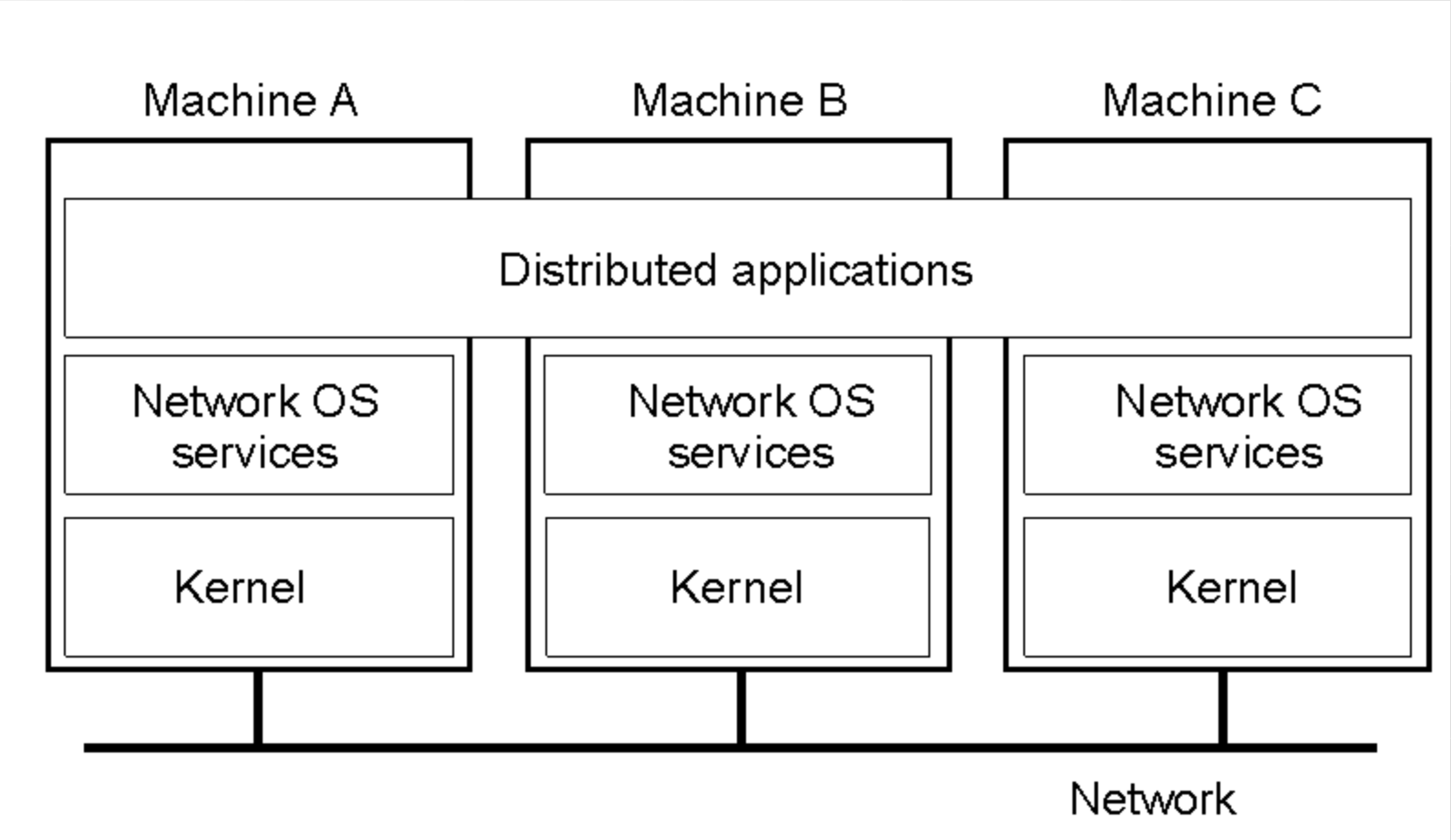
Conceitos de SW em SD

- DOS (Distributed Operating System)
  - Exemplo: SO para um Multicomputador



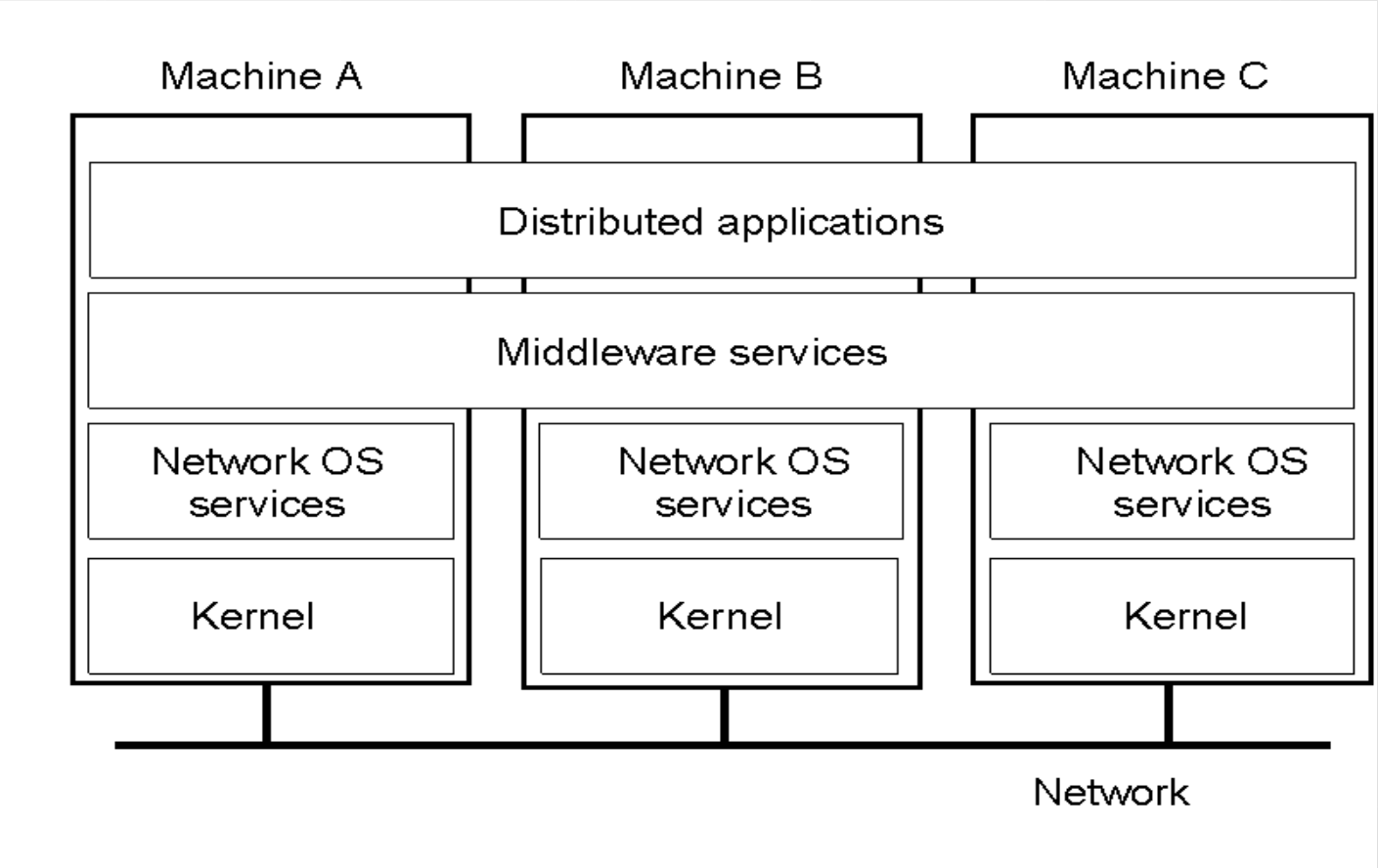
Conceitos de SW em SD

- NOS (Network Operating System)

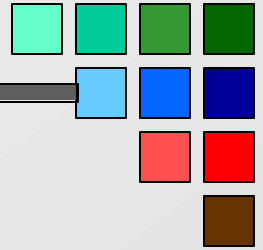


Conceitos de SW em SD

- Middleware



Conceitos de SW em SD

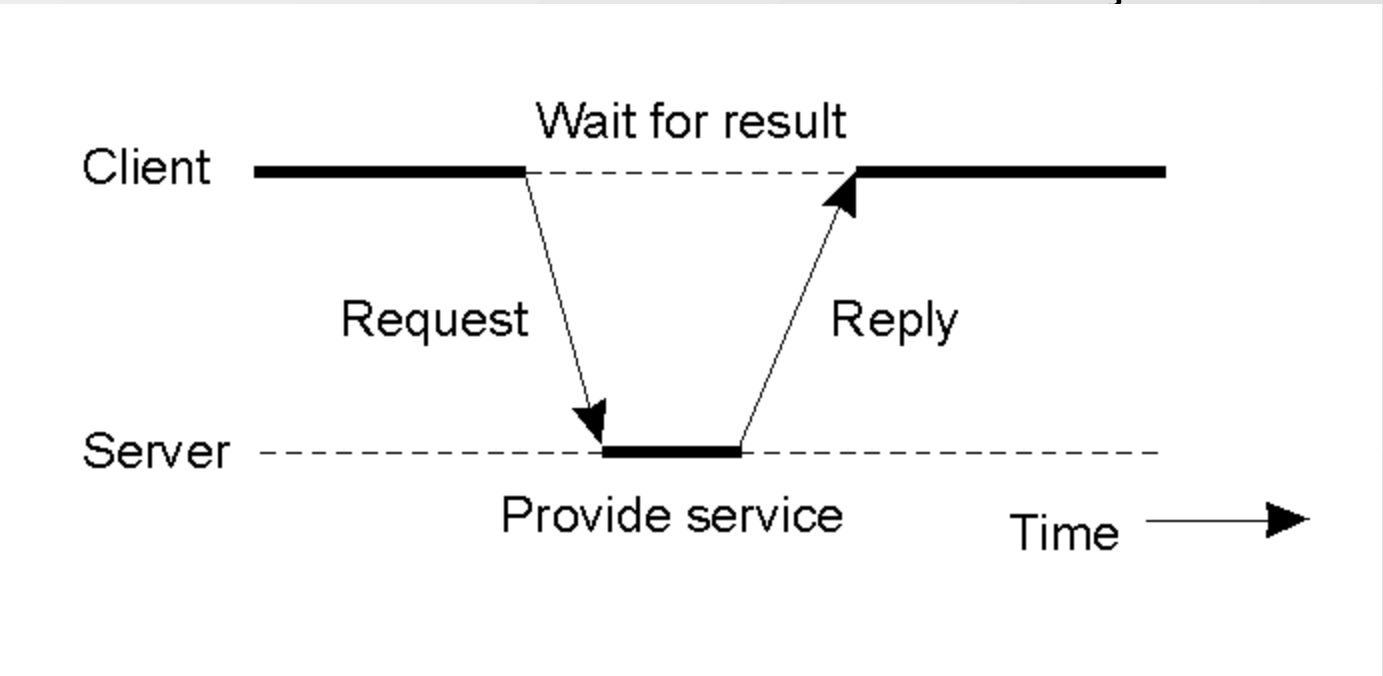


- Comparação entre as três abordagens para SW em SD

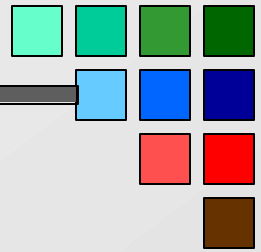
Item	**DOS**		**NOS**	**SO baseados em *Middleware***
	Multiproc.	**Multicomp.**		
Grau de transparência	Muito Alto	Alto	Baixo	Alto
Mesmo SO em todos os nós	Sim	Sim	Não	Não
Número de cópias do SO	1	N	N	N
Base para comunicação	Memória compartilhada	Mensagens	Arquivos	Modelo específico
Gerenciamento de recursos	Global, central	Global, distribuído	Por nó	Por nó

Conceitos de SW em SD

- Modelo Cliente/Servidor
  - Organização das aplicações em um SD
  - Clientes: Requisitam serviços
  - Servidores: Atendem a esses serviços



Conceitos de SW em SD

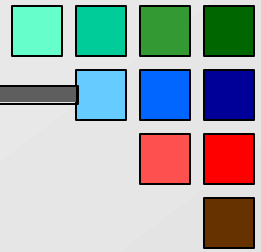


- Modelo Cliente/Servidor
  - Um exemplo baseado em primitivas *Send/Receive*

Arquivo de def. de cabeçalhos

```
/* Definitions needed by clients and servers.                    */
#define TRUE                1
#define MAX_PATH            255   /* maximum length of file name          */
#define BUF_SIZE            1024  /* how much data to transfer at once    */
#define FILE_SERVER         243   /* file server's network address        */

/* Definitions of the allowed operations */
#define CREATE              1     /* create a new file                    */
#define READ                2     /* read data from a file and return it  */
#define WRITE               3     /* write data to a file                 */
#define DELETE              4     /* delete an existing file              */

/* Error codes. */
#define OK                  0     /* operation performed correctly        */
#define E_BAD_OPCODE        -1    /* unknown operation requested          */
#define E_BAD_PARAM         -2    /* error in a parameter                 */
#define E_IO                -3    /* disk error or other I/O error        */

/* Definition of the message format. */
struct message {
      long source;                /* sender's identity                    */
      long dest;                  /* receiver's identity                  */
      long opcode;                /* requested operation                  */
      long count;                 /* number of bytes to transfer          */
      long offset;                /* position in file to start I/O        */
      long result;                /* result of the operation              */
      char name[MAX_PATH];        /* name of file being operated on       */
      char data[BUF_SIZE];        /* data to be read or written           */
};
```

Conceitos de SW em SD

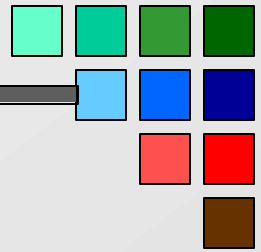


- Modelo Cliente/Servidor
  - Um exemplo baseado em primitivas *Send/Receive*

Servidor

```
#include <header.h>
void main(void) {
     struct message ml, m2;                  /* incoming and outgoing messages      */
     int r;                                  /* result code                         */

     while(TRUE) {                           /* server runs forever                 */
          receive(FILE_SERVER, &ml);         /* block waiting for a message         */
          switch(ml.opcode) {                /* dispatch on type of request         */
               case CREATE:    r = do_create(&ml, &m2); break;
               case READ:      r = do_read(&ml, &m2); break;
               case WRITE:     r = do_write(&ml, &m2); break;
               case DELETE:    r = do_delete(&ml, &m2); break;
               default:        r = E_BAD_OPCODE;
          }
          m2.result = r;                     /* return result to client             */
          send(ml.source, &m2);              /* send reply                          */
     }
}
```

Conceitos de SW em SD



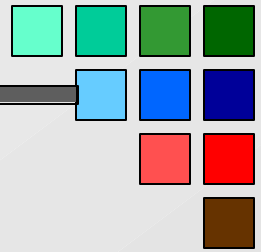
- Modelo Cliente/Servidor
  - Um exemplo baseado em primitivas *Send/Receive*

Cliente

(a)

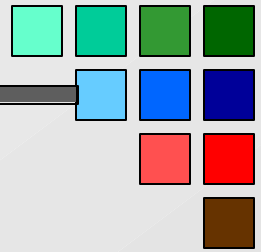
```
#include <header.h>
int copy(char *src, char *dst){
      struct message ml;                    /* procedure to copy file using the server   */
      long position;                        /* message buffer                            */
      long client = 110;                    /* current file position                     */
                                            /* client's address                          */
      initialize( );                        /* prepare for execution                     */
      position = 0;
      do {
            ml.opcode = READ;               /* operation is a read                       */
            ml.offset = position;           /* current position in the file              */
            ml.count  = BUF_SIZE;                                                         /* how many bytes to read*/
            strcpy(&ml.name, src);          /* copy name of file to be read to message   */
            send(FILESERVER, &ml);          /* send the message to the file server       */
            receive(client, &ml);           /* block waiting for the reply               */

            /* Write the data just received to the destination file.                     */
            ml.opcode = WRITE;              /* operation is a write                      */
            ml.offset = position;           /* current position in the file              */
            ml.count  = ml.result;          /* how many bytes to write                   */
            strcpy(&ml.name, dst);          /* copy name of file to be written to buf    */
            send(FILE_SERVER, &ml);         /* send the message to the file server       */
            receive(client, &ml);           /* block waiting for the reply               */
            position += ml.result;          /* ml.result is number of bytes written      */
      } while( ml.result > 0 );             /* iterate until done                        */
      return(ml.result >= 0 ? OK : ml result);   /* return OK or error code            */
}
```

Conceitos de SW em SD



- Modelo Cliente/Servidor sempre é alvo de debates e controvérsias...
  - Questão principal: Definir uma clara separação entre cliente/servidor
  - Ex: Um Servidor para um BD distribuído pode funcionar como um cliente para diferentes servidores de arquivo!

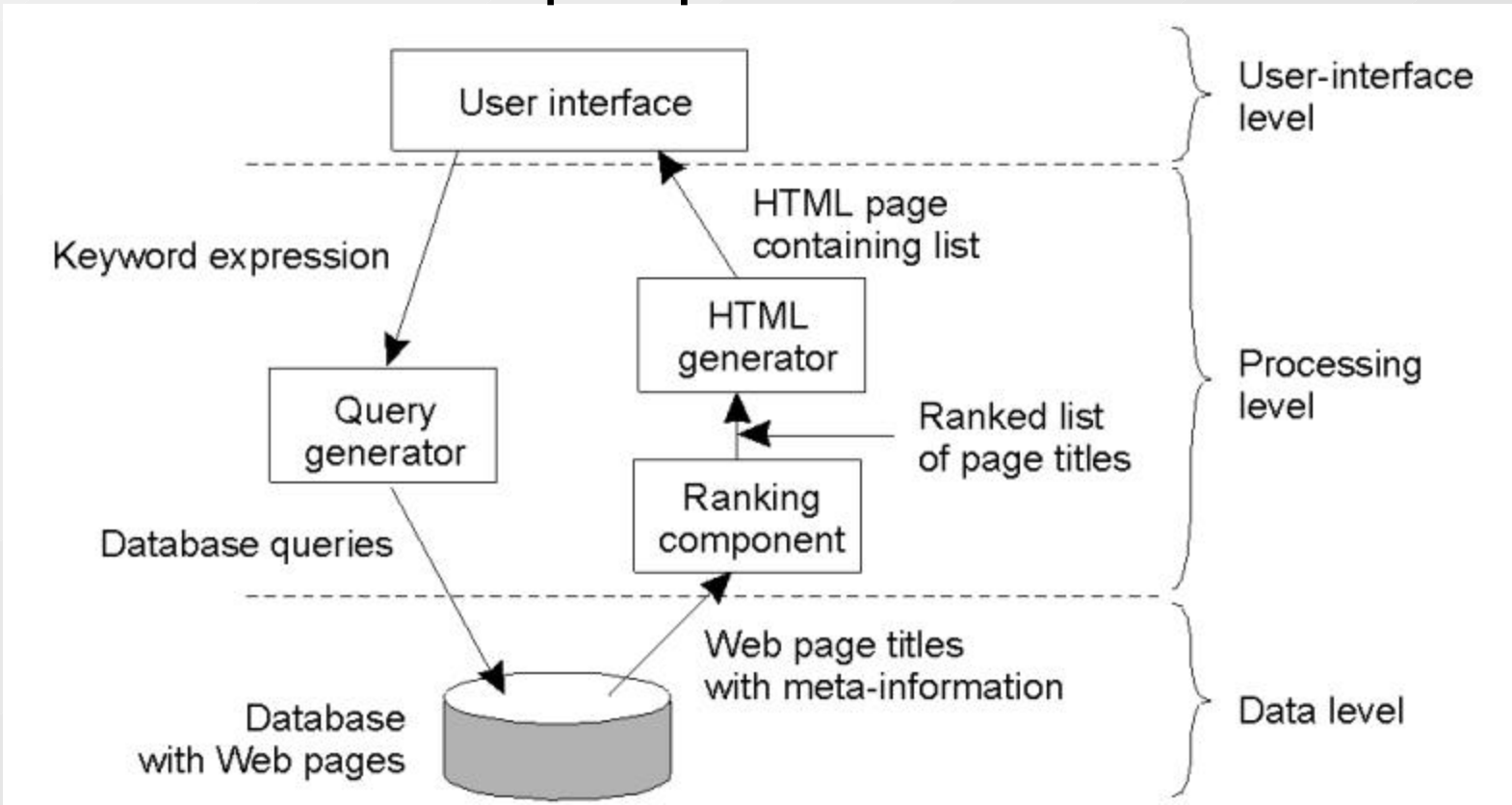
Conceitos de SW em SD



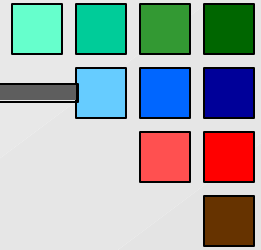
- Mesmo com diferentes sentimentos (*feelings*), concorda-se com a identificação de três níveis em aplicações cliente/servidor
  - O nível de interface do usuário
    - Interação com a aplicação; Implementado pelo cliente
  - O nível de processamento
    - Não há um aspecto “claro” para definir o que fica no nível de processamento; Geralmente, é o “core” da aplicação
  - O nível de dados
    - Programas que manipulam os dados que as aplicações necessitam; Geralmente, fica no lado servidor

Conceitos de SW em SD

- Um exemplo do modelo C/S em três diferentes camadas
  - Um sistema de pesquisas na Internet



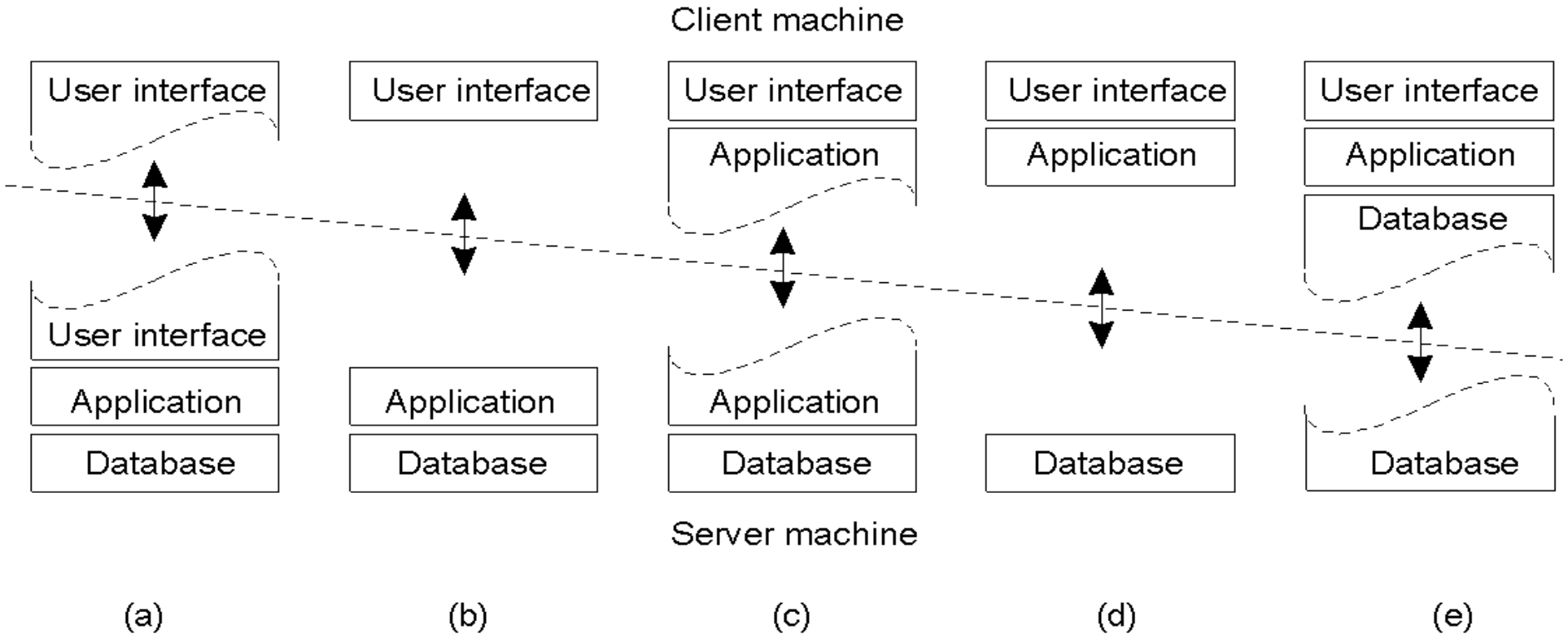
Conceitos de SW em SD



- Arquiteturas Cliente/Servidor
  - Apesar da divisão em camadas, sempre surge as dúvidas...
    - O que fica no cliente?
    - O que fica no servidor?
    - E se o servidor também funciona como cliente?
  - Não há solução única!
    - Geralmente, leva-se em consideração um conjunto de fatores para determinar como dividir a aplicação C/S
      - Nº de usuários, serviços, capacidade do HW

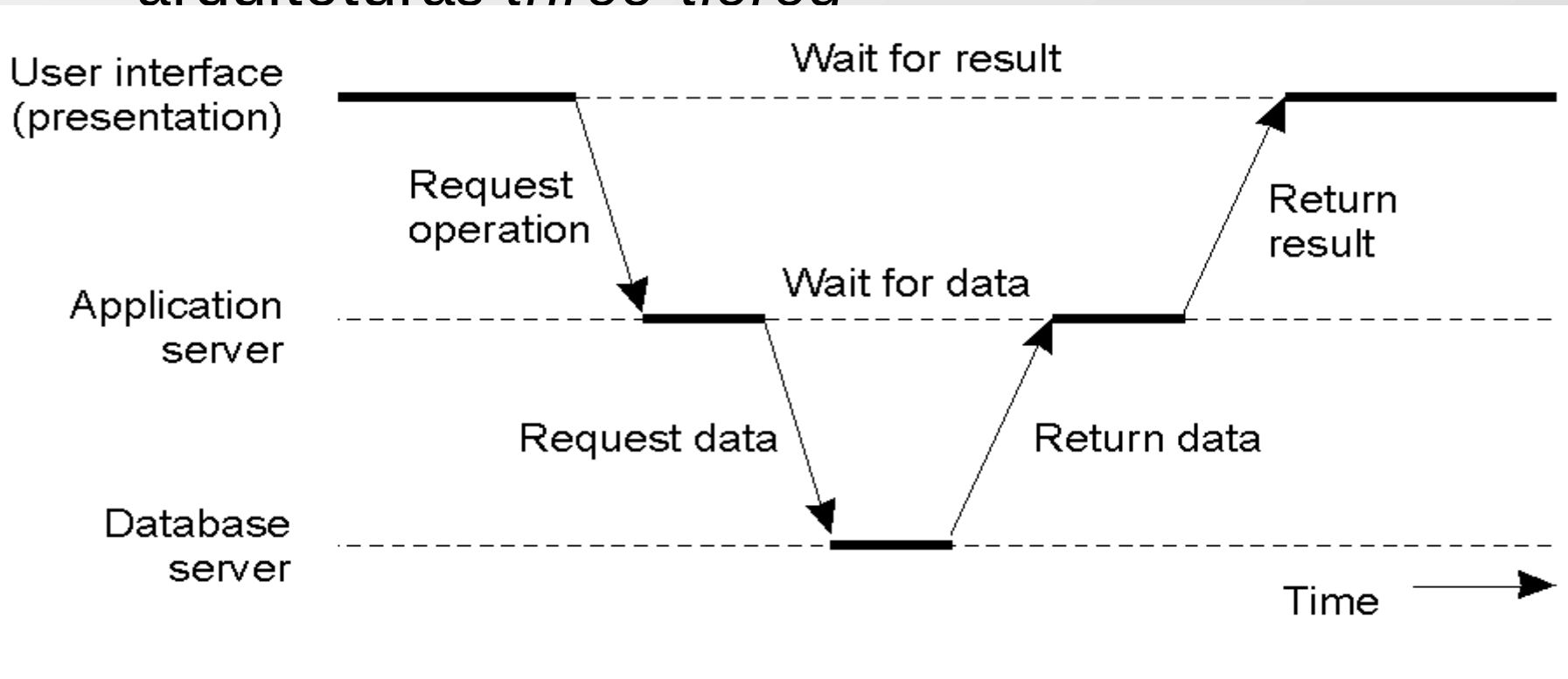
Conceitos de SW em SD

- Arquiteturas Cliente/Servidor
  - Exemplos de possibilidades de separação de funcionalidades em uma arquitetura *two-tiered*

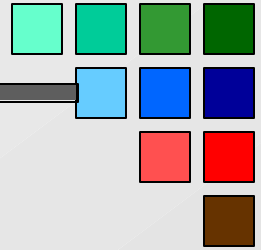


Conceitos de SW em SD

- Arquiteturas Cliente/Servidor
  - As situações onde o servidor também pode funcionar como cliente, são denominadas de arquiteturas *three-tiered*

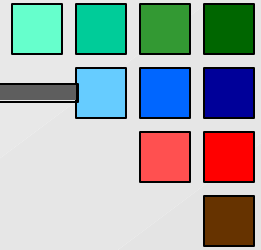


Conceitos de SW em SD



- Arquiteturas Cliente/Servidor
  - A distribuição das camadas (discutidas anteriormente) correspondem diretamente a organização lógica da aplicação
    - Isto é chamado de organização vertical
    - Diferentes componentes lógicos em diferentes máquinas...
    - Relacionado ao conceito de **fragmentação vertical** usado em BD distribuídos

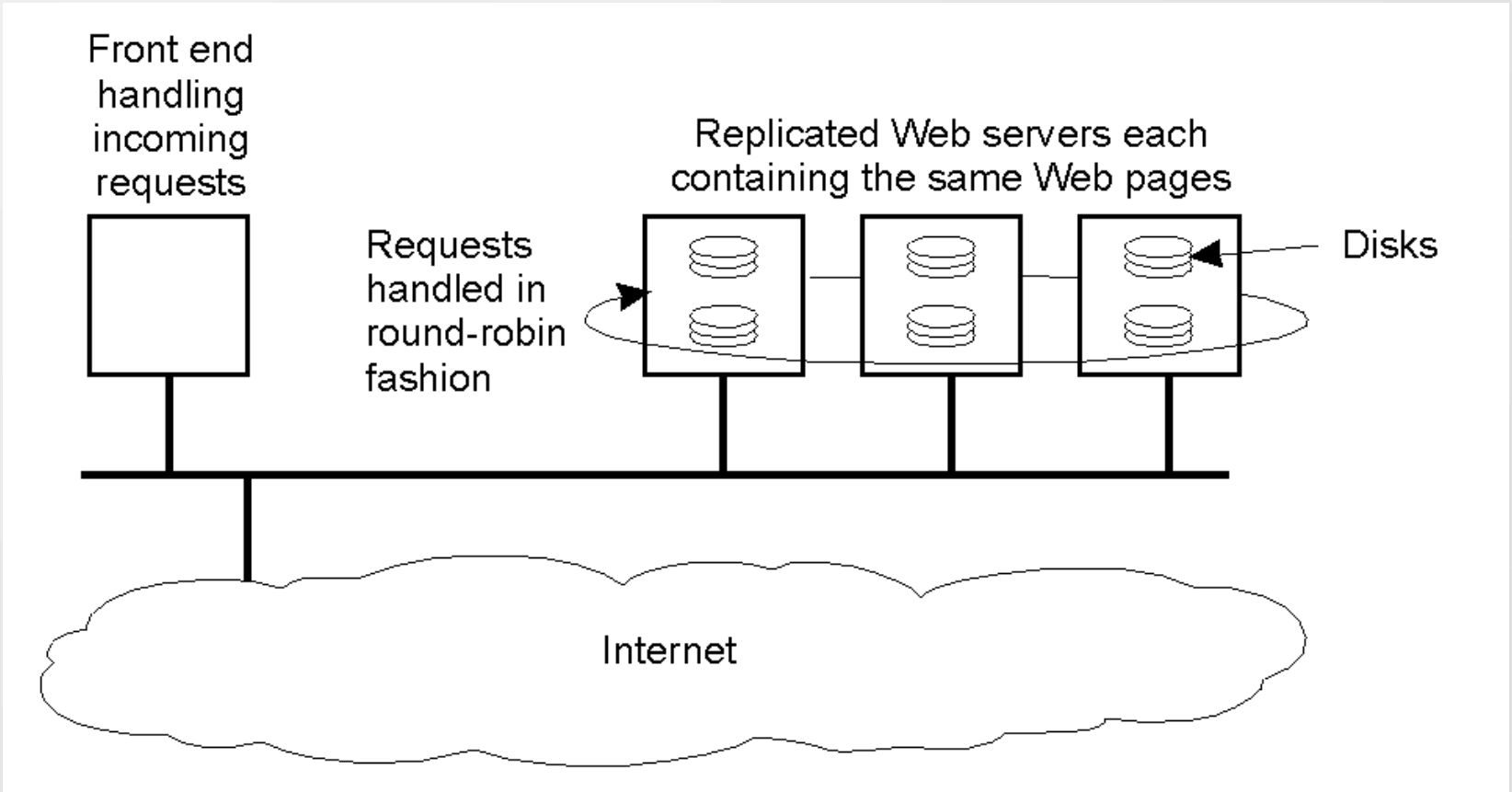
Conceitos de SW em SD



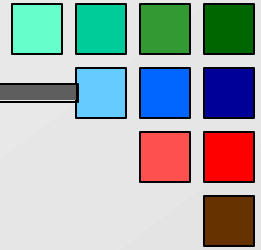
- Arquiteturas Cliente/Servidor
  - Atualmente, é comum encontrarmos uma distribuição **horizontal** de clientes e servidores!
  - No entanto cada parte dividida (geralmente, os servidores) operam em um conjunto de dados completo com o intuito de balancear as cargas
    - Ex: Servidores Web replicados em várias máquinas

Conceitos de SW em SD

- Arquiteturas Cliente/Servidor
  - Exemplo de distribuição horizontal



Conceitos de SW em SD



- Arquiteturas Cliente/Servidor
  - No exemplo anterior cada Servidor Web possui o mesmo conjunto de páginas
  - Cada vez que uma cópia é atualizada, deve ser copiada para cada um dos outros servidores
  - Requisições enviadas por clientes são escalonadas entre os servidores através de uma política *Round-Robin*