Sistemas Distribuídos
Teste 1

12 de Junho de 2023

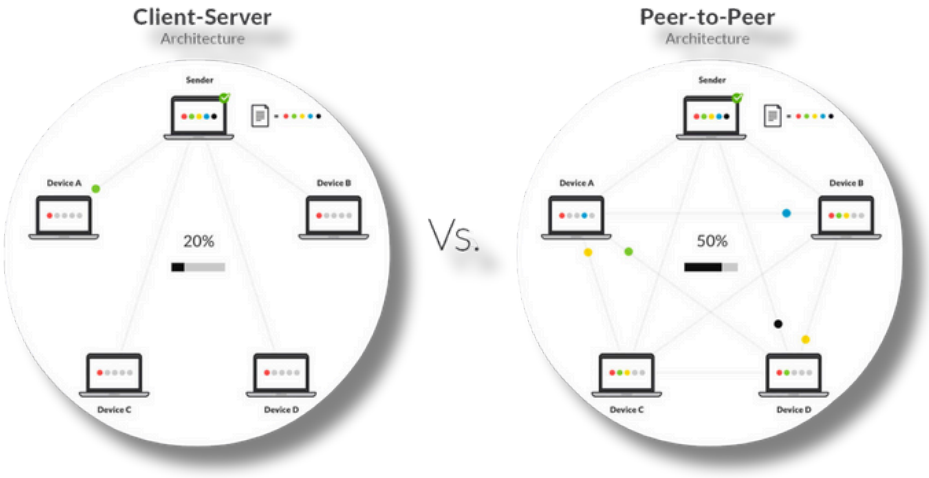
Part A (12 points)

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*. (***3 points***)

Neste tipo de sistemas podem ser enfrentadas diversas consequências, entre elas estão:

- **Escalabilidade:**
  - A capacidade de expansão do sistema a partir da integração de mais nós de processamento é uma consequência da organização interna.
- **Paralelismo:**
  - O facto de existirem sistemas computacionais que originam threads autónomas que executam simultaneamente.
- **Comunicação através de passagem de mensagens:**
  - Como, num sistema distribuído, o espaço de endereçamento não é partilhado, há a necessidade de desenvolver um mecanismo de interação entre os diferentes sistemas, que é a passagem de mensagens, que deverá ser algo simples.
- **Gestão de Falhas:**
  - Tendo em conta todos os componentes de um sistema distribuido, a gestão de falhas deverá ser uma preocupação pois qualquer um dos seus componentes poderá falhar a qualquer instante, sem que os restantes sejam prejudicados.
  - Ao implementar mecanismos de segurança contra falhas é possível contornar a capacidade que o sistema deverá possuir de lidar com falhas de modo a que estas sejam impercetíveis ao utilizador.
- **Estado Interno Global:**
  - Este estado centra-se nos estados atuais de todos os processos e mensagens a serem transmitidas entre os mesmos. É utilizado de forma a que a totalidade do sistema distribuido seja monitorizado.
  - O sistema deverá estar estruturado de modo a que seja possivel fornecer a qualquer momento uma imagem do trabalho corrente, o que requer uma coordenação de atividades entre os vários nós de processamento.

2. What distinguishes the *client-server* model from *peer-to-peer communication*? Give an example of each and explain how relevant they are in the examples you have presented. (***3 points***)

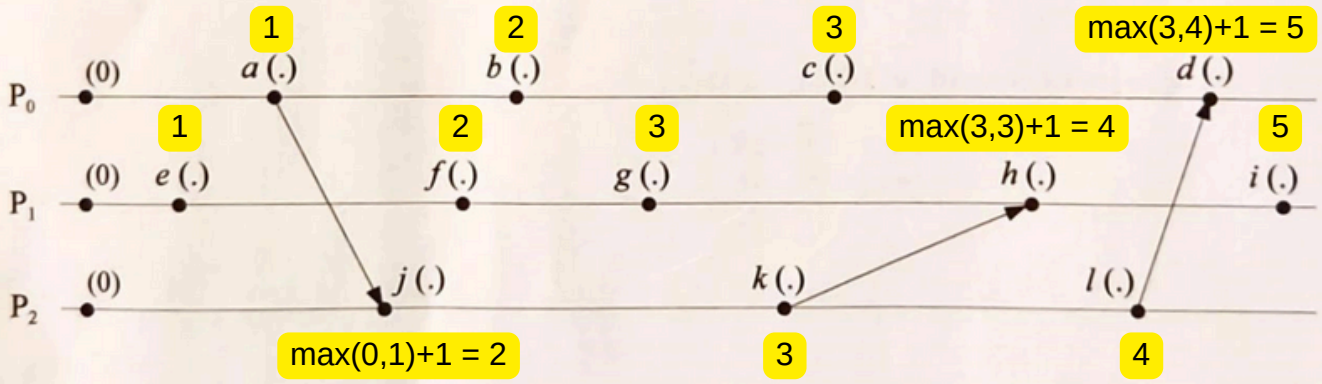


No modelo cliente-servidor, o sistema está organizado de forma hierárquica, onde o servidor fornece serviços e recursos aos clientes que os solicitam. Na comunicação peer-to-peer, os sistemas interagem diretamente entre si, sem a necessidade de um servidor central.

Um exemplo comum do modelo cliente-servidor é o sistema de uma loja online, onde os clientes (utilizadores que acedem à loja) solicitam páginas e fazem compras ao servidor (serviço de hospedagem da loja). O servidor processa os pedidos dos clientes e envia as respostas em seguida. Este modelo permite uma gestão centralizada dos recursos e da segurança, sendo adequado para aplicações onde a consistência e a autoridade central são críticas, como nas transações online.

Por outro lado, um exemplo frequente de utilização do sistema peer-to-peer é a partilha de ficheiros, onde cada participante (peer) pode atuar tanto como servidor quanto como cliente, partilhando e recebendo ficheiros diretamente uns dos outros. Este modelo é relevante em cenários onde a escalabilidade distribuída e a descentralização são valorizadas, como na partilha de recursos entre utilizadores na Internet.

3. The diagram depicts the *time* evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm. (***3 points***)



i. Assign to the different events, specified by small letters (*a* ... *l*), their associated time stamp.

ii. Give three examples of pairs of *concurrent* events and events *connected by a causality nexus* present in the schematics and justify why you have classified them in a such a way.

Tendo em conta que não existe qualquer ligação possível e->b, nem b->e, considero que o par (b-e) se trate de um par concorrente, sendo então b || e.

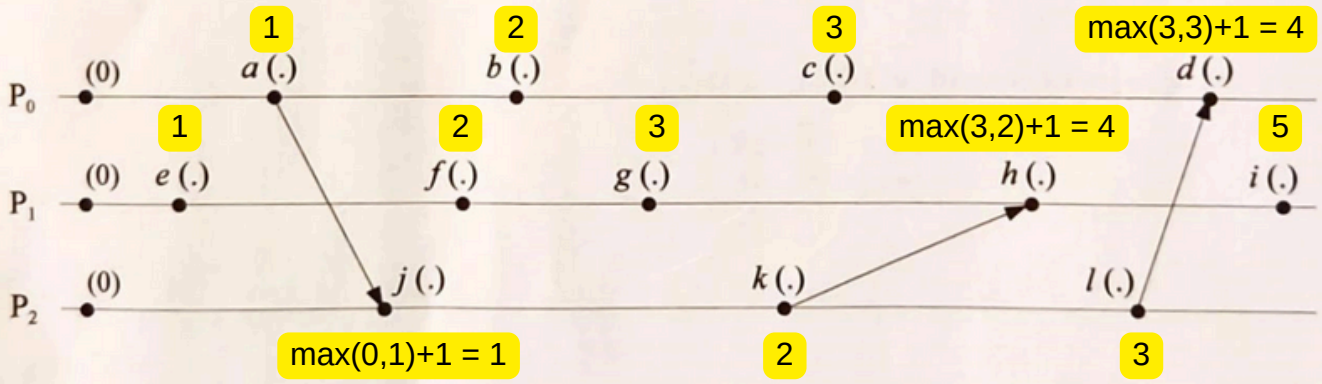
Visto que não se verifica nenhuma ligação f->c, nem c->f, podemos então concluir que (f-c) se trata de um par de eventos concorrentes podendo ser representados por f || c.

Como é ainda possível observar, não existe conexão sequencial entre o par (g-k), tendo em conta que o timestamp de g>k, então a ligação k->g é impossivel, e visto que não se verifica nenhuma ligação g->k, então o par é concorrente.

Por outro lado, seguindo as conexões:

a->j , podemos concluir que o par (a-j) é sequencial
a->j->k, podemos concluir que o par (a-k) é sequencial
a->j->k->h, podemos concluir que o par (a-h) é sequencial
a->j->k->h->i, podemos concluir que o par (a-i) é sequencial

3. The diagram depicts the *time* evolution of three processes whose local clocks are scalar logical clocks synchronized according to the Lamport algorithm. (***3 points***)



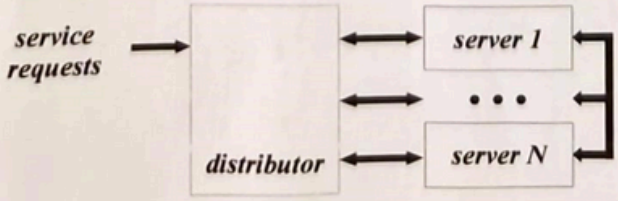
i. Assign to the different events, specified by small letters (*a* ... *l*), their associated time stamp.

ii. Give three examples of pairs of *concurrent* events and events *connected by a causality nexus* present in the schematics and justify why you have classified them in a such a way.

iii. What is the difference between *total* and *partial* ordering of a group of events? Justify clearly your claim.

Enquanto que, com uma ordem parcial, é possível ordenar todos os eventos dentro de um dado processo, não é possível ordenar todos os eventos que ocorrem entre processos concorrentes. Numa ordem total, além do timestamp resultante da aplicação do algoritmo de Lamport, é também tido em conta o ID do processo a que o evento em causa pertence. Isso permite que, em caso de empate no que diz respeito a dois eventos distintos e à sua respetiva timestamp, seja consultado o ID do processo a que cada um pertence, sendo assim possível desempatar e ordenar todos os eventos de todos os processos concorrentes existentes no sistema em análise.

4. The schematics below depicts a simple solution for *load balancing* in a client-server model.



The role of the *distributor* is to forward service requests to any of the *servers*. The decision on which *server* a specific *service request* is forward to, is based on the *working load* each is presently enduring as perceived by the *distributor*. Bear also in mind that the copies of the shared region present in the *servers* must be continuously synchronized.
Describe in general terms how the whole system should operate. What kind of criteria should the distributor use to estimate the *working load* of a server? What kind of protocol should the *distributor* implement to assess if a given server is operating properly, or has failed? Present clearly your reasoning. (***3 points***)

Neste modelo de cliente-servidor, onde foi implementada a abordagem Server Replication, o distribuidor desempenha um papel central tanto na gestão como na distribuição das solicitações. A carga de trabalho (working load) de um servidor pode ser avaliada de múltiplas formas, como através de monitorização ativa com métricas como o número de conexões ativas, o tempo médio de resposta, ou através de health checks que verificam periodicamente se o servidor em causa está responsivo e ativo, utilizando pings, por exemplo. Podem ainda ser utilizados protocolos de comunicação como HTTP ou HTTPS para esta verificação de métricas ou health checks através de endpoints específicos em cada servidor.

Quando um distribuidor recebe um pedido de um serviço, este tem de decidir para qual servidor irá enviar essa solicitação. Esta decisão é tomada com base em algoritmos de Load-Balancing que garantem que não existem servidores sobrecarregados nem subcarregados. Alguns dos algoritmos baseiam-se em informações obtidas durante a realização dos health checks e monitorizações anteriormente referidas. Itens úteis para a decisão incluem: número de conexões do servidor e o tempo de resposta do servidor.

Caso ocorra algum tipo de falha num servidor, esta será notada pelo distribuidor através dos seus health checks periódicos, que constituem mecanismos de recuperação de falhas, como o Heartbeat Protocol, que envia periodicamente heartbeats para verificar se todos os servidores estão disponíveis.

No caso de ser solicitado um serviço de um servidor indisponível no momento, devem existir mecanismos de failover implementados que permitam que o distribuidor redirecione automaticamente estas solicitações para outro servidor operacional. Isto pode ser executado utilizando uma lista de servidores de reserva (standby) ou realocando a carga entre os servidores restantes.