

2020

1 - transparência

1. What is meant by *transparency* in the context of distributed systems? Give examples, at least five, of the modes transparency may assume. Present your claims in clear terms. (3 points)

- transparência faz com que um sistema distribuído seja capaz de se apresentar com um único computador, ou seja,
- transparência é uma característica que expressa um sucesso maior ou menor de mascarar a complexidade da camada interior
- consiste em mascarar o facto de que os recursos e processo estão distribuídos em vários computadores .
- exemplos:
 - transparência de acesso - quando as mesmas operações são feitas para aceder recursos locais e remotos
 - transparência de posição - quando um acesso de um recurso é levado sem o conhecimento da sua localização física ou de rede
 - transparência de rede - quando transparência de acesso e posição existem ao mesmo tempo
 - transparência de movimento - quando a localização do acesso do cliente aos recursos pode mudar dentro do sistema sem afetar a operação a ser feita
 - transparência de migração -quando os recursos podem ser movidos sem afetar o seu acesso
 - transparência de relocalização- quando os recursos pode ser movidos sem afetar o seu acesso, mesmo quando um acesso está a ocorrer (versão mais forte de transparência de migração)
 - transparência de replicação - quando é possível instanciar várias cópias do mesmo recurso sem que isso se torne óbvio
 - transparência de desempenho - quando uma reconfiguração dinâmica do sistema pode ocorrer para lidar com as variações de carga
 - transparência de escala - quando o sistema e os aplicativos podem expandir em escala sem exigir qualquer alteração na estrutura do sistema e no algoritmo da aplicação
 - transparência de simultaneidade - quando o acesso a recursos compartilhados é realizado em paralelo por várias entidades sem estarem cientes umas das outras (os dados devem permanecer sempre consistente)
 - transparência de falha - quando falhas, ocorrendo em componentes de hardware e / ou software do sistema, podem ser mascarados e, portanto, as tarefas em execução podem ser encerradas.

2- RMI remoto e local; servidor ; migração de código

2. Take *method invocation on remote objects* as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system and assume that Java RMI was used to implement it.

Answer the following questions in a clear manner. (1 point each)

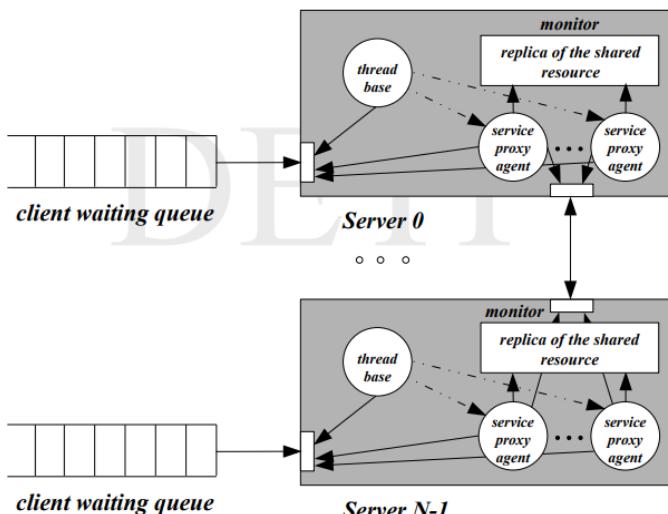
- i. What are the main differences between invoking a method on a remote and a local object?
- ii. Sketch a diagram describing the functional interaction of the different components of such a system at the server side.
- iii. When code migration for remote execution is being considered, it is essential to protect the object, and the hardware platform where it resides, against the execution of malicious code. Explain how this may be achieved.

i.

- O método de invocação remota(RMI) permite que haja interação de programas que estão a ser executados em diferentes nós de um sistema distribuído.(pelas RPC - Remote Producer Calls)
- Com o RMI a transparência de objetos é conseguida uma vez que o nível de abstração permite invocar objetos remotos da mesma maneira que se invocam objetos locais.
- existem 3 características a ter em atenção , pois uma RMI não funciona exatamente da mesma forma que a chamada de procedimento local:
 - A chamada pode falhar, pois pode ainda não estar instalada, ou a infraestrutura de comunicação não estar a funcionar corretamente.
 - Os parâmetros do procedimento e de valores de return devem ser passado por valor , pois como estão em diferentes endereços, o meio de comunicação entre eles é por passar dados relevantes juntamente com o seu formato e estrutura.
 - e o procedimento de execução remota demora mais tempo que a execução de procedimentos local.

ii.

Resource replication



- disponibilizado simultaneamente em vários sistemas informáticos, cada um executando uma variante tipo 2 do servidor

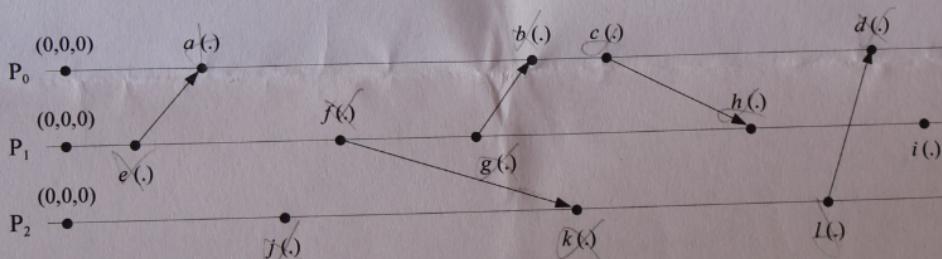
- o recurso compartilhado é, assim, replicado em cada servidor, dando origem a múltiplas cópias
- é um modelo sofisticado que visa maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga
 - o serviço é mantido operacional contra a falha de servidores específicos
 - as solicitações do cliente são distribuídas entre os servidores disponíveis usando uma política, pré-fornecido pelo serviço DNS, de associação geográfica para solicitações globais e de associação rotativa para solicitações locais
 - Quando há uma alteração de dados locais em uma das réplicas do recurso compartilhado, a necessidade de manter as diferentes réplicas consistentes surge e deve ser tratado.

iii.

- Migração de código permite alguns componentes a serem transferidos de um nó para outro do sistema, durante a sua execução, mas isto vem com alguns problemas de segurança, então tem de se garantir que o código recebido no nó destino não ponha em risco a integridade dos recursos do sistema onde o nó está. é então introduzido uma componente de código que gerencia a migração que vê quais acessos são permitidos ou negados. Esta segurança pode ser feita com o uso de mensagens que são recebidas como um array de bytes, que tem os parâmetros, o tipo e também como são estruturados, e esta forma é chamada de marshaling, que ao chegar ao destino faz a operação de unmarshalling.

3 - temporal , synchronized, Mattern and fidge algoritmo

3. The diagram illustrates the *temporal* evolution of three processes whose local clocks are vector logical clocks synchronized according to the Mattern and Fidge algorithm. (1 point each)



- i. Assign to the different events, specified by small letters ($a \dots l$), their associated time stamp.
- ii. Take from the schematics three examples of pairs of concurrent events (\parallel) and list the longest sequence of sequential ones ($->$).
- iii. Explain why logical clocks, either of the scalar or the vector type, are superior to ordinary time clocks in eliciting a causal ordering of events. Present your claims in clear terms.

i.

$$\begin{aligned}
 & a(1,1,0) \quad b(2,3,0) \quad c(3,3,0) \quad d(4,3,3) \\
 & e(0,1,0) \quad f(0,2,0) \quad g(0,3,0) \quad h(3,4,0) \quad i(3,5,0) \\
 & j(0,0,1) \quad k(0,2,2) \quad l(0,2,3)
 \end{aligned}$$

ii.

acho que a mais longa é : e -> a e g->b e c->h ou seja V1(e)<V1(h)
concorrentes : j || i (certeza) acho: j || h k|| i

iii.

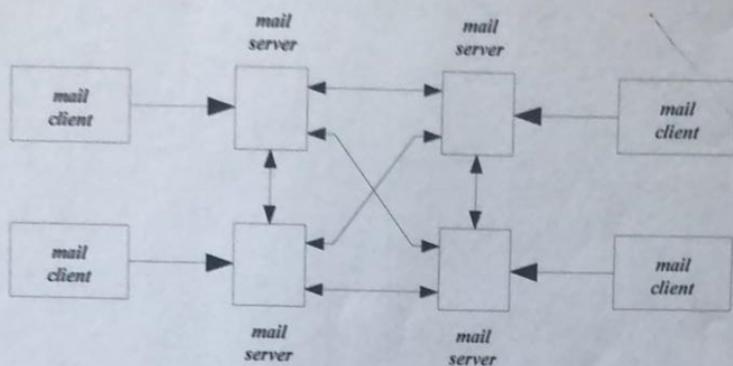
Lamport sugeriu um dispositivo que ele chamou como relógio lógico que é essencialmente um contador local de eventos que não tem qualquer associação com hora real. O que faz com que seja bastante superior no que toca a um relógio de tempo normal , dado que o tempo não se pode medir diretamente (astronômico/ atômico) é calculado um padrão definido por uma média, que dependendo das condições, o que não é tão eficaz como um relógio lógico .

4 - peer-to-peer ; leader

clocks in eliciting a causal ordering of events. Present your claims in a clear and logical manner. You may use pseudocode or pseudo-code to describe your algorithm.
4. Consider a situation in the context of peer-to-peer communication where three processes, P1, P2 are P3 are required to elect a leader. Describe in detail the main steps of an algorithm that carries out the election. Assume there are no message loss and no process fails. How would you change the algorithm if these hypothesis were no longer valid? (3 points)

- Na comunicação peer-to-peer , os processos que implementam as atividades cooperam entre eles , não existindo distinção entre eles. Logo , como não há perda de mensagens ou falhas de processo, cada peer assume-se a si próprio como líder, gerando um nº de identificação com o qual vai marcar a mensagem a passar aos peers seguintes. Cada peer vai comparar o seu nº de identificação com o que está na mensagem e se o seu número for menor que o que está na mensagem, passa a mensagem sem qualquer modificação , se acontecer o contrário , troca o número do líder pelo seu próprio. Assim sendo quando a mensagem voltar ao peer que a originou ele verifica se o número de identificação é o seu, se for ele ficará como líder.

1. Electronic mail, or e-mail, depicted in the schematics below, is a distributed application that follows the principles of the *Publisher-Subscriber Model*.



How does this model operate and what are its main components? Match the properties of these components to the aggregate of mail servers and clients that embody the mailing service. Present clearly your claims. (3 points)

No modelo publisher-subscriber existem vários provedores de serviço, publisher, e vários receptores desses serviços, subscribers que se encontram dissociados uns dos outros por meio de mediação de um serviço intermediário , um broker. Como já foi dito este modelo electronic mail, or e-mail segue os princípios do modelo publisher-subscriber, pois os servidores de mail trabalham como brokers para os clientes de mail e outros servidores de mail, para gerenciamento de mensagens locais e para encaminhamento de mensagens remotas. E os clientes de mail trabalham como publishers para o envio de mensagens e como subscribers para receber as mensagens.

2- 3 variantes do modelo client-server

2. Take message passing as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Three variants of this model were studied. Describe them. Which of these is more commonly used and which is more reliable? Present clearly your claims. (2 points)

Refer in this context to what does one mean by *message marshaling* and *unmarshaling*. How are they enforced in Java? (1 point)

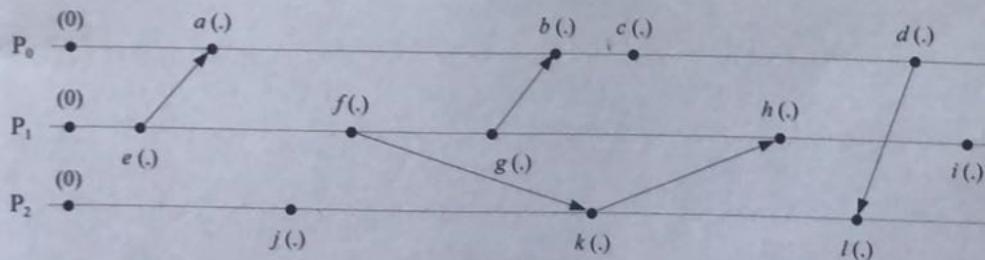
- Variante do tipo 1 (solicitação de serialização) - apenas um processo do cliente é atendido de cada vez : isso significa que a thread base, depois de receber uma solicitação de conexão, instancia um agente proxy de serviço e espera que acabe antes de começar a ouvir outra vez. Não necessita de nenhuma segurança para garantir exclusão mútua. Sendo este modelo é bastante ineficiente pois não aproveita os tempos mortos de interação devido a falta de competição.
- Variante do tipo 2 (replicação do servidor) - os processos do cliente são atendidos simultaneamente, o que significa que a thread base, mediante receber uma solicitação de conexão, instancia um agente proxy de serviço e inicia ouvindo de novo. Para garantir a exclusão mútua, o recurso compartilhado é transformado em um monitor. Sendo que esta variante é mais comum das 3. O tempo é minimizado,

pois aproveita os tempos mortos através da simultaneidade. Permite a sincronização de diferentes processos de cliente.

- Variante do tipo 3 (Replicação do recursos) - o serviço é disponibilizado simultaneamente em vários sistemas, cada um executando uma variante do tipo 2. O recurso compartilhado é replicado em cada servidor, dando origem a múltiplas cópias. Este modelo visa maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga . O servidor é mantido operacional contra falhas de servidores específicos.
- Sendo o tipo 2 mais tradicionalmente utilizado, o tipo 3 é mais de confiança contra falhas e o tempo é minimizado.
- Marshaling e unmarshalling de informação é uma forma de garantir que a mensagem transmitida pelo canal de comunicação seguir segura, é vista como uma matriz de bytes que inclui os parâmetros, o seu tipo e como são estruturados, a construção de uma mensagem destas é chamada de marshaling e a operação de decodificação desta mensagem é chamada de unmarshalling

3- lamport

3. The diagram illustrates the *temporal evolution* of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm. (1 point each)



- i. Assign to the different events, specified by small letters ($a \dots l$), their associated time stamp.
- ii. Take from the schematics three examples of pairs of concurrent events (\parallel) and list the longest sequence of sequential ones (\rightarrow).

1

29 de Junho de 2020

- iii. Explain why logical clocks, either of the scalar or the vector type, are superior to ordinary time clocks in eliciting a causal ordering of events. Present clearly your claims.

- a(1) b(3) c(4) d(5)
e(0) f(1) g(2) h(3) i(4)
j(0) k(2) l(6)

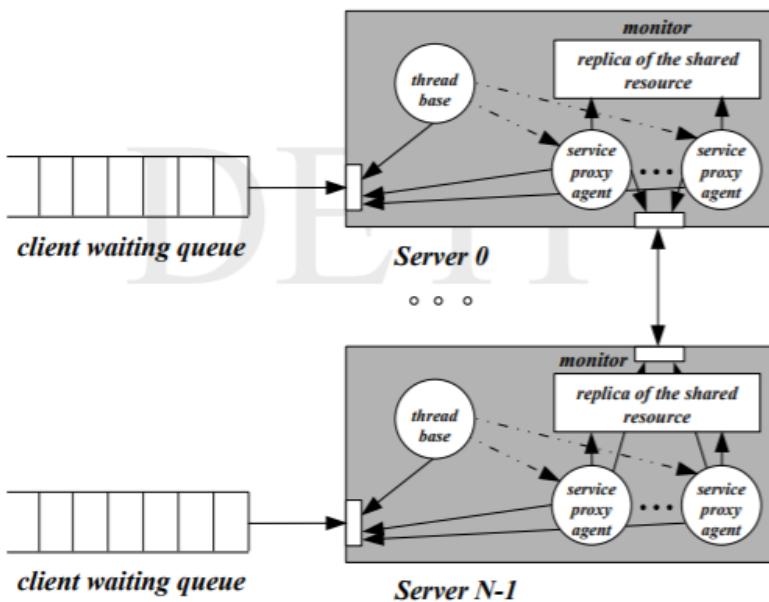
- ii.
concorrentes: $j \parallel f$, $k \parallel c$, $d \parallel i$
mais longa : $f \rightarrow k \rightarrow h \rightarrow l \rightarrow i$

- iii.

4-

4. Suppose one aims to improve service availability in a client-server model. In order to achieve this goal, a resource replication variant of the server is installed in two hardware platforms, located side by side. Only one of the units is active at a given time. However, if it fails, the other unit must take immediately its place in dealing with client requests.

Sketch a diagram describing the required organization and list three problems that must be solved so that the system may work as expected. Present clearly your claims. (3 points)

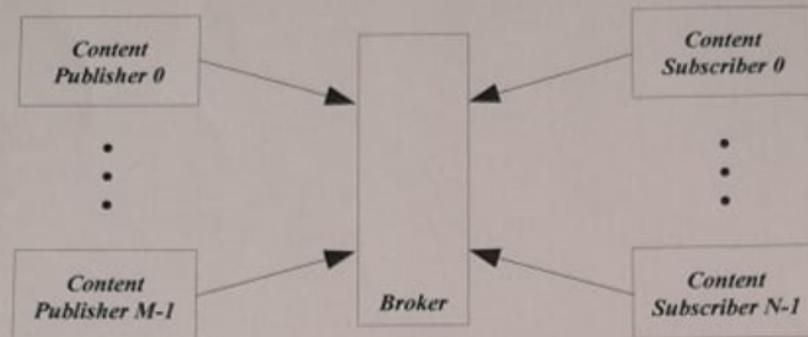


- problema da consistência de informação que surge quando há uma alteração de dados locais numa das réplicas do recurso partilhado, evidenciando assim a necessidade de manter as diferentes réplicas consistentes (lock dos dados mantendo a consistência da informação).
- O outro problema conhecido é que os servidores podem ser implementados em sistemas diferentes e neste caso é necessário assegurar que os sistemas tenham algumas semelhanças, nomeadamente a mesma capacidade a nível de memória.

2019 - Normal

1- publisher-subscriber e-commerce

1. The diagram below describes the main components of the *Publisher-Subscriber Model*.



(operando de uma aplicação real)

Describe how it operates in an e-commerce application. Are the interactions that take place synchronous or asynchronous? Present clearly your claims.

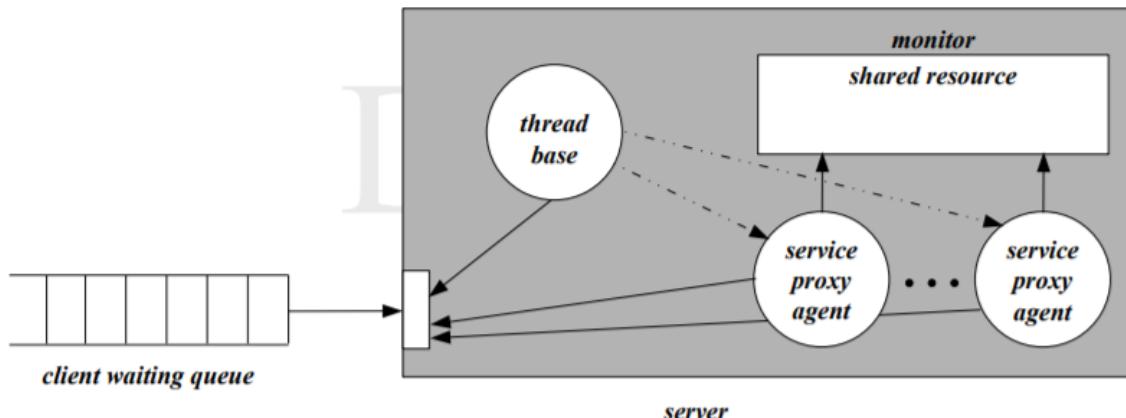
No modelo publisher-subscriber existem vários provedores de serviço, publisher, e vários receptores desses serviços, subscribers que se encontram dissociados uns dos outros por meio de mediação de um serviço intermediário , um broker. E uma das principais características deste modelo , em contraste com o modelo cliente-servidor convencional, é que não existe nenhuma interação síncrona funcionando aqui ou seja as interações são feitas de forma assíncrona. Como já foi dito o diagrama trata se de um modelo e-commerce, nos quais os vendedores agem como publishers que põem a informação sobre os produtos à venda na plataforma de negociação, enquanto os compradores funcionam como subscribers que pedem informação sobre produtos na plataforma que queiram comprar. A plataforma de negociação é o broker que gerencia possíveis transações entre eles.

2- RMI, diagrama de client-server , transparência

2. Take access to remote objects, implemented by JAVA RMI, as the communication paradigm which was selected to establish a client-server model among processes residing in a distributed system. Draw a diagram that describes the functional interaction among the components of such a model, both at client and at the server side, identifying each of them and explaining their role in the interaction. Assume the server replication variant.

Refer in this context to how object transparency is implemented.

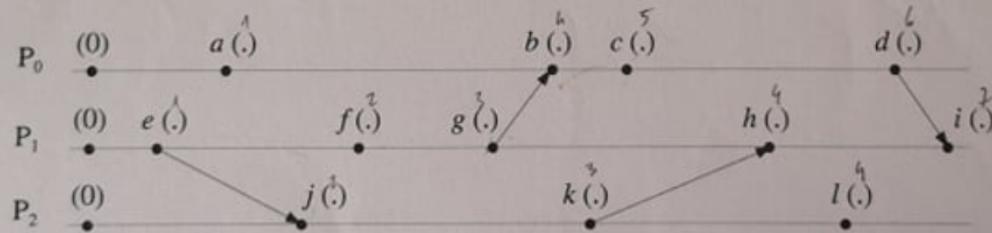
Desenho da descrição funcional base do modelo



- os processos do cliente são atendidos simultaneamente, isso é, a thread base ao receber uma solicitação de conexão instancia um agente proxy de serviço e começa a ouvir novamente por mais solicitações
- o recurso partilhado é transformado num monitor para garantir exclusão mutua, dado que agora existem varios agentes proxy de serviço ativos simultaneamente.
- Com o RMI a transparência de objetos é conseguida uma vez que o nível de abstração permite invocar objetos remotos da mesma maneira que se invocam objetos locais. Em RMI toda esta logística é escondida, simplificando bastante o código.
- A transparência de rede é tratada pelo Java pois a JVM usa os seus processos para mascarar as heterogeneidades do sistema e trata da comunicação dos sistemas de forma transparente ao programador. Com isto concluímos que, como o acesso a recursos locais e remotos é feito da mesma forma, e que a localização precisa dos recursos não é conhecida pelo utilizador, a transparência de rede é estabelecida.

3- lamport

3. The schematics below depicts the *temporal evolution* of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm.

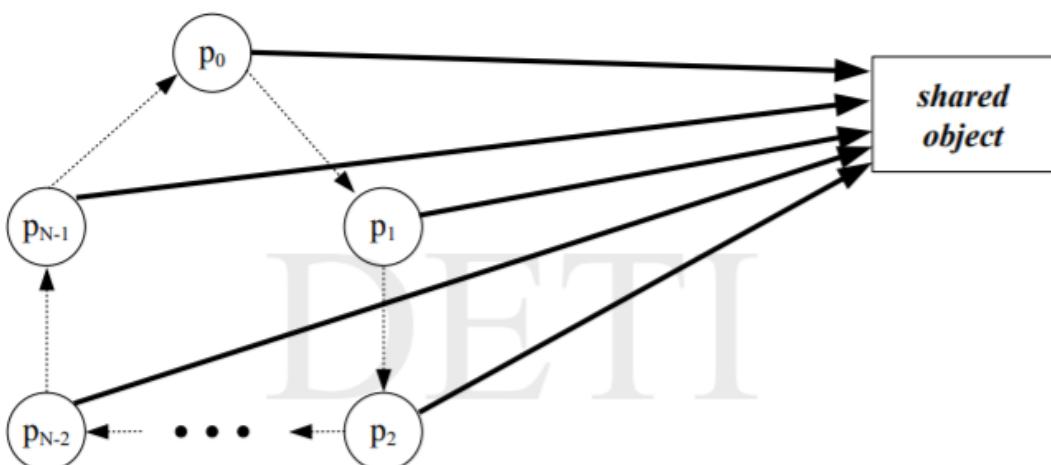


- Assign to the different events, specified by small letters ($a \dots l$), their associated time stamp.
- Take from the schematics three examples of pairs of concurrent events (\parallel) and list the longest sequence of sequential ones (\rightarrow).
- Explain why, taking examples from the schematics, the time stamp associated with each event can not be used to carry out the causal ordering of events. There is, however, a special case where this is possible. Which is it? Present clearly your claims.

- $a(1) b(4) c(5) d(6)$
 $e(1) f(2) g(3) h(4) i(7)$
 $j(2) k(3) l(4)$
- concurrentes : $e \parallel a, j \parallel f, k \parallel c$
 mais longa : $g \rightarrow b$ e $d \rightarrow i$: $g < i$

4- peer-to-peer

4. In *peer-to-peer communication*, a virtual ring based on the individual *id* was established for communication through message passing among the active processing nodes of a distributed system. Describe in detail the algorithms which allow the insertion and the retrieval of a processing node in the ring. Draw diagrams for each case to help clarify your description. Assume that there are neither message loss, nor node crashing.



Na comunicação peer-to-peer , os processos que implementam as atividades cooperam entre eles , não existindo distinção entre eles.

Sendo que é um anel , a troca de mensagens vai ser feita num ciclo fechado, e uma mensagem irá circular continuamente entre eles. O acesso a um objeto partilhado só pode ser feito pelo processo que tomou posse da mensagem. Após um nó tomar posse de um processo e ter terminado o acesso este envia um token para que o próximo processo do anel. Se o próximo processo não precisar de acesso, repassa o token para o processo seguinte.

2018 - recurso

1- Transparencia , 5 exemplos.

1. What is meant by *transparency* in the context of distributed systems? Give examples, at least five, of the modes transparency may assume. Justify clearly your claims.

igual a primeira de 2020 -recurso

2- access remote objets , server side. network transparency

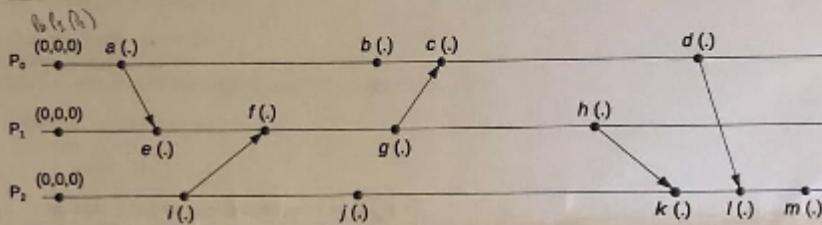
2. Take *access to remote objects* as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Draw a diagram that describes the distribution of the components of such a model, at the server side, by different computer systems, identifying each of them and explaining their role in the interaction. Assume that there are three remote objects positioned in geographically distant regions.

Refer in this context to what one means by *network transparency*. How can it be enforced in Java?

- Assumindo então que temos 3 objetos remotos em diferentes regiões , estamos perante a variante do tipo 3 , replicação de recursos.
- então ,o serviço é disponibilizado simultaneamente em vários sistemas, cada um executando uma variante do tipo 2(replicação do servidor -os processos do cliente são atendidos simultaneamente, o que significa que a thread base, mediante receber uma solicitação de conexão, instancia um agente proxy de serviço e inicia ouvindo de novo.). O recurso compartilhado é replicado em cada servidor, dando origem a múltiplas cópias. Este modelo visa maximizar a disponibilidade do serviço e minimizar o tempo de atendimento, mesmo em situações de pico de carga . O servidor é mantido operacional contra falhas de servidores específicos.
- transparência de rede - quando transparência de acesso (mesmas operações são feitas para aceder a recursos locais e remotos) e posição (quando um acesso de um recurso é feito sem conhecimento da sua localização) existem ao mesmo tempo
- Neste modelo cliente-servidor ambos os casos se verificam pois o recurso partilhado é replicado em cada servidor, dando origem a múltiplas cópias, o que é um caso de transparência de acesso. As solicitações dos clientes são distribuídas entre os servidores usando uma política de associação geográfica para pedidos globais e de associação rotativa para pedidos locais. Este último caso verifica transparência de posição, pois o acesso é feito sem se saber a localização exata dos servidores.

3 - lamport

3. The schematics below depicts the *temporal evolution* of three processes whose local clocks are vector logical clocks synchronized according to the vector algorithm.



$(V_{0,0}, V_{1,0}, V_{2,0})$

- Assign to the different events, specified by small letters ($a \dots m$), their associated time stamp.
- Take from the schematics three examples of pairs of concurrent events (\parallel) and list the longest sequence of sequential ones (\rightarrow).
- Identify all the events that belong to the *past* of event g ? Consider next all the events that belong to its *future*. Justify your reasoning clearly.

- $a(1,0,0)$ $b(2,0,0)$ $c(3,3,1)$ $d(4,3,1)$
 $e(1,1,0)$ $f(1,2,1)$ $g(1,3,1)$ $h(1,4,1)$
 $i(0,0,1)$ $j(0,0,2)$ $k(1,4,3)$ $l(4,3,4)$ $m(4,3,5)$
- concorrência: $i \parallel a, f \parallel b, b \parallel g$
mais longa : $a \rightarrow e \rightarrow g \rightarrow c \rightarrow d \rightarrow l \rightarrow a$

4- peer-to-peer ring

belong to its *future*. Justify your reasoning clearly.

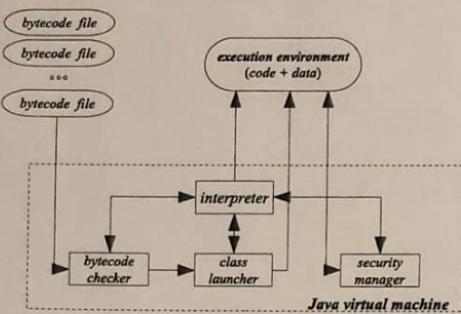
4. In *peer-to-peer communication*, a virtual ring based on the individual *id* was established for communication through message passing among the active processing nodes of a distributed system. Describe in detail the algorithms which allow the insertion and the retrieval of a processing node in the ring. Draw diagrams for each case to help clarify your description. Assume that there are neither message loss, nor node crashing.

igual a 4 de 2019 normal

2018 - normal

1-middleware , JVM

1. The diagram below describes the main components of the Java Virtual Machine (JVM).



Define *middleware* and explain why the Java Virtual Machine (JVM) can be considered part of the middleware. Present clearly your claims.

- Middleware pode ser descrito como um conjunto de processos que se comunicam entre si de forma a mascarar as heterogeneidades dos sistemas operativos de tal modo que o programa possa ser executado em todos eles.
- JVM constitui de uma camada de middleware que faz com as aplicações sejam totalmente independentes da plataforma hardware e do sistema operativo onde correm, implementando uma camada tripla de segurança (bytecode checker, class launcher, gestor de segurança)

2- message passim marshaling

2. Take *message passing* as the communication paradigm which was chosen to establish a client-server model among processes residing in different nodes of a distributed system. Draw a diagram that describes the functional interaction among the components of such a model, both at client and at the server side, identifying each of them and explaining their role in the interaction. Assume the server replication variant.

Refer in this context to what one means by *message marshaling* and *unmarshaling*. How are they enforced in Java?

2 - 2020 normal

3-

4- variante de replicação de recursos, 3 problemas

4. Suppose we want to ensure load balancing of the server tasks in a client-server model. In order to achieve this aim, a resource replication variant where the server is installed in two hardware platforms, is implemented. Both server instantiations are active at the same time and interact with the clients. Draw a functional diagram that depicts the organization and list three problems which must be solved for the system to work properly. Justify clearly your claims.

2020 normal

2017

1-

Parte A (12 valores)

1. Tanenbaum defines a *distributed system* as a *collection of independent computers that appears to its users as a single coherent system*. Using this definition as the starting point, try to elicit some of the distinctive features that this kind of systems present, namely *communication through message passing, failure handling and global internal state*.
2. Draw a functional diagram that depicts the organization and list three problems which must be solved for the system to work properly. Justify clearly your claims.

A definição convencional de sistema distribuído afirma que se está a lidar com um sistema operacional cujos componentes, localizados nos diferentes nós de processamento de um sistema de computador paralelo, comunicam e coordenam as suas ações através de message passing. Uma definição alternativa de sistema distribuído, embora essencialmente equivalente, é dizer que se trata de uma coleção de sistemas computacionais independentes percebidos pelos usuários como um sistema coerente (Tanenbaum).

Várias consequências surgem no seguimento disso:

- paralelismo, pois a existência de sistemas computacionais independentes dá origem a threads autónomas de execução;

- estado interno global, pois a necessidade de fornecer uma imagem de trabalho coerente requer alguma forma de coordenação de atividades entre os diferentes nós de processamento;
- comunicação através de message passing, pois o facto de nenhuma suposição ser feito de como os sistemas computacionais estão interligados, envolve um mecanismo minimalista baseado em message passing;
- escalabilidade, pois expandir o sistema através da integração de mais nós de processamento é um resultado imediato da organização prescrita;
- tratamento de falhas, pois qualquer um dos componentes do sistema distribuído pode falhar a qualquer momento, deixando os componentes restantes em operação.

2- publisher -subscriber e-commerce e email

2. Describe the *publisher-subscriber* model and explain how it fits in two application areas where it has become very popular.

email 1 - 2020 normal
ecommerce 1- 2019 normal

2016

1- middleware e diagrama

1. O que se entende por camada de *middleware*? Faça um diagrama ilustrativo que sinalize a presença da camada para o caso de uma máquina paralela de acoplamento solto, formada por três nós de processamento, que executa uma aplicação distribuída. Explique porque é que a *máquina virtual de Java (JVM)* pode ser considerada um componente do *middleware*.

2018 -normal resposta

