Trabalho Prático Nº2 – Serviço *Over the Top* para entrega de multimédia

Duração: 7 aulas (8 semanas)

Motivação

Ao longo do último meio século de vida da Internet (a rede das redes), observou-se uma mudança irreversível de paradigma. A comunicação extremo-a-extremo, de sistema final para sistema final, dá lugar ao consumo voraz de conteúdos de qualquer tipo, a todo o instante, em contínuo e muitas vezes em tempo real. Este novo padrão de uso coloca grandes desafios à infraestrutura IP de base que a suporta. Apesar de não ter sido originalmente desenhada com esse requisito, tem sido possível resolver a entrega massiva de conteúdos com redes sofisticadas de entrega de conteúdos (CDNs) e com serviços específicos, desenhados sobre a camada aplicacional, e por isso ditos *Over the Top* (OTT). Um serviço de multimedia OTT, pode por exemplo usar uma rede *overlay* aplicacional (ex: *multicast aplicacional*), devidamente configurada e gerida para contornar os problemas de congestão e limitação de recursos da rede de suporte, entregando em tempo real e sem perda de qualidade os *media* diretamente ao cliente final. Serviços bem conhecidos como o Netflix ou o Hulu, fazem *streaming* sobre a rede IP pública. Daí a designação de "*Over-the-top streaming services".* Para tal, formam uma rede *overlay* própria, assente em cima dos protocolos de transporte (TCP ou UDP) e/ou aplicacionais (HTTP) da Internet. Neste trabalho, pretende-se conceber e prototipar um desses serviços, que promova a eficiência e a otimização de recursos para melhor qualidade de experiência do utilizador.

Objetivos

Usando primariamente o emulador CORE como bancada de teste, e uma ou mais topologias de teste, pretende-se conceber um protótipo de entrega de áudio/vídeo/texto com requisitos de tempo real, a partir de um servidor de conteúdos para um conjunto de N clientes. Para tal, um conjunto de nós pode ser usado no reenvio dos dados, como intermediários, formando entre si uma rede de *overlay* aplicacional, cuja criação e manutenção deve estar otimizada para a missão de entregar os conteúdos de forma mais eficiente, com o menor atraso e a largura de banda necessária. A forma como o *overlay* aplicacional se constitui e se organiza é determinante para a qualidade de serviço que é capaz de suportar. A Figura 1 ilustra esta visão geral.

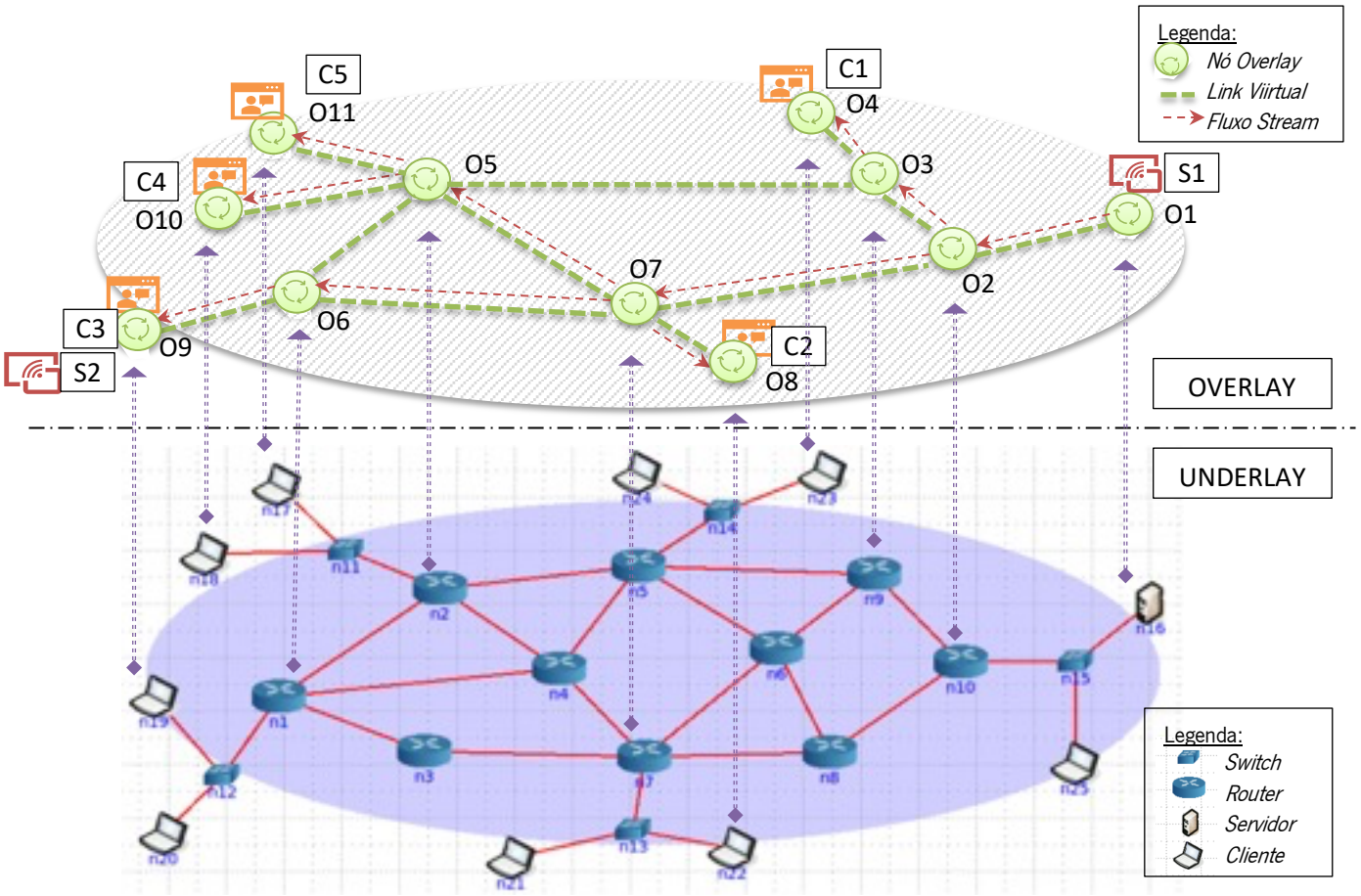


Figura 1: Visão geral de um serviço OTT sobre uma infraestrutura IP

Descrição detalhada

Na Figura 1 estão representados 5 clientes (**C1**, **C2**, **C3**, **C4** e **C5**) com interesse em consumir uma *stream multimédia* disponível no servidor **S1**, replicada no servidor **S2**. Não havendo *overlay* aplicacional, cada um dos 5 clientes tem de abrir uma conexão para um dos servidores, e esse servidor tem de gerar 5 fluxos de dados idênticos, um por cada cliente. Esta solução tem problemas óbvios de escalabilidade, pois a partir de um determinado número de clientes N, o servidor ativo e/ou a rede deixam de conseguir suportar os N fluxos sem perda de qualidade. Uma abordagem mais eficiente pode ser conseguida adicionando nós de uma rede *overlay* (**O1**, **O2**, ... **O11**) construída sobre a rede *underlay* de suporte. Os nós da rede *overlay* estão instanciados diretamente em nós da rede *underlay*, pois são na verdade aplicações em execução em determinados sistemas físicos previamente selecionados. Para facilitar o cenário, vamos considerar que inicialmente apenas há um servidor de *streaming* e que esse servidor é também um nó da rede *overlay*. Ou seja, o servidor **S1** da Figura 1, é também o nó **O1** da rede *overlay*, em execução no sistema servidor **n16** da rede de suporte. Por seu lado os clientes de *streaming* (**C1**, **C2**, **C3**, **C4** e **C5**), responsáveis por receber e reproduzir os conteúdos, são também nós *overlay* (respetivamente **O4**, **O8**, **O9**, **O10** e **O11**), instanciados em equipamentos da rede *underlay* (**n23**, **n22**, **n19**, **n18** e **n17**).

De salientar que a rede *overlay* tem muito menos nós e ligações que a rede de suporte, pois não é imaginável nem desejável que o mapeamento possa ser de um para um. Logo, as ligações na rede *overlay* são na realidade conexões de transporte (TCP ou UDP), que atravessam uma ou mais ligações na rede física. Por exemplo, a ligação virtual entre **O3** e **O5**, vai provavelmente ser suportada pelos links **n9** ←→ **n5** ←→ **n2** da rede física, se essa for a melhor rota IP de **n9** para **n2** (por defeito calculada com recurso ao protocolo OSPF, no caso do emulador CORE). A localização dos nós da rede *overlay* pode ser calculada ou escolhida manualmente. Na figura os nós da rede *overlay* estão instanciados em todo o tipo de equipamentos, incluindo *routers*, o que sendo possível no emulador CORE não é realista. Numa rede real, os *switches* e *routers* são equipamentos dedicados, que não executam aplicações genéricas. No CORE é possível porque todos os sistemas são *containers* Linux.

Quando um nó da rede *overlay* começa a sua execução, precisa de conhecer pelo menos um outro nó qualquer da rede a quem se ligar. Ou, em alternativa, usar um nó bem conhecido e predeterminado (*bootstrapper ou tracker*), cuja função é ajudar a construir a rede. Esse primeiro contacto serve simultaneamente para se dar a conhecer e para obter uma lista de outros nós com quem se ligar. Pode-se usar por exemplo um dos servidores de *streaming* como *bootstrapper* permanente ou outro nó, que vai conhecer todos os nós da rede *overlay*. Uma estratégia simples para se manter o *overlay* a funcionar é procurar garantir que cada nó tem pelo menos N vizinhos (ex: N=3). Estas ligações virtuais com os vizinhos devem ser mantidas ativas, e eventualmente monitorizadas, durante todo o tempo de execução.

Uma vez construído o *overlay*, torna-se necessário identificar os melhores caminhos para o fluxo de dados, desde o servidor aos clientes. Para isso, os nós da overlay precisam de criar uma rota para o fluxo, ou uma tabela de rotas. Essa tabela pode ser preenchida por iniciativa do servidor de *streaming*, que envia uma mensagem a inundar a rede em todas as ligações para se dar a conhecer, ou por iniciativa dos clientes, que inundam a rede a pedir pelo conteúdo. Ou outra estratégia dinâmica que preencha as tabelas com caminhos e custos desses caminhos. Como métrica pode-se usar o nº de saltos, ou outras medidas de qualidade das ligações (atraso, largura de banda). Na figura vemos facilmente que o melhor caminho de **S1** para o **C1** é **O1-O2-O3-O4**. Por sua vez, o melhor caminho de **S1** para **C2** é **O1-O2-O7-O8**. Mas não queremos que os dados passem duas vezes na ligação **O2-O1** que é comum a estes clientes, gerando tráfego redundante e ineficiência. Para tal, a tabela com a rota em **O2** deve dizer o seguinte: os pacotes do fluxo **F**, chegam vindos de **O1**, pela conexão de entrada **O1-O2**, com uma métrica de 1 salto, e devem ser replicados para **O7** e **O3**, nas conexões de saída **O2-O7** e **O2-O3**. Ou, de forma resumida: *{Fonte:S1, Fluxo: F, Origem: O1, Métrica: 1, Destinos: O7, O3, Estado: ativa }*.

Etapas sugeridas

Nota: as etapas não precisam ser seguidas obrigatoriamente de forma sequencial!

- Etapa 0: Preparação das atividades
  - Escolher a linguagem de programação mais conveniente para todos os elementos do grupo;
  - Preparar uma topologia para testes e criar cenários de teste nessa topologia (ver cenários sugeridos);
  - Definir qual o protocolo de transporte que vai ser usado no *overlay*: TCP ou UDP (também seria possível equacionar o uso de um protocolo de aplicação como o HTTP, mas talvez seja mais complexo);

- Implementar um cliente/servidor simples, baseado em exemplos, para o protocolo de suporte escolhido (TCP ou UDP), eventualmente que seja simultaneamente cliente e servidor;

- **Etapa 1**: Construção da Topologia *Overlay*
  - Construir uma aplicação (ex: **oNode**) que é simultaneamente cliente e servidor, que atende numa porta predefinida, e que é capaz de receber e enviar mensagens em modo full-duplex; testar com envio e receção de mensagem simples de "HELLO";
  - Definir uma estratégia de construção da rede *overlay*, como alternativas pode considerar:
  - Estratégia 1: Abordagem manual. Ao executar o programa indicam-se os X vizinhos em configuração ou na linha de comando. Exemplo: **$ oNode <vizinho1> <vizinho2> <vizinho3>**. A tabela de vizinhos e respetivas conexões não se altera durante a execução. Não há necessidade de nenhum nó de controlo.
  - Estratégia 2: Abordagem baseada num controlador. Ao executar o programa, indica-se apenas um nó como contacto para arranque da rede. Exemplo: **$ oNode <bootstrapper>**. O novo nó envia mensagem de registo, a identificar-se, e recebe como resposta uma lista de X vizinhos. Cabe ao servidor definir quais são esses vizinhos com base no conhecimento que tem de todo o *overlay*, que para facilitar esta tarefa, sugere-se que o servidor obtenha essa informação de quem se deve ligar a quem a partir de ficheiro de configuração, construído manualmente. Definir estratégia para abandono, se o nó avisa, se deve registar-se periodicamente, etc..
  - Outras estratégias... (sugerir ou procurar alternativas, discutindo-as com o docente);
  - Manter as ligações com os vizinhos ativas, começando por monitorar o seu estado (ver Etapa 3).

- **Etapa 2**: Serviço de *Streaming*
  - Estratégia 1: Implementar um cliente e um servidor simples com base no código do livro de apoio [1];
    - Usar o código do livro de apoio (disponível em Python e em Java) como ponto de partida;
    - Adaptar o código se a linguagem de programação escolhida não for nem Python ou Java;
    - Com base no exemplo, previamente ajustado e comentado pela equipa docente [2], fazer um servidor capaz de ler o vídeo de um ficheiro e o enviar em pacotes, numerados, para a rede *overlay*;
    - Com base no exemplo, previamente ajustado pela equipa docente, fazer um cliente capaz de receber pacotes da rede *overlay*, com um número de sequência, e reproduzir o vídeo numa janela;
    - Usar como vídeo de teste o exemplo do livro *movie.Mjpeg* [2] (trata-se de um vídeo básico, de fluxo constante, que é uma sequência simples de imagens independentes, a enviar a intervalo de tempo fixo);
  - Estratégia 2: Adaptar o código do livro de apoio para utilizar uma biblioteca alternativa, capaz de ler codecs adicionais, e recorrendo a outros vídeos para difusão.
  - Outras estratégias... (sugerir ou procurar alternativas, discutindo-as com o docente).

- **Etapa 3**: Monitorização da Rede Overlay
  - Para além do que será a rede de entrega de conteúdos, pretende-se que cada servidor da topologia difunda periodicamente uma mensagem de prova (teste) na rede que irá permitir obter um conhecimento razoavelmente atualizado das condições de entrega na rede *overlay*. Cada nó que receba a mensagem deve reenviá-la a todos os vizinhos, com o cuidado de evitar repetições em ciclo e de enviar a quem lhe enviou a ele (inundação controlada, por exemplo enviando só uma vez cada mensagem, evitando enviar para trás pela conexão de onde recebeu).
    Essa mensagem pode ser estruturada incluindo a identificação do servidor, o *nº de saltos* que a mensagem dá e o instante temporal em que a mesma foi enviada, para que possa ser calculado o *atraso* sofrido desde que o seu envio até à sua receção. Como sugestão, registe também a interface que conduz à melhor rote de volta à fonte. Desta forma, cada nó da rede terá sempre conhecimento do número de nós envolvidos até ao servidor (fonte de dados) e uma estimativa do atraso na ligação. Se entender pode usar outras métricas adicionais.
    Como teste inicial, pode considerar apenas um dos servidores (S1), definir atrasos de, por exemplo, 10ms por ligação (para facilitar pode introduzir o atraso em ligações da rede *underlay*), e verificar as métricas observadas por nó da *overlay*. Uma vez testada e operacional, pode estender a monitorização da rede ao servidor S2. Note que com este

tipo de controlo, os nós *overlay* folha (de acesso direto aos clientes) passam a conhecer qual dos servidores está em melhores condições para realizar a entrega de conteúdo.

- Etapa 4: Construção de Rotas para a Entrega de Dados

    - Esta etapa pode ser vista como uma extensão da etapa anterior.
    - Para escolha da melhor rota, cada nó da overlay deve considerar a métrica mais favorável; por exemplo, o menor atraso, e para atrasos idênticos, o menor número de saltos.
    - Estratégia 1: Por iniciativa do servidor de *streaming*, com anúncios periódicos.
        - Servidor envia mensagem de controlo com anúncio; mensagem inclui identificação do servidor, identificação do fluxo, etc.;
        - Ao receber a mensagem cada nó constrói uma tabela de rotas, com informação de Servidor, Fluxo, Origem, Métrica, Destinos, Estado da Rota; até serem necessárias as rotas ficam num estado inativo.
        - Cada nó que receba a mensagem deve enviá-la a todos os vizinhos, usando inundação controlada;
        - Antes de reenviar a mensagem, a métrica deve ser atualizada;
        - As entradas estão inicialmente inativas, sendo ativadas pelo cliente quando se liga, enviado uma mensagem de ativação da rota, pelo percurso inverso;
        - Cada mensagem com pedido de ativação do cliente deve ser reencaminhada em cada nó seguido o campo "Origem" da rota;
        - O cliente só envia um pedido de ativação, quando deseja receber os dados; não havendo clientes, as rotas existem nas tabelas, mas estão inativas, não havendo tráfego;

    - Estratégia 2: Por iniciativa do Cliente de *streaming*, enviando um pedido diretamente ao servidor
        - A iniciativa pode ser do cliente que pretende receber a *stream*, enviando um pedido explícito de rota para essa *stream*;
        - Esta estratégia pressupõe que o servidor tem informação de registo e conhece o *overlay*, podendo por isso escolher o percurso e ensiná-lo ao cliente, que usa a informação obtida para efetivar o percurso;
        - Se o servidor não conhece a topologia, o pedido do cliente tem de ser enviado para todos (inundação controlada) até chegar a um nó que já tem informação da *stream*;
        - Alternativamente podem ser criadas rotas estáticas na *overlay*.

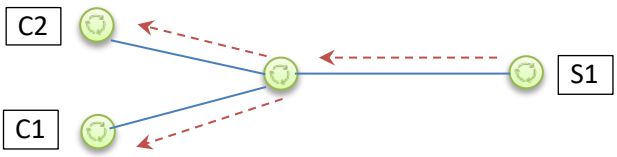
    - Estratégia 3: Construção de uma infraestrutura de entrega (árvore) partilhada
        - Nesta estratégia deve ser designado um nó *overlay* específico (*rendezvous point (RP)*) a partir do qual são criadas as rotas de entrega recorrendo à Estratégia 1 ou Estratégia 2. Neste cenário, os servidores S1 e S2 enviam a *stream* multimédia em *unicast* ao RP, efetuando-se a difusão do fluxo a partir daí. Como no presente serviço de *streaming*, os servidores disponibilizam o mesmo conteúdo, esse conteúdo deve ser enviado apenas pelo servidor que apresente melhores métricas no caminho até ao RP.
    - Outras estratégias... (sugerir ou procurar alternativas, discutindo-as com o docente).

- Etapa 5: Ativação e Teste do Servidor Alternativo
    - O objetivo desta etapa é forçar a ativação do servidor alternativo de forma transparente ao cliente quando as condições na entrega da *stream* se degradam. Para isso pode-se provocar uma situação adversa na rede *overlay* introduzindo um atraso excessivo num das ligações da rede *underlay* (para simplificar) que afete a entrega do fluxo em curso. Desta forma, com base nos resultados de monitorização, um nó da *overlay* adjacente a um cliente ativo pode, no interesse do cliente, abandonar a árvore com origem em S1 e ativar as rotas necessárias para passar a receber o conteúdo de S2. No caso de ser usada uma árvore de entrega partilhada, a decisão de ativação ou não do servidor alternativo deve ser tomada no RP.

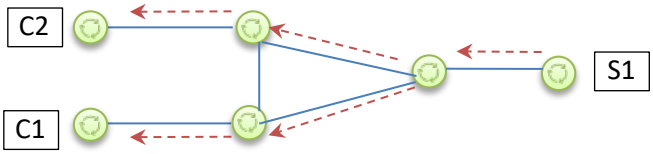
- Etapa Opcional: Definição do método de recuperação de falhas

- Estratégia 1: Cálculo e redistribuição de rotas centralizado
  - Iniciativa pode partir dos nós adjacentes no *overlay*, ou do servidor central, dependendo do método de monitorização de serviço implementado;
  - Deve assegurar que as rotas calculadas são as mais eficientes para a nova configuração do *overlay*;
  - Poderão ser testados diversos métodos de convergência, de modo a minimizar o período em que o serviço não está disponível.
- Estratégia 2: Restabelecimento de rotas por inundação controlada
  - Estratégia com potencial para recuperar muito mais rapidamente do que uma implementação centralizada, porém mais desafiante tecnicamente;
  - A iniciativa partirá necessariamente dos nós remanescentes no *overlay*;
  - Podem ser implementados métodos para minimizar o número de pacotes trocados;
  - A estratégia adotada para a criação de novas ligações pode ter também uma componente de profundidade para que o seu estabelecimento seja o mais célere possível.
- Outras estratégias a discutir com o docente.

- **Sugestões para cenários de teste:**
  - Cenário 1: *overlay* com 4 nós (um servidor, dois clientes e um nó intermédio).
    Posteriormente, considerar um servidor adicional (S2) conectado ao nó intermédio.



  - Cenário 2: *overlay* com 6 nós (um servidor, dois clientes, 3 nós intermédios).
    Posteriormente, considerar um servidor adicional S2 conectado a qualquer dos nós intermédios.



  - Cenário 3: *overlay* complexo usando a Figura 1 como fonte de inspiração.

Avaliação

Consideram-se os seguintes itens na avaliação com respetivos pesos relativos:

1. Etapa 1 – Construção da Topologia Overlay (15%)
2. Etapa 2 – Serviço de Streaming (15%)
3. Etapa 3 – Monitorização da Rede Overlay (10%)
4. Etapa 4 – Construção das Rotas para Entrega de Dados (20%)
5. Etapa 5 – Ativação e Teste do Servidor Alternativo (10%
6. Solução final integrada (30%)

Para cada um dos itens, consideram-se os seguintes subitens:
a. Especificação conceptual
b. Qualidade da implementação
c. Qualidade dos testes e da própria apreciação crítica da solução apresentada

Adicionalmente, na solução final (item 6) consideram-se também os subitens:
d. Qualidade da defesa/apresentação
e. Qualidade do relatório

A lista mínima de requisitos a considerar é a seguinte:

- Um nó da rede *overlay* é um programa (aplicação) que se executa manualmente numa máquina da rede física
- Quando se junta à rede *overlay*, um nó deve estabelecer e manter conexões de transporte (TCP ou UDP) com X nós da rede, que serão seus vizinhos;
- Ao iniciar a execução, o nó da rede tem de conhecer pelo menos um outro nó da rede, que pode ser o servidor de conteúdos, que o ajuda a obter a sua lista de X vizinhos;
- Os nós da rede *overlay* precisam de construir e manter uma tabela de rotas (uma por fluxo). Exemplo de elementos constantes na tabela: (Servidor, Fluxo, Custo, Origem, Destinos, Estado);
- Os nós da rede *overlay* reenviam todos os dados de acordo com essa tabela, reenviando por inundação controlada para todos quando não tiverem destino para os mesmos;
- Os clientes devem ligar-se utilizando o mesmo executável, podendo conhecer o endereço do *bootstrapper*, ou dos nós aos quais deve tentar obter o *stream* desejado;
- Deve ser possível visualizar o *stream* de vídeo difundido pelo servidor, obtendo-o do nó de *overlay* mais próximo;
- A rede deve suportar vários clientes concorrentes, minimizando sempre o número de fluxos (e consequentemente a quantidade de dados transmitida).

Adicionalmente, poderão ser implementadas funcionalidades extra, como por exemplo:

- Tolerância a falhas dos nodos da rede *overlay*
- Configuração automática para topologias distintas

  Outras ideias a discutir com o docente

Referências

- Kurose, J. (2016). *Computer Networking: A Top-Down Approach* (7th edition.). Pearson. Retrieved from https://www.pearson.com/us/higher-education/program/Kurose-Computer-Networking-A-Top-Down-Approach-7th-Edition/PGM1101673.html?tab=resources
- Exemplos do Livro anterior (adaptados): https://marco.uminho.pt/disciplinas/ESR/ProgEx.zip

  (*wget https://marco.uminho.pt/disciplinas/ESR/ProgEx.zip)*

Relatório

O relatório deve ser escrito em formato de artigo com um máximo de 12 páginas (recomenda-se o uso do formato LNCS - *Lecture Notes in Computer Science*, instruções para autores em http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0). Deve descrever o essencial da conceção e implementação, devidamente justificado, com a seguinte estrutura recomendada:

- Introdução
- Arquitetura da solução
- Especificação do(s) protocolo(s)
  * formato das mensagens protocolares
  * interações
- Implementação
  * detalhes, parâmetros, bibliotecas de funções, etc.
- Testes e resultados
- Conclusões e trabalho futuro

Submissão

Cada grupo deve fazer a submissão (*upload*) do trabalho, usando a opção de "troca de ficheiros" associada ao seu grupo nos "Grupos" do Blackboard (*elearning.uminho.pt*), da seguinte forma:

- **ESR-TP2-PLxy-Rel.pdf** para o relatório
- **ESR-TP2-PLxy-Codigo.zip** ou **ESR-TP2-PLxy-Codigo.rar** para a pasta com o código

em que **x** é o identificador do turno PL e **y** o número do grupo (e.g. ESR-TP2-PL33-Rel.pdf para o relatório TP3 do grupo 3 do PL3).