Modelos de Falha e Semânticas em Sistemas de RPCs

Sistemas Distribuídos 2013/2014

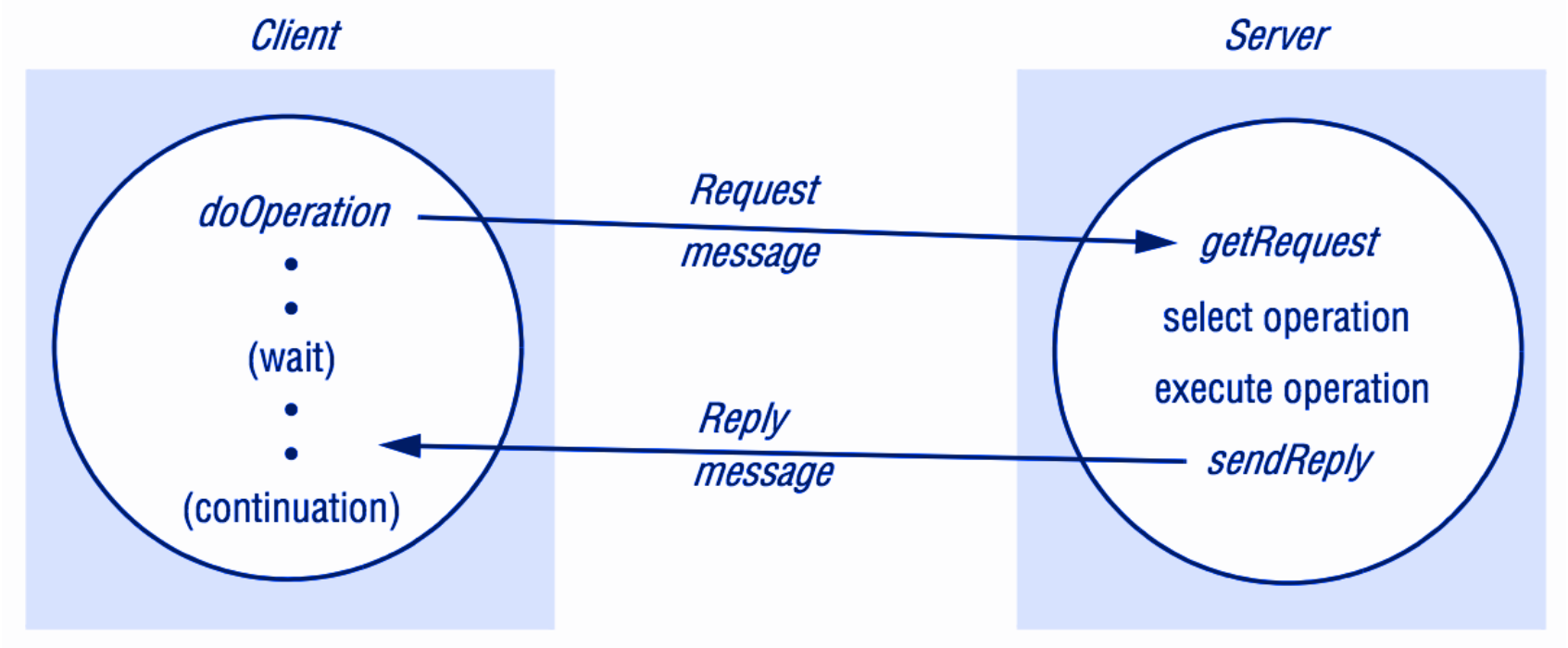
Classes de avarias

Classe de avarias	Afeta	Descrição
Fail-stop	Processo	O processo pára e permanece parado. Outros processos podem detetar este estado.
Crash	Processo	O processo pára e permanence parado. Outros processos poderão não detetar este estado.
Omission	Canal	Uma mensagem enviada nunca chega ao destinatário.
Send-omission	Processo	Um processo executa uma operação de envio, mas a mensagem nunca chega ao buffer de saída.
Receive-omission	Processo	Uma mensagem é colocada no buffer de receção de um processo, mas este nunca a recebe.
Arbitrary (Byzantine)	Processo ou canal	O processo ou o canal exibe um comportamento arbitrário: pode enviar mensagens arbitrárias em momentos arbitrários; pode omitir mensagens; pode parar ou executar operações incorretas.

Protocolos request-reply

- Interações típicas cliente-servidor.
- Muitas implementações usam TCP.
- Existem implementações sobre UDP, que eliminam overheads desnecessários do TCP:
  - Os acknowledgements são redundantes, dado que os requests são seguidos de replies.
  - Evita-se o estabelecimento de uma ligação.
  - O controlo de fluxo é redundante (a maior parte das invocações envolvem poucos argumentos e resultados).

Comunicação request-reply



Operações do protocolo request-reply

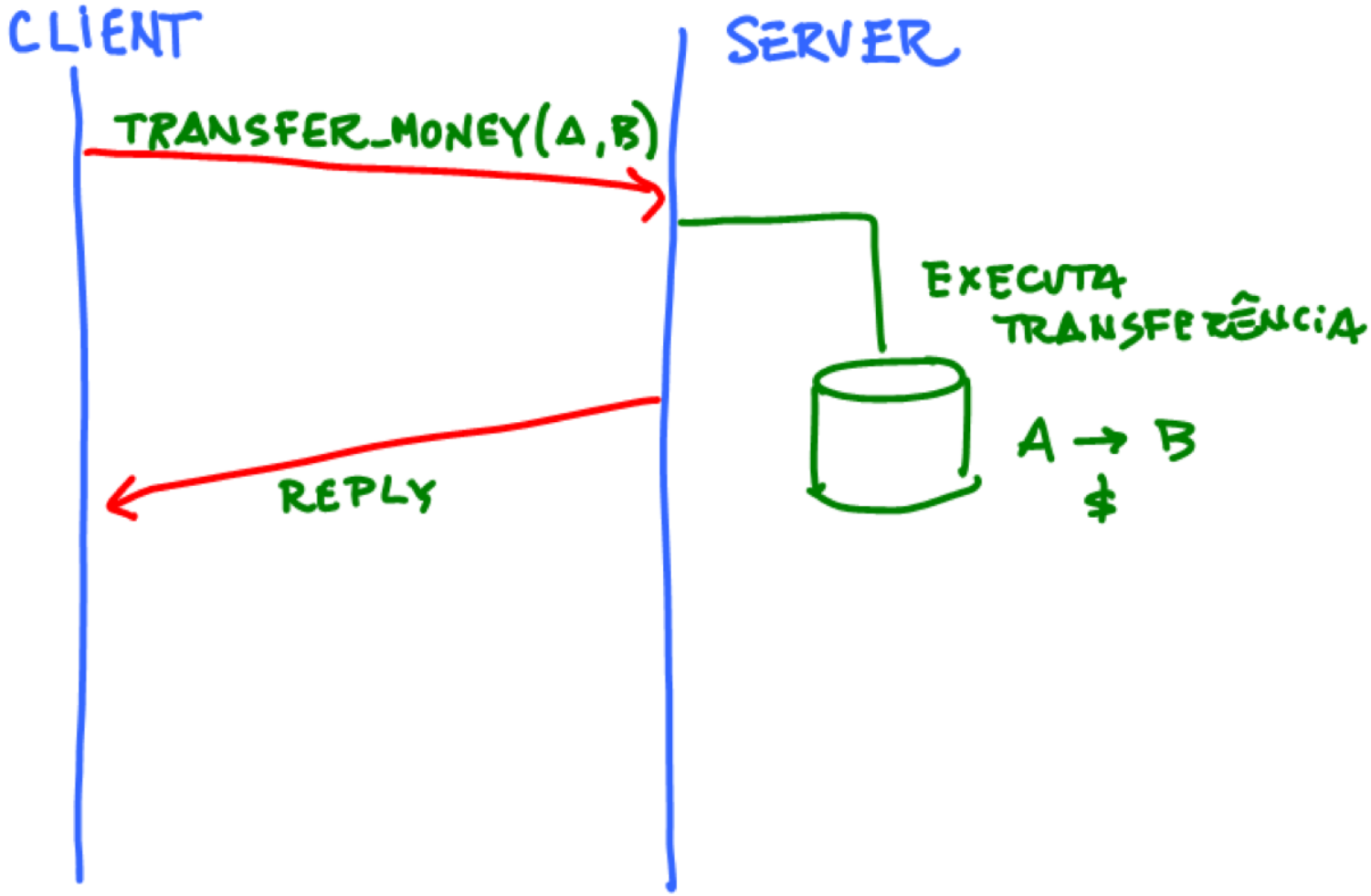
- `public byte[] doOperation(RemoteRef s, int operationId, byte[] arguments)`
  Envia um pedido ao servidor remoto e devolve a resposta.
- `public byte[] getRequest();`
  Obtém o pedido de um cliente através do porto do servidor.
- `public void sendReply(byte[] reply, InetAddress clientHost, int clientPort);`
  Envia a resposta ao cliente (para o seu endereço e porto).

Idempotência

Operações idempotentes Uma operação idempotente pode ser executada repetidamente, sendo o efeito igual a executar apenas uma vez.

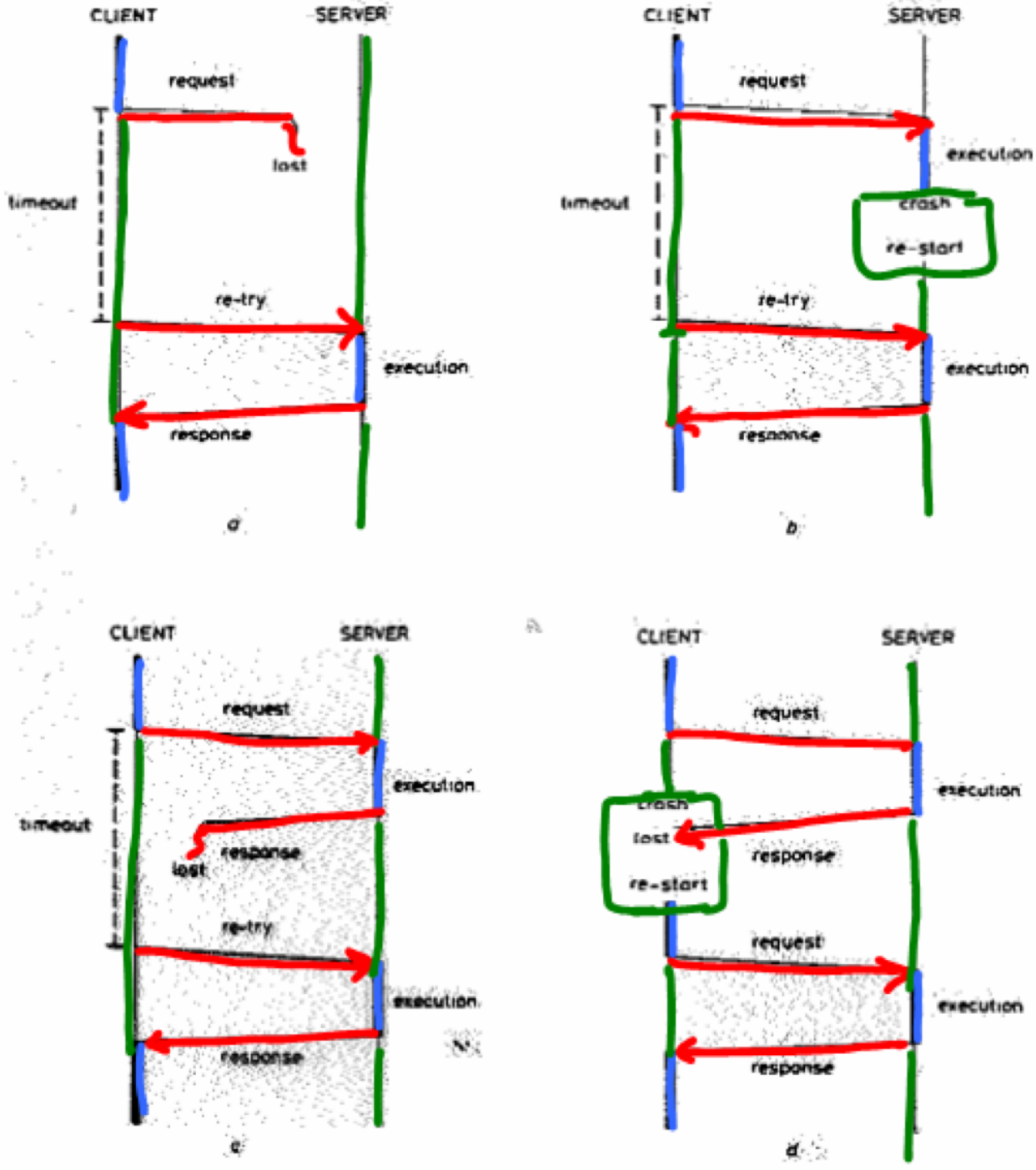
Operações não-idempotentes São operações cujo resultado final depende do número de vezes que forem aplicadas.

Exemplo de operação não-idempotente



Cenários de falha

- A mensagem com o pedido pode-se perder.
- A mensagem com a resposta pode-se perder.
- O servidor está muito lento e aparentemente não responde.
- O servidor vai abaixo e não responde.
- O servidor vai abaixo e volta acima.
- O cliente vai abaixo e volta acima.

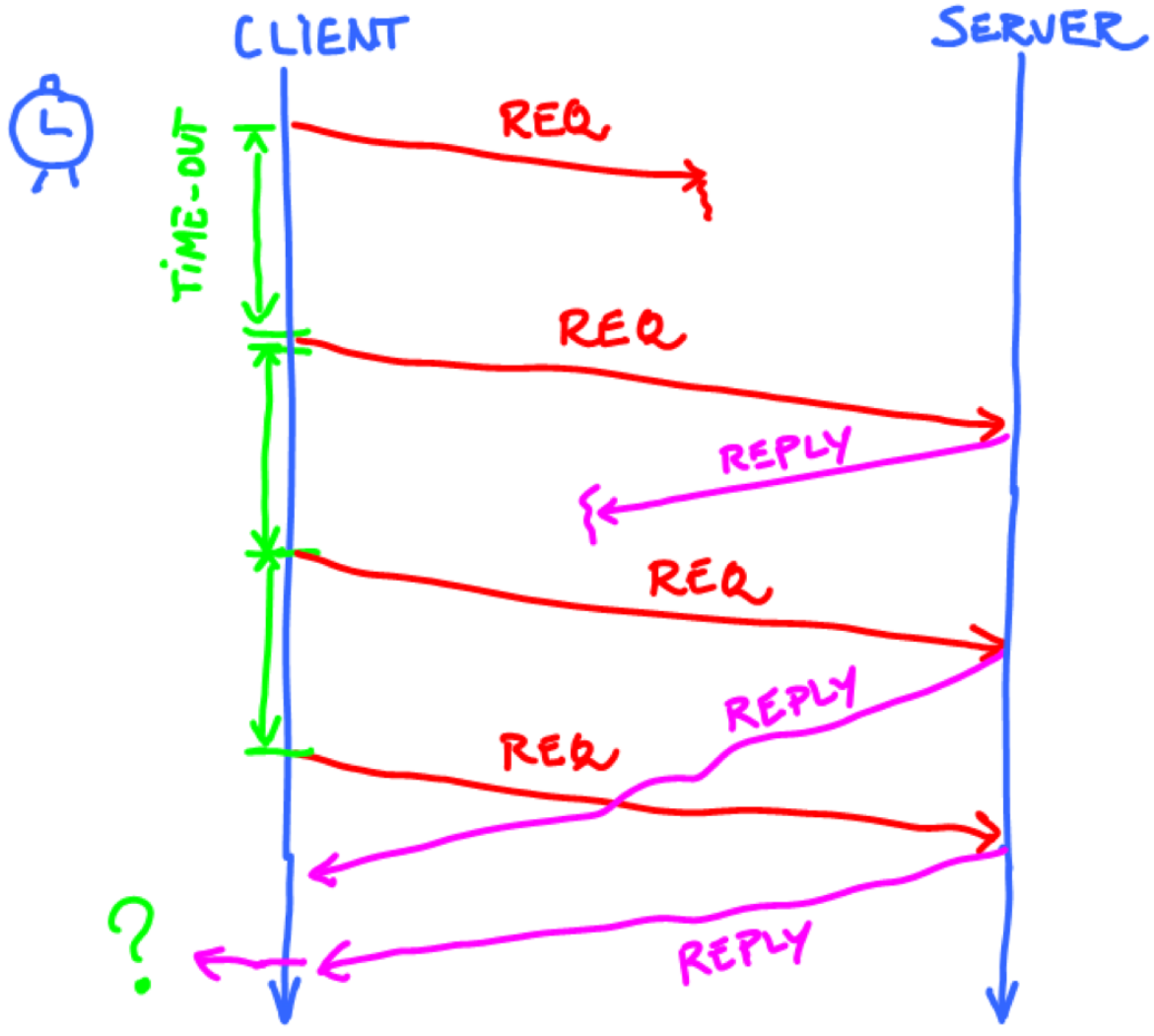

CLIENT
SERVER
request
lost
timeout
re-try
execution
response
a
CLIENT
SERVER
request
execution
timeout
crash
re-start
re-try
execution
response
b
CLIENT
SERVER
request
execution
timeout
response
lost
re-try
execution
response
c
CLIENT
SERVER
request
execution
crash
lost
response
re-start
request
execution
response
d

Medidas de recuperação

Em caso de timeout...

- Retorna-se imediatamente da operação (pouco comum).
- Repete-se o pedido...
  - O pedido anterior pode ter chegado.
    - Se as operações forem idempotentes, não há problema.
    - Se as operações forem não-idempotentes, há que re-enviar a *mesma* resposta.

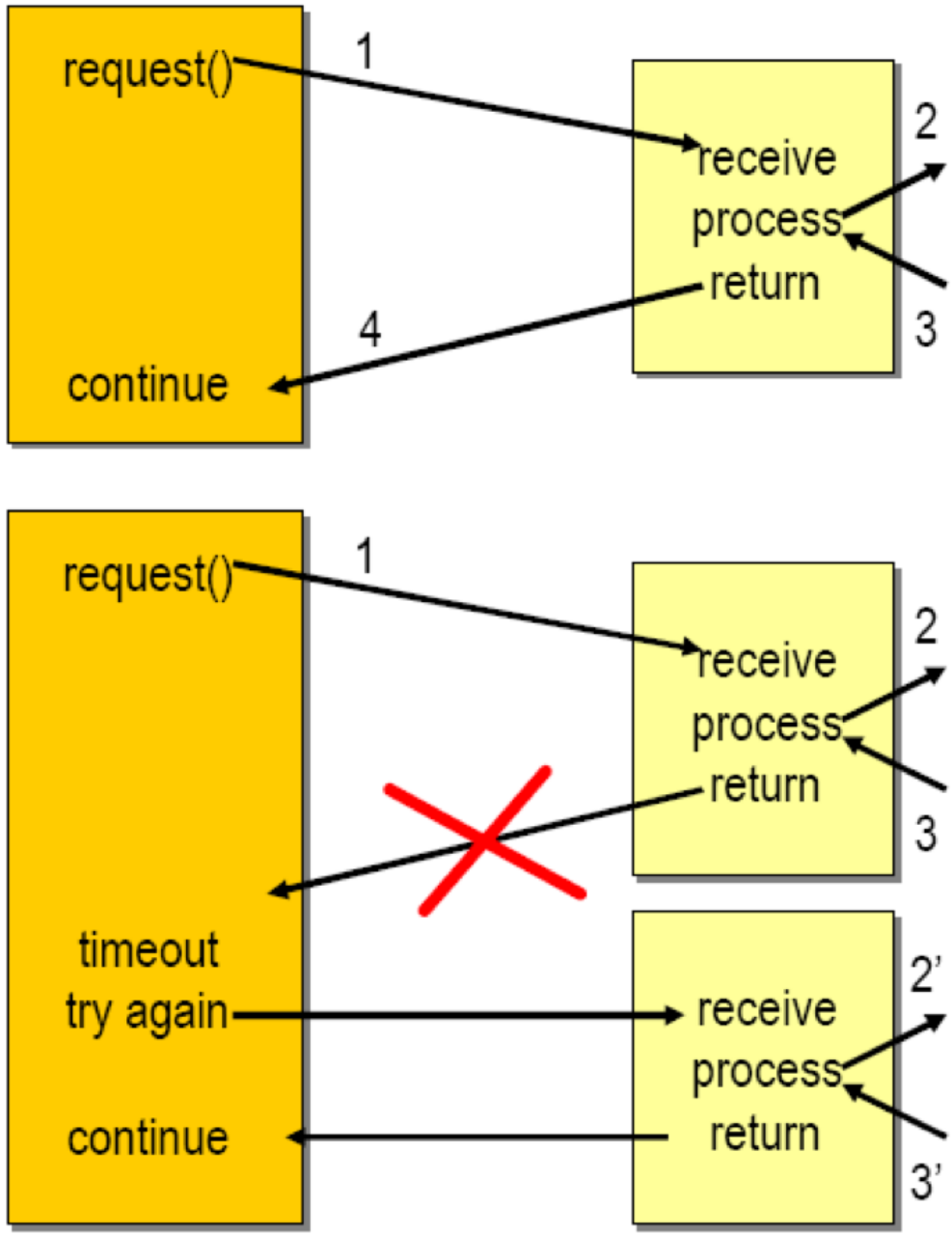
Re-transmissões



Em caso de falha, o que se passou?

- A operação pode nunca ter sido executada, pode ter sido executada uma vez, ou pode ter sido executada mais do que uma vez.
- É necessário filtrar os duplicados.

Falhas do lado do servidor



Semânticas dos RPCs

Maybe O procedimento pode ou não ter sido executado; usa-se quando a aplicação não requer garantias.

At-least-once Se houver retorno, o procedimento foi executado uma ou mais vezes; caso contrário, o procedimento foi executado zero ou mais vezes. Acontece quando o protocolo executa re-transmissões mas não suporta filtragem de duplicados.

At-most-once Se o cliente receber um resultado, o procedimento foi executado uma vez; caso contrário, o procedimento foi executado zero ou uma vezes. Semântica característica de um protocolo de invocação remota que suporta a filtragem de duplicados e o re-envio de respostas a procedimentos anteriormente executados.

Exactly-once O procedimento é executado uma só vez.

Invocation semantics

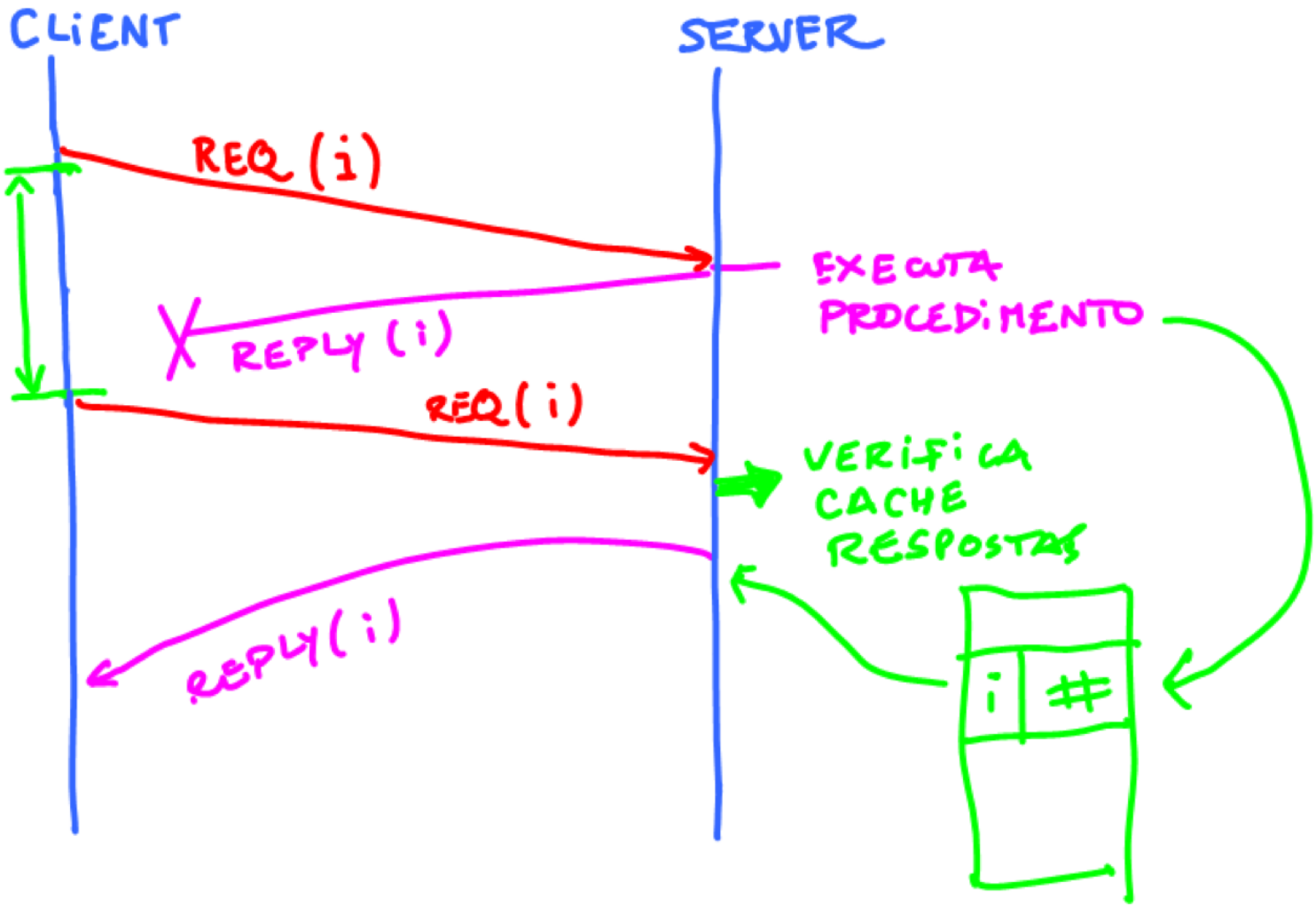
Fault tolerance measures			Invocation semantics
Retransmit request	Filter duplicates	Re-execute procedure or retransmit reply	
No	Not applicable	Not applicable	Maybe
Yes	No	Re-execute procedure	At-least-once
Yes	Yes	Retransmit reply	At-most-once

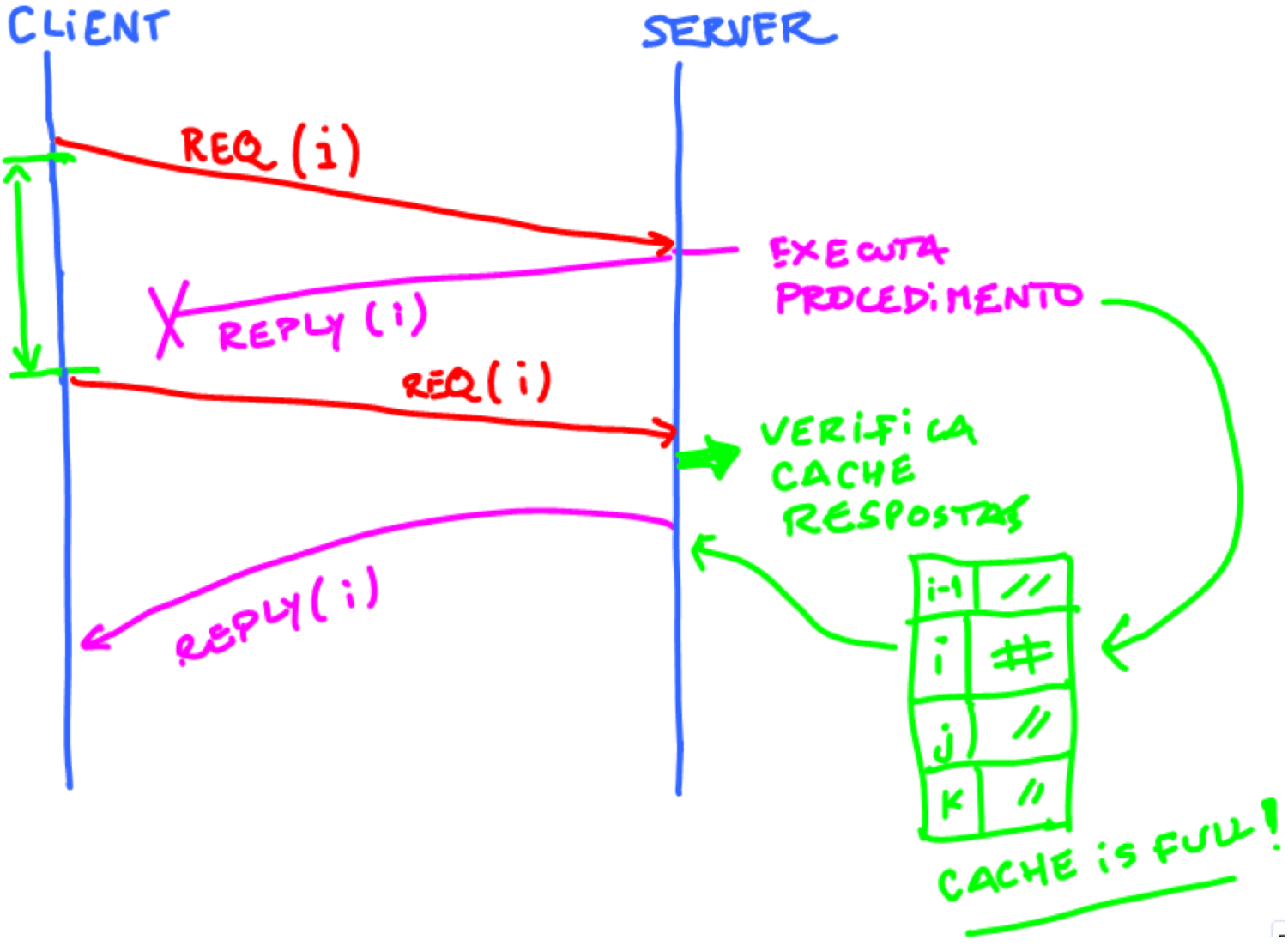
At-least-once *vs.* At-most-once

- As operações idempotentes podem ser usadas com um protocolo de invocação remota com a semântica at-least-once.
- As operações não-idempotentes devem ser usadas com um sistema de RPCs que assegure uma semântica at-most-once.

Implementação sobre UDP

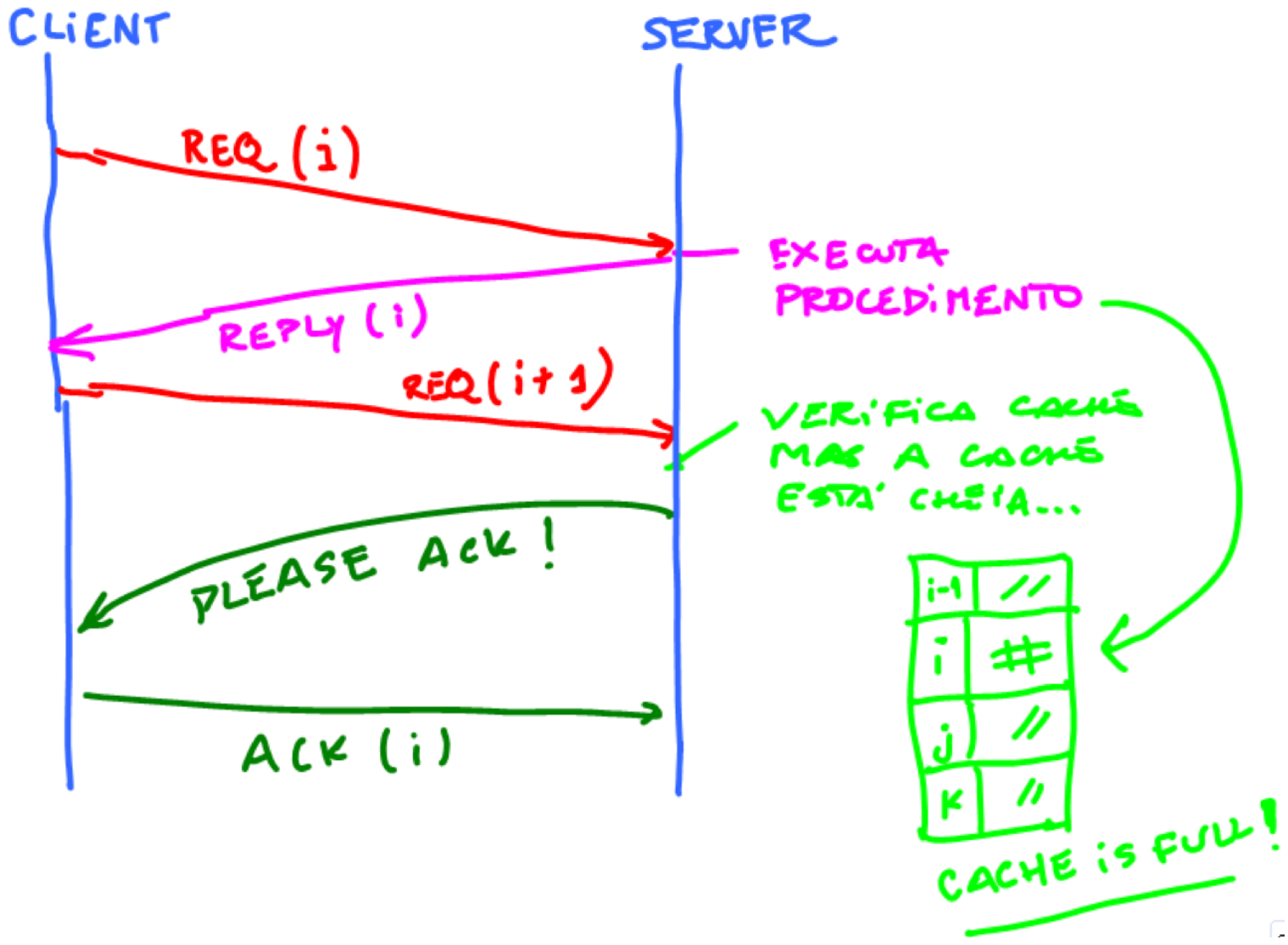
- Número de vezes que uma mensagem é enviada = N.
- Se N = 1 não se mascaram falhas de omissão (ou timeouts mal regulados)
- Se N > 1 e não se filtrar os duplicados a semântica é "At-least-once".
- É possível com UDP implementar a semântica "At-most-once" com filtragem de duplicados e re-envio de respostas anteriores.
- Vamos ver como...

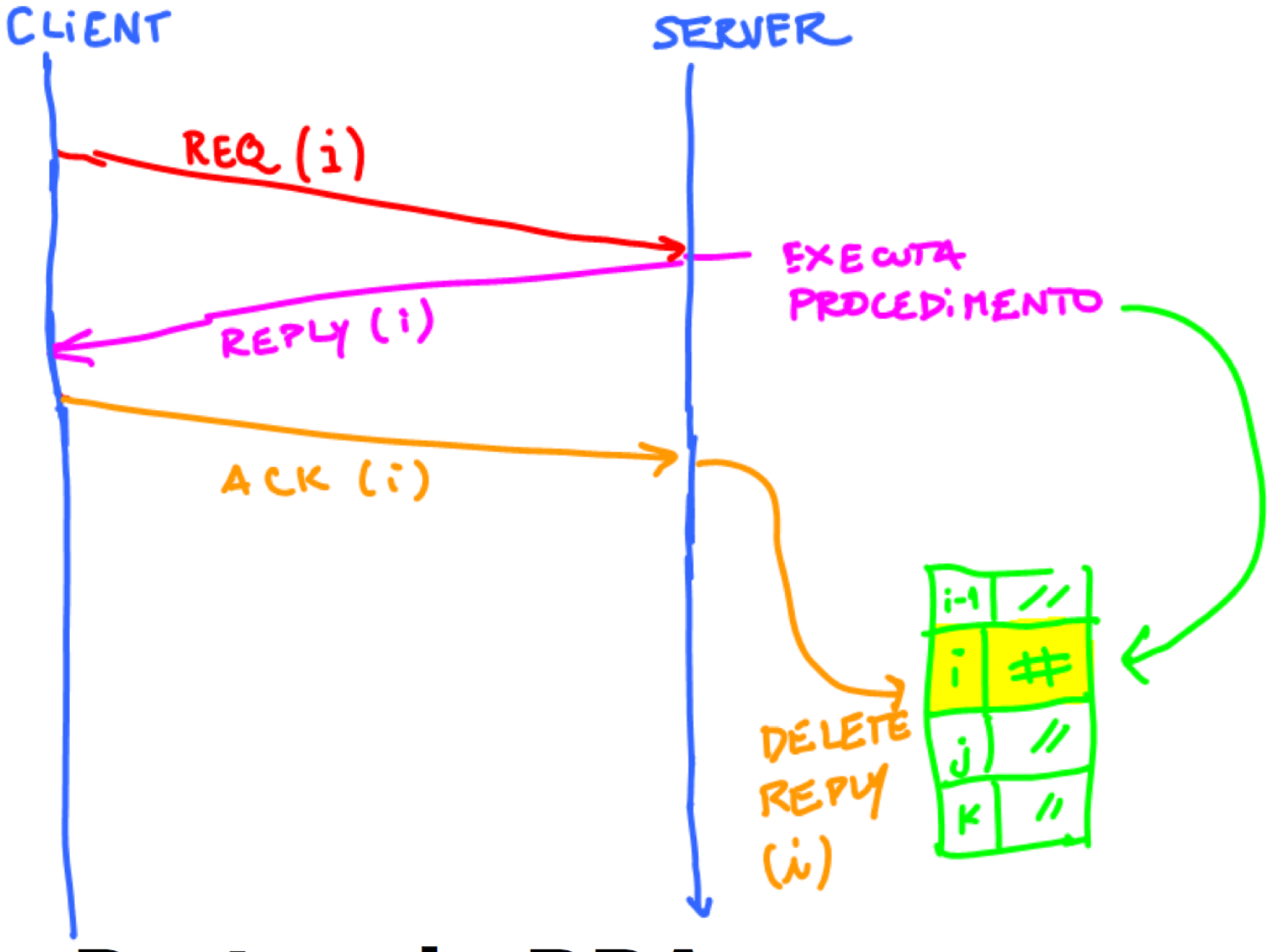

CLIENT
SERVER
REQ (i)
EXECUTA PROCEDIMENTO
REPLY (i)
REQ (i)
VERIFICA CACHE RESPOSTAS
REPLY (i)
i
#


CLIENT
SERVER
REQ (i)
EXECUTA PROCEDIMENTO
REPLY (i)
REQ (i)
VERIFICA CACHE RESPOSTAS
REPLY (i)
i-1
i
j
K
CACHE is FULL!

Como limpar a cache?

- Se o servidor não limpar a cache pode ficar sem recursos de memória.
- Soluções:
  - Garbage-Collection (on-demand)
  - Modelo RRA (Request-Reply-Acknowledge)
  - Piggyback info nas mensagens


CLIENT
SERVER
REQ (i)
EXECUTA PROCEDIMENTO
REPLY (i)
REQ (i+1)
VERIFICA CACHE MAS A CACHE ESTÁ CHEIA...
PLEASE ACK !
i-1
i
j
k
ACK (i)
CACHE IS FULL !



Protocolo RRA

RPC exchange protocols

Name	*Messages sent by*		
	Client	*Server*	*Client*
R	*Request*		
RR	*Request*	*Reply*	
RRA	*Request*	*Reply*	*Acknowledge reply*

Gestão da cache

- Importante: Evitar que a cache cresca e consuma os recursos de memória.
- Protocolo RRA: o que fazer quando se perde o ACK?...
- É preciso implementar uma solução para prever os crashes do lado do servidor: *a cache não pode ser volátil*.
  - Persistent queuing.
- Ter em atenção também às falhas do lado do cliente...
- Número de identificação deve ser acrescido de um número de "incarnação" do cliente.

Protocolos de RPCs

Com transporte connection-oriented (TCP):

- Request-reply protocol (RR) – facilita a implementação da política "At-most-once".

Com transporte connectionless (UDP):

- Request protocol (R) – permite implementar a semântica "Maybe".
- Request-reply protocol (RR) – permite implementar facilmente "At-least-once".
- Request-reply-acknowledge protocol (RRA) – usado para melhorar RR no caso de de se estar a implementar a semântica "At-most-once".

Soluções adotadas

- Java RMI e Corba suportam apenas a semântica "at-most-once" tipicamente implementada sobre ligações TCP.
- Corba permite também a utilização da semântica "maybe".
- SUN RPC permite utilizar as semânticas "maybe", "at-most-once" e "at-least-once" quer sobre UDP quer sobre TCP. O programador pode também manipular os valores dos timeouts e o número de repetições.