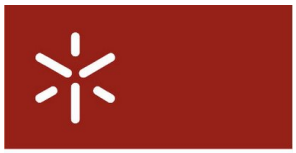

Universidade do Minho

# Comunicação por Computador

MIEI - 3º ANO - 2º SEMESTRE

UNIVERSIDADE DO MINHO

## TRABALHO PRÁTICO - 3

## UDP: DATA TRANSFER

António Gomes (a67645),
João Coutinho (a86272),
Aventino Menezes (a79187)

10 de Maio de 2019

# Conteúdo

# .1 Introdução

O UDP nasce para servir de interface entre a cmada IP e a camada de aplicação. O seu principal objectivo é agir como um protocolo simples e de comunicação rápida, reduzindo por isso ao máximo o tempo e latência da conexão.

Para efeitos de minimizar o tempo de coneção não há estabelecimento de conexão, isto é, no ato de envio de dados atravéz de UDP, os dados são enviados porém nenhuma confirmação da sua receção chega de volta permitindo assim reduzir o tempo de conexão, ou seja, o típico three-way-handshake que ocorre nas comunicações sobre TCP não existe em UDP. Contudo esta característica tem efeitos secundários permite reduzir o tempo de transferência de dados sobre pena de perder controlo sobre o estado atual dos dados enviados ao destinatário. Também, o UDP não guarda buffers de dados relativos a envio e receção de packets tendo como objectivo suportar um maior número de clientes ativos em simultâneo. Esta informação, contudo, tem de ser retida de alguma forma pelo que pacotes UDP sofrem de uma pequena sobrecarga de informação retida no seu cabeçalho.

De forma a evitar a aceitação de pacotes que chegam ao destino com erros surgidos durante o seu transporte, cada pacote possui um valor dedicado à validação dos dados nele contidos. A este valor chamamos de checksum. O Checksum é gerado na criação do pacote e verificado no seu destino de modo a verificar se os dados sofreram alguma alteraçao pelo caminho. Embora o UDP forneça verificação de erros, erros ocorridos apenas são detetados e não recuperados. Algumas implementações de UDP simplesmente descartam o segmento danificado, outras passam o segmento errado à aplicação acompanhado de algum aviso.

## .2 Motivações

Como visto anteriormente, o protocolo UDP visa a coneção símples e acima de tudo rápida tornando-se este protocolo adequado a qualquer tipo de serviço cuja minimação do tempo de coneção seja vital.

Aplicações do protocolo UDP podem ser, entre outras, transações relativas à bolsa de valores, comunicações real time como video-chamadas e live streaming. No contexto deste tipo de conexões, o tempo de conexão é vital pois potências atrasos na conexão podem causar a uma degradação da qualidade de serviço prestado entre outras piores consequências. No caso da realização de transações relativas à bolsa de valores, tempo é literalmente dinheiro e a diferença de milisegundos pode corresponder a ganhar ou perder vastas quantias monetárias.

Contudo todos os mecanismoms tomados pelo UDP de forma a garantir a rápida transferência dos dados causam problemas no que toca à monitorização da conexão, autenticação dos dados e hosts entre outros problemas.

O senário ideal sería por isso juntar o melhor de ambos os protocolos, isto é o controlo e segurança de TCP com a rapidez de UDP porém estas realidades tendem a se comportarem como sendo inversamente proporiconais. Com isto tentaremos desenvolver um pequeno e símples protocolo de coneção baseado em UDP possuindo algumas estruturas e características adicionais com objectivo de garantir mais segurança e fiabilidade da conexão tentando não perder a velocidade da conexão.

## .3 Funcionamento geral da aplicação

Como plataforma de teste da conexão foi desenvolvida uma aplicação em linguagem JAVA que permite a transferência de ficheiros de texto entre clientes e um servidor. A conexão entre ambos os extremos é efecutada com base em sockets UDP (datagram sockets) que já existem definidos em JAVA. A aplicação cliente liga-se ao servidor imediatamente ao iniciar e esta pode realizar o upload de ficheiros contidos na máquina do cliente para o servidor assim como realizar o download de ficheiros contidos no servidor. A aplicação cliente-servidor pode ser vista como uma espécie de dropbox de ficheiros de texto.

O cliente é responsável pelo inicio de conexão e fecho do socket existente entre cliente e servidor. O servidor tem apenas um papel de reação aos inputs do cliente podendo por isso funcionar de forma independente após iniciado. Isto é um mecanismo importante em qualquer aplicação servidor pois permite o serviço ser fornecido aos clientes sem a necessidade de um interveniente humano poupando assim ao sevidor o salário de um funcionário adicional. A interação humana com o servidor deve-se apenas às necessidades naturais de manutenção de qualquer servidor.

A interface de interação do utilizador para com a aplicação cliente é o terminal, sendo fornecido ao cliente apenas os ficheiros .class de forma a este ter acesso à aplicação sem possuir acesso ao código fonte. A comunicação pode ser efectuada remontamente entre clientes e servidor em redes diferentes, porém para simplificar o teste a coneção é feita por default para servidor dentro da mesma rede que o cliente. O sevidor possui a capacidade de se ligar com mais do que um cliente em simultâneo como será visto em secções seguintes.

## .4 Conexão cliente - servidor

Como mencionado anteriormente, a coneção entre clientes e o servidor é realizada com base em datagram sockets de JAVA. Cada cliente conecta-se ao servidor no ato de inicialização da aplicação e o servidor préviamente ligado cria um thread para cada novo cliente que se conecta, permitindo assim garantir que o blqueio de um cliente não impede o servidor de continuar a servir outros clientes.

No ato de coneção, um idênctificador é atribuido ao cliente que servirá como a sua ficha pessoal de autênticação. Este mecanismo visa a monitorizar a fiabilidade dos pacotes transferidos entre os pontos garantido assim que não haja uma terceira entidade a adulterar de alguma forma a coneção. O servidor verifica se idêntificador de origem de cada pacote é válido.

Para efeitos da aplicação, um pacote UDP é criado inicialmente sobre a forma de um objecto de dados JAVA chamado PDUnit que possui os dados do header necessários à conexão e os dados própriamente ditos.

## .5 Escalonamento e fragmentação de pacotes

A dimensão de um pacote UDP não é infinita e existe um limite máximo para a dimensão do mesmo, contudo, por vezes os ficheiros que se pretende enviar ou receber excedem a dimensão máxima destes pacotes pelo que necessitam de ser sub-divididos em mútliplas secções de tamanho reduzido de modo a poderem ser enviadas para o outro extremo da coneção e montadas novamente no destino.

Cada parágrafo de um ficheiro de texto é dividido e dá origem a um pacote individual que, cada um, armazena um parágrafo em string assim como o nome do ficheiro correspondente. Cada parágrafo é transposto para um PDUnit de modo sequêncial e recebe um idêntificador também sequêncial, isto é, o primeiro parágrafo dá origem a um pacote com o idêntificador id que servirá como um idêntificador original do documento a ser enviado, o segundo parágrafo dá origem a um segundo pacote de idêntificador id+offset sendo offset o número de pacotes já gerados até ao momento.

O número total de pacotes PDUnit que compõem um documento é conhecido a priori permitindo assim ao destinatário saber quantos pacotes é suposto possuir relativos a um documento para este estar completo podendo assim regenerar o documento assim que todos os pacotes chegam. Os pacotes que vão sendo recebidos pelo destinatátrio são armazenados numa ArrayList e cada vez que um novo pacote chega este é validado e introduzido na estrutura contida num objecto TransferCC que age como api um buffer de receção e o seu estado é analizado. Caso todos os pacotes esperados já tenham chegado o documento é regenerado e armazenado. O pacote de menor índice corresponde ao primeiro parágrafo e assim succesivamente.

## .6 Pacotes em circulação num socket

Como mencionado em secções anteriores, o protocolo UDP não emite relatórios de receção de cada pacote que recebe pelo que torna a monitorização da conexão mais dificil. Na aplicação, as mensagens de confirmação de receção (acknowledgments) são enviadas apenas apoós todos os pacotes de um documento terem sido recebidos, isto é, após o documento ter sido transferido na sua totalidade com successo. Com isto reduz-se a transmissão de pacotes na rede, melhorando por isso o tempo de serviço, e mantém-se alguma monitorização da conexão.

Numa normal conexão podemos constatar que pacotes de diversas naturezas são transferidos pelo que não é necessário que todos os pacotes sejam tratados da mesma forma. Para isto pacotes PDUnit podem ser de 3 tipos com 3 prioridades diferentes. Um pacote do tipo 0 é um pacote de acknowledgment e a sua prioridade é 1 sendo por isto média. pacotes que contém parágrafos de um documento são pacotes de upload se estiverem a ser enviados ou download seu estiverem a ser recebidos e são de tipo 1 de prioridade 2 que é máxima. Pacotes de pedido de dados (Data Request) são pacotes do tipo 1 de prioridade 1 que é intermédia. O inicio e término de conexão é realizado em pacotes de tipo 0 com prioridade 0 que é mínima. Com isto é possível garantir que a transmissão dos dados é na mesma rápida sem que as mensagens de controlo interfiram no tempo de serviço.

Os pacotes são enviados sequêncialmente ao seu destinatário sendo que não se pode inferir a sua ordem de chegada, pelo que estes têm de ser armazenados no destinatário e o documento apenas pode ser regenerado assim que todos os pacotes tenham sido recebidos.

## .7 Deteção e tratamento de erros

A transferência de dados está sujeita a danos que possam ser causados no decorrer em algum nodo no caminho entre origem e destino. Para efeito de garantir a fiabilidade dos dados armazenados e transferidos, o recetor tem de ser capaz de verificar potênciais erros e lidar com estes adequadamente.

Erros podem surgir de diversas formas. Os dados podem sofrer degeneração pelo caminho, pacotes podem ser descartados por algum router algures entre origem e destino da comunicação e não chegarem ao destino entre outros problemas que podem surgir.

Quando existem pacotes em falta o recetor está responsável por enviar um data request ao extremo oposto da conexão solicitando um novo envio dos dados em falta. Para que a origem possa saber quais dados a enviar, o data re-request é um pacote de data request cujo seu id é igual ao id do pacote/pacotes em falta que pode ser determinado pois o número de pacotes total que compõem um documento e o ID do pacote original está contido em cada PDUnit daquele documento. O parágrafo em falta será determinado por ID - ID original e o nome do documento em questão está em filename.

Caso um data request seja enviado mas nenhum pacote de dados chegue de volta o data request é enviado de novo após um tempo definido até se poder constatar com algum grau de certeza que a conexão não está mais garantida.

De modo a validar os dados de um PDUnit no ato da sua receção, o seu checksum é verificado e comparado com os demais checksums dos PDUnis correspondetes dos restantes PDUnits do mesmo documento. Este método não permite verificar a validação dos dados de pacotes unitários contudo apenas pacotes de transferência de dados possuem dados para serem verificados e estúdos probabilísticos indicam que a chance de um documento de um parágrafo único ser enviado é de tal maneira reduzida que pode ser descartada com algum grau de certeza.

Caso um parágrafo tenha dimensão suficientemente grande para ultrapassar o limite de dados do socket, este pode ser dividido também e a sua montagem ocorre de forma igual.

## .8 Segurança na comunicação

Na comunicação entre dois pontos nada se sabe sobre o que ocorre no meio de comunicação pelo que é necessário garantir que os dados recebidos são correspondentes aos dados enviados pela origem. Agentes terceiros podem afetar a comunicação tirando partido da mesma de forma prejudicial a algum ou ambos os extremos da comunicação.

Cada cliente, ao se conectar com o servidor, recebe um idêntificador que terá de ser parte de cada pacote que este envia. O servidor também possui um idêntificador que é gerado aleatóriamente assim que este é iniciado (de modo a permitir que este seja alterado caso algum agente malicioso consiga detetar o id do servidor). Um pacote ao ser recebido, a sua origem é detetada tendo de possuir o idêntificador do cliente correspondente de modo a poder ser garantida como fiável.

Mecanismos de segurança adicionais poderíam ser adicionados porém cada novo nível de segurança acrescenta à complexiadade dos PDUnits e adiciona tempo de processamento da comunicação pelo que um grau de segurança mínimo é tomado de froma a poder garantir alguma fiabilidade da comunicação minimizando o seu efeito no tempo de comunicação.

## .9 Conclusões

Uma abordagem intermédia aos protocolos de TCP e UDP é necessária de modo a permitir comunicação rápida e segura sendo os efeitos de um sejam minimizados na tarefa do outro.

Com isto pudemos criar um pequeno e simples protocolo de dados que visa não a substituir UDP ou TCP mas sim fornecer uma comunicação aceitvelmente rápida e com algum grau de segurança e fiabilidade do serviço prestado tendo em conta alguns dos aspectos mais importantes de uma conexão segura e fiável e tentando minimizar o efeito das estruturas e mecanismos de validação no tempo de comunicação.