

**Trabalho Prático N°2 – Desenho e implementação de um jogo distribuído na Internet**

Duração: 6 aulas (não consecutivas, ver calendário)

**Motivação**

Em aplicações distribuídas na Internet com partilha de dados a partir dum ou várias fontes, em que o número de programas clientes dos utilizadores da aplicação é muito maior que o número de servidores disponível, é normal utilizarem-se técnicas/tecnologias que permitam a poupança da largura de banda. Por exemplo, na distribuição de emissões radiofónicas e de televisão digitais, em que os conteúdos são distribuídos a tecnologia *Multicast* é a mais utilizada. Um dos requisitos é a de que os conteúdos sejam distribuídos em árvore, desde a raiz (servidor principal) até todas as folhas (clientes), ou a grupos de folhas. No entanto, quando não existe uma hierarquia definida nos servidores de conteúdos e os conteúdos podem ser distribuídos a partir de qualquer um dos servidores, a tecnologia *multicast* já não é tão eficiente e a gestão do encaminhamento do tráfego pode ser muito complexa. Além disso, o paradigma *multicast* não suporta conexões TCP. Nesses casos, mecanismos mais simples que integrem técnicas de *broadcast* e algumas das características do *multicast* podem ser utilizados com melhores resultados.

Neste trabalho pretende-se implementar um pequeno protótipo dum jogo em tempo real que se baseia num serviço de distribuição de conteúdos com posterior interação por parte dos utilizadores e que tente otimizar a utilização da largura de banda. O sistema distribuído deve implementar comunicações TCP entre servidores (não existe hierarquia entre servidores) e comunicações UDP entre clientes e servidores.

O objetivo do jogo é dois ou mais utilizadores jogarem numa sessão/desafio de perguntas com escolha de resposta múltipla. O vencedor desse desafio é o utilizador que acertar mais perguntas (em caso de empate no número de respostas certas desempata-se pelo tempo total que os utilizadores demoraram a responder às questões certas). No fim de cada jogo, a pontuação obtida é integrada num *score* global do utilizador. Os utilizadores são ordenados num *ranking* a partir do seu *score* global.

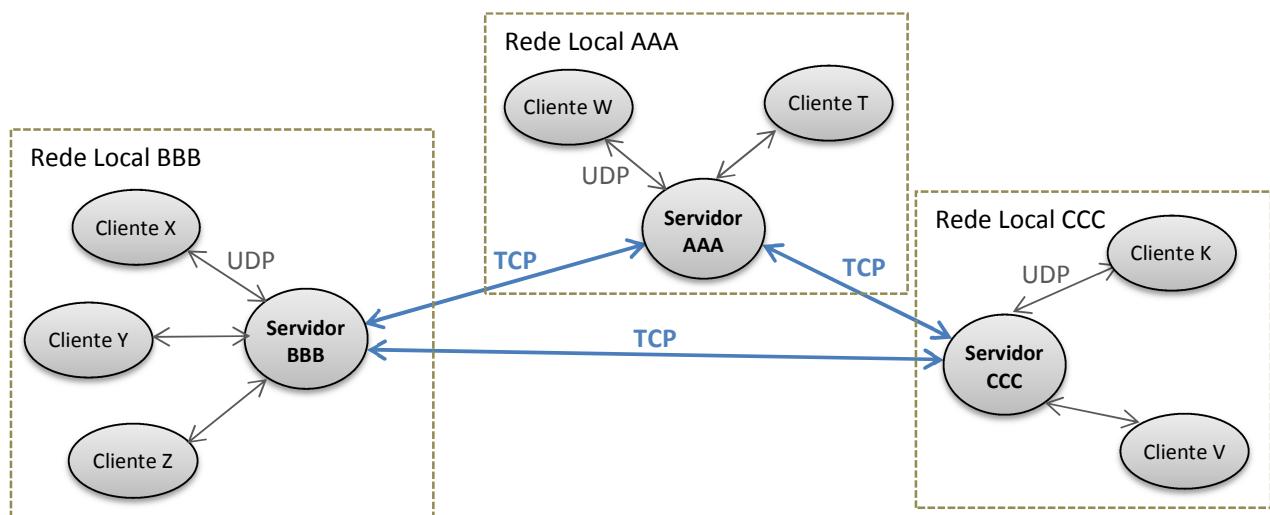
**Objectivo Geral**

Planeamento e implementação dum serviço de distribuição de conteúdos que tente optimizar a utilização da largura de banda por transmissão seletiva, fiável e com controlo de fluxo e de erros, dos conteúdos entre servidores usando *unicast TCP* (apenas para grupos de servidores que sirvam utilizadores que estão a jogar num determinado momento) e transmissão por *unicast* e *broadcast* UDP para todos os clientes da rede local IP (onde não costuma haver restrições relevantes na largura de banda). As comunicações do projeto têm que ser todas implementadas à custa de programação em *sockets TCP* e *UDP* e não deve utilizada nenhuma tecnologia baseada em *http* ou implementados interfaces baseados em serviços web.

**Descrição**

Em cada rede local onde o jogo irá estar disponível aos utilizadores (Ver Figura 1) deve ser instalado um servidor numa porta fixa, pré-definida, para que os clientes (programas que os clientes utilizam para aceder ao jogo) possam contactar o respetivo servidor (cliente implementa através de pedido **HELLO** por *broadcast*). O utilizador deve então poder entrar no sistema para jogar se já estiver registado (cliente implementa através de pedido **LOGIN**) ou registar-se (através de pedido **REGISTER**).

Depois de estar no sistema (sessão) o utilizador pode: ver/listar o seu ou os *rankings* de todos os utilizadores (através do pedido **LIST\_RANKING**), ver/listar os desafios disponíveis no sistemas (através do pedido **LIST\_CHALLENGES**), lançar um desafio (através do pedido **MAKE\_CHALLENGE**) e aceitar um desafio (através dum pedido **ACCEPT\_CHALLENGE**).



Depois de aceitar um desafio, assim que o jogo começar, o utilizador responderá a cada uma das perguntas pré-definidas dentro dum prazo limite (através do pedido **ANSWER**) ou desistir (através do pedido **QUIT**). No final do jogo, o utilizador é informado do *score* de todos os intervenientes (através do pedido **END**). O utilizador pode sair do sistema e abandonar a sessão a qualquer altura (através do pedido **LOGOUT**).

Cada desafio consiste numa série de dez perguntas de resposta múltipla que demora dez minutos. Cada questão é sobre um pequeno trecho de música de um minuto complementado com uma imagem. O utilizador terá até um minuto para responder (ou ignorar) a pergunta e escolher de entre as três respostas a que lhe parece estar correta. Por cada resposta acertada o utilizador amealha dois pontos positivos e por cada resposta errada o utilizador amealha um ponto negativo. No final do jogo, ganha o jogador com mais pontos e amealha três pontos adicionais. Em caso de empate considera-se o tempo total que os utilizadores empatados levaram a responder às questões certas.

Um utilizador poderá jogar vários jogos consecutivos numa única sessão no sistema (nunca em paralelo). Um jogo é um desafio lançado por um utilizador para começar a uma determinada hora (por defeito o desafio é para começar dali a cinco minutos). Qualquer outro utilizador pode aceitar entrar no jogo. Pelo menos um outro jogador terá que aceitar o desafio para que o jogo venha a começar (senão é cancelado pelo servidor). Um utilizador só pode ter um jogo ativo em simultâneo e tendo proposto um desafio não pode jogar desafios de terceiros enquanto o seu não ocorrer. O servidor local do utilizador que lançou o desafio irá gerir os jogos dos seus utilizadores locais, escolhendo aleatoriamente as perguntas da sua base de dados e começando o *broadcast* do jogo para a sua rede local (também pode ser utilizado *unicast UDP*), controlando os jogadores para os quais tem que processar as respostas (apenas para os que aceitaram ou criaram o desafio). As aplicações clientes dos utilizadores têm que saber qual o jogo para a qual aceitaram o desafio (ou propuseram o desafio) e vão processando a informação que recebem do servidor para fazer o interface para o utilizador em tempo adequado, ou seja, armazenam os dados das questões (música, imagem e textos das questões e respostas) num *buffer* e vão mostrando as questões, tocando a respetiva música, no *timing* correto (uma questão a cada minuto).

Cabe aos servidores:

- Fazerem a gestão dos utilizadores da sua rede local (para cada jogador só se precisa manter informação de autenticação com nome ou alcunha e password e do seu *score* global);
- Fazerem a gestão dos trechos musicais, imagens, respetivas perguntas e respostas; esta gestão pode ser feita sem qualquer interface específico, i.e., o administrador do servidor pode utilizar um ficheiro por jogo/desafio, de formatação simples, com a associação entre um *file path* do trecho musical, outro da imagem, uma pergunta, três respostas e o número da resposta correta por cada uma das dez perguntas (ver exemplo da Tabela 1);
- Fazerem a gestão do desenrolar dos jogos ativos dos utilizadores da sua rede local;
- Num sistema com multi-servidores, anunciem/distribuírem a informação local (dados dos desafios e respetivos resultados) pelos outros servidores;
- Manterem uma lista local de Ranking;
- Num sistema com multi-servidores, registar um novo servidor e anuncia-lo aos outros servidores do sistema, etc.

Música	Imagen	Pergunta	Resposta 1	Resposta 2	Resposta 3	Certa
one.mp3	none.jpg	De que grupo se trata?	Abba	U2	The Beatles	2
love.mp3	heart.jpg	De que cantora se trata?	Nina Simone	Sai	Bjork	1
[...]	[...]	[...]	[...]	[...]	[...]	[...]

Tabela 1: Exemplo de dados para duas perguntas dum jogo.

A comunicação entre servidores deve ser sempre feita por *sockets unicast TCP* (através do tipo **INFO**). A comunicação entre clientes e servidores deve ser feita por *sockets UDP* (*unicast* e/ou *broadcast*). Neste último caso, todas as interações são feitas através dum mecanismo de pedido (do cliente para o servidor) e resposta (do servidor para o cliente, sempre através do tipo de comunicação **REPLY**). Todas as comunicações aplicacionais do sistema devem utilizar o mesmo PDU base.

Versão [1 byte] (por defeito, 0)
Segurança [1 byte] (por defeito, 0 – sem segurança)
Label [2 bytes] (identificação que associa respostas a um pedido, 0 – broadcasting)
Tipo [1 byte] (0 – <b>REPLY</b> , 1 – <b>HELLO</b> , 2 – <b>REGISTER</b> , ...)
Número de Campos Seguintes [1 byte]
Tamanho em bytes da Lista de Campos Seguintes [2 bytes]
<i>Lista de Campos Seguintes</i> (dependente do Tipo, pode ser vazia)

PDU Base

A Lista de Campos Seguintes é uma lista dos argumentos/informação respeitante ao pedido ou à resposta a um pedido. Quando não existem argumentos (por exemplo no tipo **HELLO**) o valor do número de campos seguintes e do tamanho da lista de campos seguintes deve ser igual a zero e o PDU acaba aí. Cada campo é formatado/codificado na seguinte sequência: Campo [1 byte] (identificador do campo e que define a sua formatação/codificação, e.g., 1 – Nome de utilizador, 3 – Informação de segurança, 4 – Data, 5 – Hora, etc., ver Tabelas 2-4); Tamanho em bytes [1 byte]; Valor (tamanho máximo de 255 bytes). Os tipos de campos são simples e para se transmitir informações associadas, devem utilizar-se campos consecutivos no PDU.

Tipo de Pedido	Campo	Descrição	Tipo de dados
01 HELLO	-	-	-
02 REGISTER	1 Nome	Nome do utilizador.	Null terminated String
	2 Alcunha	Alcunha do utilizador.	Null terminated String
	3 Sec_Info	Informação de autenticação.	Binário
03 LOGIN	2 Alcunha	Alcunha do utilizador.	Null terminated String
	3 Sec_Info	Informação de autenticação.	Binário
04 LOGOUT	-	-	-
05 QUIT	-	-	-
06 END	-	-	-
07 LIST_CHALLENGES	-	-	-
08 MAKE_CHALLENGE	7 Desafio	Nome dado ao desafio.	Null terminated String
	4 Data	Data para início do desafio.	AAMMDD (6 bytes)
	5 Hora	Hora para início do desafio.	HHMMSS (6 bytes)
09 ACCEPT_CHALLENGE	7 Desafio	Nome do desafio aceite.	Null terminated String
10 DELETE_CHALLENGE	7 Desafio	Nome do desafio a apagar.	Null terminated String
11 ANSWER	6 Escolha	Escolha entre as respostas.	Inteiro (1 byte)
	7 Desafio	Desafio associado à resposta.	Null terminated String
	10 #Questão	Nº da questão a que se responde.	Inteiro (1 byte)
12 RETRANSMIT	7 Desafio	Nome dado ao desafio.	Null terminated String
	10 #Questão	Nº da questão que se quer jogar.	Inteiro (1 byte)
	17 #Bloco	Número de ordem do bloco áudio de que se precisa.	Inteiro (1 byte)
13 LIST_RANKING	-	-	-

Tabela 2: Campos dos tipos de pedidos que os clientes podem enviar aos servidores.

Seguem-se alguns exemplos de interação entre cliente e servidor:

Cliente →	← Servidor
<p>[No interface aplicacional o utilizador preenche formulário de autenticação com alcunha e password e submete...]        &gt;&gt;PDU[ver=0,seg=0,label=6,tipo=LOGIN]        Lista de Campos: [02="Zézé", 03='JDU7736SG668']</p> <p>[No interface aplicacional do cliente aparece o nome do utilizador José Silva e o seu score global de 29 pontos...]</p>	<p>[O servidor reconhece e confirma a autenticação devolvendo o nome do utilizador e o seu score...]        &lt;&lt;PDU[ver=0,seg=0,label=6,tipo=REPLY]        Lista de Campos: [01="José Silva",20=29]</p>

Exemplo 1: Interação de entrada no sistema a utilizador já registado.

Cliente →	← Servidor
<p>[No interface aplicacional o utilizador escolhe lançar um desafio deixando a data e hora para início por defeito, i.e., para dali a cinco minutos...]        &gt;&gt;PDU[ver=0,seg=0,label=12,tipo=MAKE_CHALLENGE]        Lista de Campos: [07="Joguinho"]</p> <p>[No interface aplicacional do cliente aparece a confirmação do lançamento do desafio que começará às 21h:10m:45s do dia 10 de Março de 2015...]</p>	<p>[O servidor reconhece e confirma a criação do desafio...]        &lt;&lt;PDU[ver=0,seg=0,label=12, tipo=REPLY]        Lista de Campos: [07="Joguinho",04=150310,05=211045]</p>

Exemplo 2: Interação quando o utilizador pede para criar um desafio.

Quando um utilizador cria um desafio no seu servidor local, este deve associa-lo, por sorteio, a um dos ficheiros de definição de desafios (com a informação das dez perguntas que fazem parte do jogo – ver exemplo da Tabela 1) pré-existentes na diretoria respetiva.

Cliente →	← Servidor
<p>[No interface aplicacional o utilizador escolhe listar os desafios registados no sistema...]</p> <p>&gt;&gt;PDU[ver=0,seg=0,label=223,tip=LIST_CHALLENGE]</p> <p>Lista de Campos: [ ]</p> <p>[No interface aplicacional do cliente aparece a lista dos desafios pendentes e utilizador aceita um deles:]</p> <p>Joguinho - 21h:10m:45s do dia 10 de Março de 2015</p> <p>Jogatina - 21h:15m:00s do dia 10 de Março de 2015</p> <p>Challenge - 21h:40m:12s do dia 10 de Março de 2015...]</p> <p>&gt;&gt;PDU[ver=0,seg=0,label=224,tip=ACCEPT_CHALLENGE]</p> <p>Lista de Campos: [07="Jogatina"]</p>	<p>[O servidor reconhece e devolve a lista pretendida...]</p> <p>&lt;&lt;PDU[ver=0,seg=0,label=223,tip=REPLY]</p> <p>Lista de Campos: [07="Joguinho",04=150310,05=211045,</p> <p>07="Jogatina",04=150310,05=211500,</p> <p>07="Challenge",04=150310,05=214012,]</p> <p>Nota: O servidor poderia também ter enviado a informação em três PDUs separados.</p> <p>[O servidor reconhece e confirma...]</p> <p>&lt;&lt;PDU[ver=0,seg=0,label=224,tip=REPLY]</p> <p>Lista de Campos: [00=0]</p>

Exemplo 3: Interação do utilizador quando pede a listagem dos desafios e aceita um.

Cliente →	← Servidor
<p>[No interface aplicacional o utilizador escolhe listar os desafios registados no sistema...]</p> <p>&gt;&gt;PDU[ver=0,seg=0,label=344,tip=LIST_RANKING]</p> <p>Lista de Campos: [ ]</p> <p>[No interface aplicacional do cliente aparece o ranking: Josefa Pá (Zefa) 44 pontos, Carlos Só (Carlão) 39 pontos, etc...]</p>	<p>[O servidor reconhece e devolve a lista pretendida...]</p> <p>&lt;&lt;PDU[ver=0,seg=0,label=344,tip=REPLY]</p> <p>Lista de Campos: [01="José Silva",02="Zézé",20=29,</p> <p>01="José Sá",02="Zé",20=19,01="Carlos</p> <p>Só",02="Carlão",20=39,01="Josefa Pá",02="Zefa",20=44]</p> <p>Nota: O servidor poderia também ter enviado a informação em quatro PDUs separados.</p>

Exemplo 4: Interação quando o utilizador pede para listar o ranking.

Um sistema deve ter, no mínimo, um servidor. Nesse caso o serviço estará apenas disponível na sua rede local. Por cada rede local para onde se queira expandir o sistema terá que se adicionar um servidor local aos já existentes. Um sistema com mais do que um servidor terá a vantagem de distribuir o jogo por um número maior de utilizadores.

Na implementação do sistema, os clientes ligam-se sempre ao servidor onde fizeram o seu registo e é-lhes indiferente o número e organização dos restantes servidores, caso existam. Os clientes utilizam sempre o mesmo servidor como interface para o sistema.

Nalgumas redes locais não é possível a utilização de comunicações broadcast. Neste caso, a solução é o utilizador instalar um pacote de servidor + cliente na sua máquina ou ligar-se a um servidor remoto, sendo que toda a comunicação entre cliente e servidor será unicast UDP. O cliente tenta encontrar primeiro um servidor na rede local por broadcast e, não encontrando, deve contactar diretamente um servidor com endereço por defeito e que pode até ser na sua própria máquina em 127.0.0.1 (ou num endereço IP e porta fixos, pré-determinados num ficheiro de configuração ou registados em DNS). Um servidor deve registrar o endereço IP e porta dum cliente que o tenha contactado com HELLO sem utilizar broadcast e nunca usar broadcast no envio de informação das questões durante um desafio, i.e., para esse cliente deve usar-se sempre unicast UDP.

O PDU do tipo RETRANSMIT serve também para implementar um mecanismo simples de controlo de erros por parte dos clientes. Quando um cliente deteta a falta de receção (ou receção errada) de dados duma questão, utiliza este tipo de PDUs com os campos 7, 10 e 17 para requerer ao servidor o reenvio da informação em falta.

Cliente →	← Servidor
[No interface aplicacional o utilizador está pronto para jogar um desafio, ou o próprio ou um que tenha aceite (ver exemplo 3)...]	
[Cliente recebe PDU e guarda em memória/buffer a informação parcial recebida...]	[Quando se chega à data e hora de determinado desafio o servidor envia os dados da primeira questão do jogo por broadcasting/unicasting...] <<PDU[ver=0,seg=0,label=0,tipo=REPLY] Lista de Campos: [07="Jogatina",10=1,11="De que grupo se trata?",12=1,13="Abba",12=2,13="U2",12=3,13="The Beatles",16=<conteúdo dum ficheiro de imagem>,254=0]
[Cliente recebe PDU e guarda em memória/buffer a informação parcial recebida...]	<<PDU[ver=0,seg=0,label=0,tipo=REPLY] Lista de Campos: [07="Jogatina",10=1,17=1,18=<conteúdo do 1º bloco áudio de 48Kbytes>,254=0]
[Cliente recebe PDU e guarda em memória/buffer a informação parcial recebida...]	<<PDU[ver=0,seg=0,label=0,tipo=REPLY] Lista de Campos: [07="Jogatina",10=1,17=2,18=<conteúdo do 2º bloco áudio de 48Kbytes>,254=0]
[...]	[...]
[Cliente recebe último PDU de informação sobre a 1ª questão e guarda em memória/buffer. No interface aplicacional do utilizador aparece um cronómetro decrescente, a imagem e o texto da questão e das respostas alternativas e começa a ouvir-se o trecho musical reconstruído a partir de todos os blocos recebidos...]	<<PDU[ver=0,seg=0,label=0,tipo=REPLY] Lista de Campos: [07="Jogatina",10=1,17=99,18=<conteúdo do 99º e último bloco áudio de 30Kbytes>]
[Utilizador escolhe a resposta 2 no interface...] >>PDU[ver=0,seg=0,label=983,tipo=ANSWER] Lista de Campos: [07="Jogatina",10=1,6=2]	[O servidor reconhece e indica se a resposta está correta ou não e os pontos amealhados...] <<PDU[ver=0,seg=0,label=983,tipo=REPLY] Lista de Campos: [07="Jogatina",10=1,14=1,15=2]
[Utilizador é informado que a resposta está correta e amealhou mais 2 pontos positivos...]	

Exemplo 5: Interação quando o utilizador responde à 1ª questão dum jogo num desafio.

### Sistema Multi-Servidor

A existência de mais do que um servidor potencia a distribuição do serviço a um maior número de utilizadores. O sistema deve começar com um único servidor. Sempre que se acrescenta um servidor ao sistema este deve ser registado num dos servidores já existentes que depois difundirá a informação pelos outros servidores já existentes. Todos os servidores sabem sempre quem são os outros servidores. Quando um servidor é colocado em execução devem ser passados como argumentos a porta UDP em que fica à escuta para interação com os clientes e a porta TCP em que fica à escuta para interação com outros servidores. Além disso, devem ainda ser passados como argumentos o endereço IP e a porta TCP dum servidor já existente para registo no sistema (se estes dois argumentos não forem indicados então o servidor é o primeiro e o único no sistema).

Todas as comunicações entre servidores são feitas por difusão em *unicast* TCP utilizando o tipo INFO e respetivos campos enumerados na Tabela 4. Ou seja, a rede de servidores funciona em malha, com difusão total ou parcial, dependendo da informação em causa.

Tipo da Resposta	Campo	Descrição	Tipo de dados
00 REPLY	0 OK	Resposta a <b>HELLO</b> , <b>REGISTER</b> , <b>LOGOUT</b> , <b>QUIT</b> e <b>ACCEPT_CHALLENGE</b> .	- (um byte a zero)
	255 Erro	Pequena descrição do erro.	<i>Null terminated String</i>
	254	A lista continua noutro PDU.	- (um byte a zero)
	1 Nome	Nome do utilizador. Usado na resposta a <b>LOGIN</b> e <b>LIST_RANKINGS</b> .	<i>Null terminated String</i>
	2 Alcunha	Alcunha do utilizador. Usado na resposta a <b>LIST_RANKINGS</b> .	<i>Null terminated String</i>
	4 Data	Data para início do desafio. Usado na resposta a <b>LIST_CHALLENGES</b> e <b>MAKE/DELETE_CHALLENGE</b> .	AAMMDD (6 bytes)
	5 Hora	Hora para início do desafio. Usado na resposta a <b>LIST_CHALLENGES</b> e <b>MAKE/DELETE_CHALLENGE</b> .	HHMMSS (6 bytes)
	7 Desafio	Nome dado ao desafio. Usado na resposta a <b>LIST_CHALLENGES</b> , <b>MAKE/DELETE_CHALLENGE</b> e na identificação do desafio a que correspondem outros dados.	<i>Null terminated String</i>
	10 #Questão	Número da questão num desafio.	Inteiro (1 byte)
	11 Questão	Texto da questão num desafio.	<i>Null terminated String</i>
	12 #Resposta	Número da resposta num desafio.	Inteiro (1 byte)
	13 Resposta	Texto da questão num desafio.	<i>Null terminated String</i>
	14 Certa	Indica se a resposta do jogador está certa ou errada (1/0). Resposta a <b>ANSWER</b> .	Inteiro (1 byte)
	15 Pontos	Resposta a <b>ANSWER</b> . Devolve pontos amealhados numa questão.	Inteiro (1 bytes)
	16 Imagem	Imagen complementar duma questão.	Formato JPG
	17 #Bloco	Número de ordem do bloco áudio.	Inteiro (1 byte)
	18 Audio	Bloco áudio para o trecho musical. Máx. 48Kbytes e 250 blocos. Os blocos podem ou não ser numerados através do campo 17.	Formato WAV ou MP3
	20 Score	Resposta a <b>END</b> (score obtido no jogo), <b>LOGIN</b> (score global), <b>LIST_RANKINGS</b> (score global).	Inteiro (2 bytes)

Tabela 3: Campos dos tipos de respostas que os servidores podem enviar aos clientes.

Um servidor faz difusão total da informação (para todos os outros servidores) quando anunciar:

- Um registo dum novo servidor no sistema;
- Um registo dum novo desafio;
- A lista dos desafios disponíveis localmente (criados por utilizadores locais);
- Um registo de aceitação dum desafio;
- Os resultados dum desafio;
- O ranking dos utilizadores locais.

Por outro lado, um servidor faz difusão parcial da informação (apenas para um servidor) quando receber a primeira aceitação remota (por parte dum utilizador que está registado num servidor remoto) para um desafio criado por um utilizador local. Neste caso, o servidor local difunde a informação do desafio (data e hora de início, músicas, imagens e textos das perguntas e respostas alternativas) apenas para esse servidor remoto. O servidor remoto recebe a informação que armazena na estrutura de ficheiros predefinida (uma diretoria para trechos musicais, outra para imagens e outra para ficheiros com a definição dos desafios) e que depois utiliza na implementação do desafio à hora e na data pré-determinada.

Depois dum novo servidor fazer o seu registo num servidor já existente, este último difunde a informação desse registo para todos os outros servidores já existentes no sistema. Depois disso, todos os servidores existentes difundem para o novo servidor a lista dos desafios locais pendentes e o ranking de todos os utilizadores locais, ficando assim terminada a fase de integração dum novo servidor no sistema.

Sempre que é criado um desafio num servidor, a informação desse novo desafio deve ser difundido para todos os outros servidores do sistema. Sempre que um utilizador aceita um desafio, o seu servidor local tem que difundir a informação pelos outros servidores do sistema. Da mesma forma, quando termina a sessão dum desafio num servidor, os resultados desse desafio (pontos obtidos pelos jogadores locais) devem ser enviados para os clientes dos utilizadores que jogaram localmente e difundidos para os outros servidores.

Tipo de Pedido	Campo	Descrição	Tipo de dados
14 INFO	1 Nome	Nome do utilizador.	Null terminated String
	2 Alcunha	Alcunha do utilizador.	Null terminated String
	7 Desafio	Nome dum desafio.	Null terminated String
	4 Data	Data para inicio dum desafio.	AAMMDD (6 bytes)
	5 Hora	Hora para inicio dum desafio.	HHMMSS (6 bytes)
	10 #Questão	Número duma questão.	Inteiro (1 byte)
	11 Questão	Texto duma questão.	Null terminated String
	12 #Resposta	Número duma resposta.	Inteiro (1 byte)
	13 Resposta	Texto duma questão.	Null terminated String
	14 Certa	Indica qual a resposta certa.	Inteiro (1 byte)
	16 Imagem	Imagen complementar duma questão.	Formato JPG
	19 Música	Trecho musical de aproximadamente 1 minuto.	Formato WAV ou MP3
	20 Score	Score dum jogador num desafio ou num ranking.	Inteiro (2 bytes)
	30 IP Servidor	Endereço IP dum servidor.	Endereço IPv4 (4 bytes)
	31 Porta Servidor	Porta aplicacional dum servidor.	Porta aplicacional (2 bytes)

Tabela 4: Campos do tipo de informação que os servidores podem difundir entre si.

Tal como na conversação entre clientes e servidores, a informação dum PDU do tipo **INFO** pode conter vários campos associados permitindo assim o envio de variadas informações num único pacote. Por exemplo, para enviar a informação da criação pelo utilizador “Zézé” dum novo desafio chamado “Jogatina”, que irá decorrer no dia 10 de Março de 2015 às 21h:15m:00s, o servidor onde o registo do desafio foi criado originalmente tem que enviar a cada um dos restantes servidores um PDU do tipo **INFO** com os seguintes campos: 07=“Jogatina”, 04=150310, 05=211500, 01=“José Silva”, 02=“Zézé”.

## 1ª FASE

No arranque do trabalho será fornecido um *kit* de arranque com ficheiros de dados suficientes para um desafio. Estão incluídos ficheiros áudio, imagens e ficheiro de definição das questões do desafio.

Na primeira fase do trabalho, que deve estar pronta ao fim de três aulas, devem implementar-se versões base do servidor e cliente com os seguintes requisitos:

- Não existe autenticação segura, isto é, todos os acessos são autorizados e os registos são simples sem encriptação. Também não existe noção de sessão aberta no servidor nem noção de *ranking* global de pontuações.
- As funcionalidades que devem estar implementadas são: entrar no sistema (sem autenticação segura), criar desafio, aceitar desafio, jogar um desafio e obter a pontuação desse desafio.
- Os únicos desafios disponíveis são os já fornecidos no *kit* de arranque.
- Além disso, existe um único servidor no sistema e não precisa ser encontrado por *broadcast*, e pode estar acessível na mesma máquina onde correm os clientes (pelo menos dois). Ou seja, só é preciso implementar a comunicação *unicast* UDP entre dois clientes e um servidor.

## 2ª FASE

Nesta fase, que deve estar pronta ao fim de mais duas semanas, deve ser acrescentada a possibilidade do sistema funcionar em malha de servidores (i.e., com pelo menos dois servidores).

Assim, no mínimo, devem ser implementadas as seguintes funcionalidades nos servidores:

- Registo local de utilizadores;
- O registo e difusão de novos servidores;
- A difusão da lista de desafios disponíveis;
- A difusão dos dados dum desafio e dos seus resultados;
- Cálculo de *ranking* local e do *ranking* global (a partir da informação agregada dos *rankings* locais);
- A difusão do *ranking* dos utilizadores locais.

Além disso, o interface nos clientes deve ser atualizado para permitir ao utilizador aceder às novas funcionalidades.

### **3<sup>a</sup> FASE**

Na última semana dedicada ao trabalho os grupos podem, opcionalmente, incluir/adicionar as outras funcionalidades (as já previstas neste enunciado e outras novas que achem pertinentes). Devem, por fim, produzir um relatório final.

### **Relatório**

O relatório deve ser escrito em formato de artigo com um máximo de dez páginas (não incluindo anexos com exemplos da aplicação, listagens diversas e código fonte) e recomenda-se o uso do formato *Lecture Notes in Computer Science* (instruções para autores em <http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0>).

O relatório deve descrever o essencial do desenho e implementação com a seguinte estrutura recomendada:

- Introdução;
- Diferenças/Adições na especificação do protocolo em relação à apresentada no enunciado;
- Implementação (detalhes, parâmetros, bibliotecas de funções, etc.);
- Testes e resultados;
- Conclusões e trabalho futuro.