Trabalho Prático Nº2 – Serviço de transferência rápida e fiável de dados sobre UDP

Duração: 6 aulas (não consecutivas, ver calendário)

Motivação

Na Internet, transferência fiável é sinónimo de TCP, que é um serviço de transporte que oferece um leque de funcionalidades completo, incluindo multiplexagem das aplicações, controlo de conexão (início, troca de dados e fim), entrega ordenada, controlo de erros, controlo de fluxo e controlo de congestão. Este serviço é necessário porque na camada de rede não há garantias de entrega, podendo ocorrer, perdas, atrasos e trocas de ordem. Desenhar um serviço fiável sobre um canal de comunicação não fiável é uma das tarefas mais complexas das comunicações entre sistemas remotos. O que faz do TCP um dos protocolos mais complexos alguma vez desenhados. Hoje em dia, já com quase 50 anos de vida, aponta-se o dedo ao TCP no que concerne a desempenho (débito) e à forma como reage a qualquer falha da camada inferior de forma conservadora. Um simples link WiFi, cuja qualidade de sinal pode ser variável num curto intervalo de tempo, pode fazer aninhar uma conexão TCP.

Neste trabalho não se pretende substituir o TCP nem tão pouco redesenhá-lo, mas simplesmente implementar uma alternativa sobre UDP com objetivos pedagógicos de consolidação de conhecimentos sobre a camada de transporte. A comunicação deverá ser feita em unidades de dados que caibam dentro de um datagrama UDP e que sejam enviados e recebidos via Sockets UDP. Cada grupo deverá implementar um conjunto mínimo de funcionalidades que permita transferir um ficheiro de forma fiável de um sistema para outro. Sobre esse conjunto base de funcionalidades, poderão ser acrescentadas algumas funcionalidades adicionais, existentes ou não atualmente.

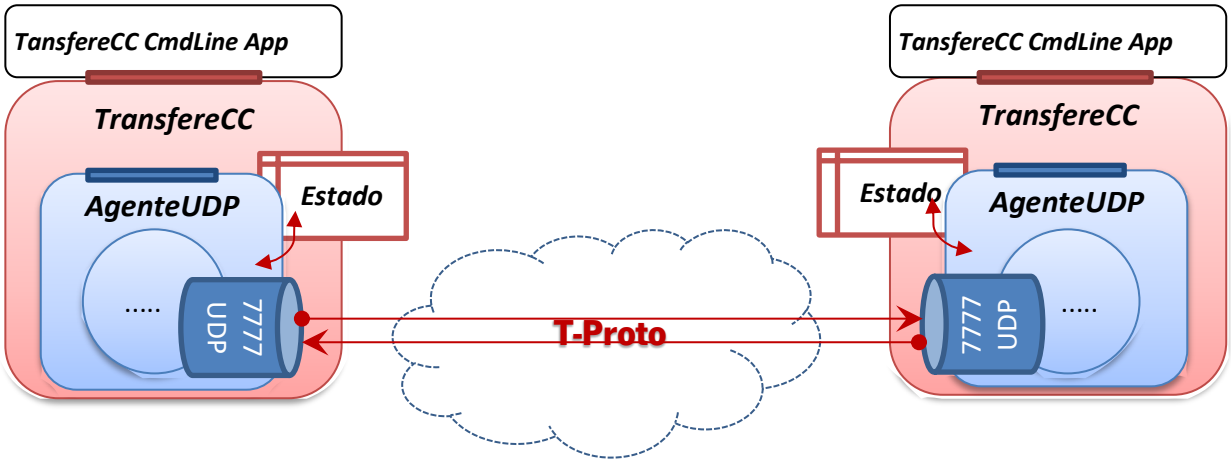

Figura 1: Transferência de ficheiros suportado por UDP

Objetivos

O objetivo deste trabalho é desenhar e implementar um serviço de transferência de ficheiros (dados), fiável, sobre uma conexão UDP genérica. A transferência pode ser em qualquer dos sentidos (*upload* ou *download*) e deve ter uma fase inicial (início de conexão) e uma fase de término (fim de conexão). Estas funcionalidades, designadas genericamente de controlo de conexão são obrigatórias. A transferência pode estar sujeita a perdas de pacotes e a receção desordenada, uma vez que o UDP que suporta a transferência não garante nem ordem nem entrega. É, pois, obrigatório que se defina um mecanismo de verificação de integridade (*checksum* ou outro), seguido de mecanismo de confirmação (ACK) e retransmissão se for caso disso (controlo de erros). Todas as outras funcionalidades (controlo de fluxo, segurança, controlo de congestão, múltiplas transferências em simultâneo, etc), são, para efeitos deste trabalho, consideradas opcionais. Cada grupo decide se implementa ou não alguma ou algumas delas, ou mesmo outras não elencadas neste enunciado, em função dos seus objetivos individuais tendo por base os critérios de avaliação abaixo definidos. Valoriza-se em termos de implementação a modularidade (módulos com funcionalidade e interface bem definido).

Descrição

A *Figura 1* mostra uma visão detalhada do sistema. A arquitetura descrita serve apenas de exemplo e de ponto de partida, não pretendendo ser vinculativa e muito menos única. Na arquitetura apresentam-se três componentes mais ou menos autocontidos: **1)** O agente UDP que comunica enviando e recebendo datagramas UDP e sobre o qual assenta toda a comunicação (**AgenteUDP**); **2)** A componente que implementa o transporte fiável (**TransfereCC**) e que terá de implementar as funcionalidades necessárias para que a transferência ocorra como pretendido; **3)** Uma tabela de estado, que para cada transferência tem alguma informação útil sobre a transferência e o estado da mesma, como por exemplo, ficheiro de dados, origem, destino, porta de origem, porta de destino, etc. (**Estado**); 4) uma aplicação simples de linha de comando que permite escrever comandos do tipo GET file ou PUT file (***TransfereCC CmdLine App***);

Esta estrutura assim definida procura dar destaque à modularidade e à separação de funções por módulos ou camadas. Ao **AgenteUDP** caberá apenas receber e enviar *datagramas*, de acordo com o tamanho e o formato que vier a ser definido para o PDU (*Protocol Data Unit*). A tabela de estado, por seu lado, tem por missão guardar num único local toda a informação que possa ser útil para controlar o estado da transferência. O módulo **TransfereCC** implementa o serviço de transporte fiável. E finalmente a aplicação de interface com o utilizador, que numa primeira fase de desenvolvimento pode perfeitamente ser substituído por um programa de testes com os pedidos de ficheiros *hardcoded*.

Funcionalidades base obrigatórias

- Controle de conexão, que permita às partes envolvidas iniciar, aceitar/recusar, terminar conexões;
- Identificar o ficheiro ou ficheiros a transferir, download ou upload, com nome local e opcionalmente um nome remoto;
- Garantir uma receção ordenada (caso os segmentos cheguem fora de ordem, deve ser possível ordená-los de forma correta);
- Controlo de erros, com verificação de integridade e retransmissão se necessário (retransmitir pacotes perdidos ou em erro);
  NOTA: é expectável que se procure o melhor desempenho possível e nesse sentido uma implementação segundo um algoritmo Stop-and-Wait é considerada demasiado naive para poder ser avaliada sem ser com nota mínima nesta componente;

Funcionalidades Adicionais (opcionais, pelo menos uma para poder passar o patamar médio)

- Segurança (por exemplo: autenticação das partes envolvidas – originador e destinatário, verificação de integridade no final da transferência com assinatura digital do conteúdo, confidencialidade); [Dificuldade: ***]
- Controlo de fluxo (débito controlado pelo destinatário); [Dificuldade **]
- Controlo de congestão (deteção de congestão na rede e mecanismos de ajustamento); [Dificuldade ***]
- Múltiplas conexões em simultâneo, numa e noutra direção (multiplexagem de várias transferências entre as mesmas entidades usando o mesmo socket); [Dificuldade ****]
- Transferência multiservidor (se o mesmo conteúdo estiver disponível em N servidores, poder obter os segmentos desses N servidores em paralelo, para aumentar o desempenho); [Dificuldade ****]
- Outras... (a propor pelo grupo; serão valorizadas pelo docente do turno de acordo com o seu critério);

Cenário de Teste

Pode-se usar como plataforma de teste o emulador CORE, embora tal não seja obrigatório, pois pode ser testado entre computadores dos membros do grupo. Os testes não se devem cingir apenas à própria máquina, de "*localhost*" para "*localhost*", embora isso possa até ser útil na fase de desenvolvimento. Recomenda-se dois tipos de testes: i) sem erros provocados, entre dois quaisquer nós da topologia (ou *hosts* físicos); ii) com erros provocados numa das ligações (perdas ou atrasos); Na topologia Core de CC 2019 (*CC-Topo-2019.imn*) os testes com erros podem ser feitos entre um dos sistemas da LAN3 (Alfa, Beta ou Gama) e outro sistema de outra LAN (por ex: Cliente1), uma vez que o link entre o Router4 e o SwitchLan3 já está configurado para gerar perdas da ordem dos 10%, atrasos de 5ms e 15% de pacotes duplicados. Estes parâmetros podem ser ajustados. Adicionalmente pode ser comparado o desempenho com o TCP ;-)

FASE 1: Conceção e desenho do protocolo

Espera-se que no final da fase 1 já possa ser feita uma transferência de dados de forma fiável.
Etapas propostas:

- Arquitetura da solução (módulos, interfaces, etc.);
- Proposta de PDU a ser enviado entre elementos ***AgenteUDP***;
- Primeira versão funcional do ***AgenteUDP***, capaz de enviar e receber os PDU definidos;
- Validação do PDU recebido;
- Controlo de conexão a funcionar;

Controlo de fluxo: não sobrecarregar receptor

FASE 2: implementação final com as funcionalidades pretendidas

Espera-se que no final da fase 2 a implementação final a funcionar e testada.
Etapas propostas:

- Implementação do controlo de erros (confirmações, retransmissões);
- Implementação de alguma funcionalidade adicional (opcional);
- Teste da implementação;

Planeamento

O desenho e implementação deste serviço devem ser feitos ao longo das aulas práticas de CC, quer nas explicitamente dedicadas ao trabalho, como em todas as outras onde existir tempo disponível. As fases de desenvolvimento estão devidamente calendarizadas no Planeamento das Aulas práticas (ver MIEI-CC-PLANEAMENTO-2019.pdf no E-Learning).

Relatório

O relatório deve ser escrito em formato de artigo com um máximo de 8 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 do desenho e implementação com a seguinte estrutura recomendada:

- Introdução;
- Especificação do protocolo;
  * formato das mensagens protocolares (PDU);
  * interações;
- Implementação;
  * estruturas dados, métodos/funções, parâmetros, bibliotecas de suporte usadas, etc.;
- Testes e resultados;
- Conclusões;

Regras de 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:

- **CC-TP2-PLxx-Rel.pdf** para o relatório;
- **CC-TP2-PLxx-Codigo.zip** ou **CC-TP2-PLxx-Codigo.rar** para a directoria com o código desenvolvido (devidamente documentado);

em que **xx** é o identificador de cada grupo (e.g. CC-TP2-PL21-Rel.pdf para o relatório TP2 do grupo PL21).