1.

C - FACTO

2.

C - Uma vez que o enunciado fala de IP multicast, que não é reliable, este está disponível apenas via UDP que não oferece garantias de fiabilidade.

3.

D - FACTO!!!!!!!!!!

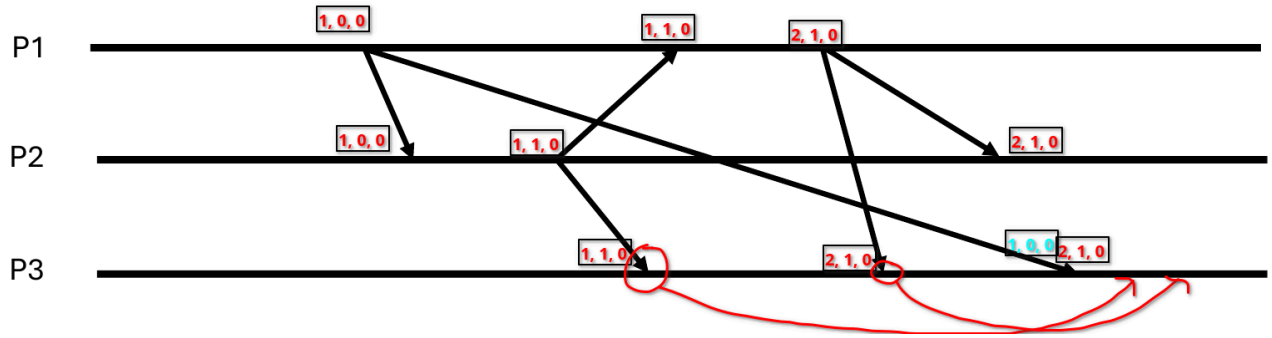
4.

A - Juro

5.

1. $a \rightarrow b$ porque b acontece após a no mesmo processo
2. $b \rightarrow c$ porque c é uma receção da imagem enviada em b

6.



7.

- Assumimos que A e B ambos possuem $K_{A_{\text{pub}}}$ e $K_{B_{\text{pub}}}$, bem como as respetivas chaves privadas.
- Queremos garantir:
  - Confidencialidade (apenas o B pode ler D)
  - Integridade (B pode ver que D não foi alterado e que veio de A)

- Confirmação da receção e acesso (A deve receber uma prova de que B recebeu, desencriptou e leu D, com não-repúdio da receção por parte de B)

1. $A \rightarrow B : \text{TransactionID}, \{D\}^{K_s}, \{K_s\}^{K_{B_{\text{pub}}}}, \{H(D)\}^{K_{A_{\text{priv}}}}$
   - Gera um hash de D, $H(D)$ e assina com a sua chave privada $K_{A_{\text{priv}}}$
   - Gera uma chave privada da sessão K_s e assina com a chave pública de B, $K_{B_{\text{pub}}}$
   - Inclui TransactionID para identificar unicamente esta ligação
2. $B \rightarrow A : \text{TransactionID}, \{H(D')\}^{K_{B_{priv}}}$
   - Gera hash do documento recebido D', $H(D')$
   - Verifica se é igual ao $H(D)$ recebido
3. Resta agora a A verificar a confirmação de B.
   - Se a assinatura de B for válida, usando $K_{B_{\text{pub}}}$, consegue extrair $H(D')$ e comparar com o seu original.

8.

Mover a responsabilidade da autenticação para o provedor de identidade, permitindo que as aplicações de terceiros operem com "tokens" de acesso limitados, em vez de senhas completas.
Bué mais seguro, conveniente e rápido, olha aí:

- **Segurança:**
  - Não partilha senhas com terceiros
  - O user pode controlar o acesso de maneira mais refinada, o gajo pode selecionar permissões parciais
  - Podemos revogar o acesso a qualquer momento
  - Os consumidores de OAuth não podem dar lock-out do user no serviço principal
- **Melhoria da experiência do user**:
  - Não precisa de inserir credenciais todas as vezes, feito parvo
  - Rápido graças a redirecionamentos HTTP
  - Não repúdio: se a Alice quiser publicar uma avaliação de um livro no GoodReads e partilhar essa avaliação no Facebook, pode fazê-lo sem partilhar diretamente as suas credenciais do Facebook com o GoodReads

9.

- **Versão inicial**:
  - Não havia necessidade explícita de os nós publicarem ou anunciarem os seus ficheiros de forma prévia.

        - A descoberta do conteúdo baseava-se em query flooding, ou seja, quando um node queria procurar um ficheiro, ele enviava a sua consulta aos vizinhos diretos, que, por sua vez, reencaminhavam a consulta para os seus próprios vizinhos, e assim sucessivamente, até que o ficheiro fosse encontrado.
        - Não havia índice centralizado ou distribuído que os nodes preenchessem ativamente com os seus ficheiros disponíveis
- **Introdução de ultrapeers**:
    - Mudou para modelo híbrido, exemplificado pela arquitetura do KaZaA, que é um overlay não estruturado com super-nodes. Neste novo modelo:
        - Quando um novo node se junta à rede, ele contacta um super-node e passa a publicar os ficheiros que possuía, enviando info sobre esses ficheiros para o super-node associado.
        - Para um node procurar a localização de um ficheiro, ele enviava a sua query ao super-node ao qual estava associado.
        - As pesquisas na rede passavam a ser tratadas por uma camada de super-nodes que realizavam o query flooding entre si, mas de forma eficiente e controlada do que no modelo anterior de flooding por todos os nodes.
- Foi necessária para resolver um problema de escalabilidade, o query flooding original levava a congestionamento maluco da rede com mensagens de query.

10.

- **Desacoplamento espacial**:
    - O emissor (publisher) não tem de conhecer a identidade ou localização do(s) recetor(es) (subscriber(s)), e vice-versa.
    - Usam um serviço de eventos como intermediário.
    - Os publishers publicam eventos para esse serviço de eventos, sem especificar quais subscribers os receberão.
    - Os subscribers expressam interesse em eventos específicos através de subscrições.
    - O serviço de eventos tem a tarefa de garantir a entrega das notificações aos subscribers.
- **Desacoplamento temporal**:
    - O emissor não precisa de existir ao mesmo tempo que os recetores para comunicarem.
    - As notificações de eventos são enviadas assincronamente pelos publishers, portanto pode mandar eventos quando não há ninguém ativo.

- O serviço de eventos armazena esses eventos até que os subscribers interessados estejam disponíveis e possam recebê-los.