CENTRO 2030 Os Fundos Europeus mais próximos de si. PORTUGAL 2030 Cofinanciado pela União Europeia
estga universidade de aveiro

Fundamentos de Redes de Comunicação

CTESP – Redes e Sistemas Informáticos

5 – Transporte

António Godinho

1

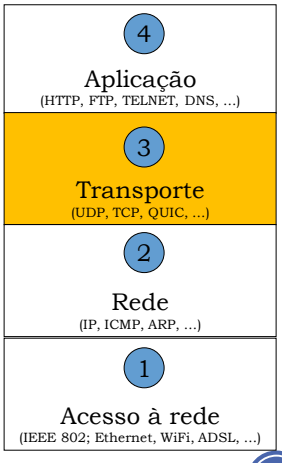
Transporte

Terminado o estudo do nível Rede do modelo TCP/IP vamos passar à **camada de transporte**.

O protocolo IP (Camada de Rede) tem algumas características que o tornam "incompleto" para algumas aplicações:

- Não garante a entrega dos pacotes
- Não solicita a retransmissão em caso de perdas
- Não implementa controlo de fluxo em caso de congestionamento.

É função da Camada de Transporte implementar estas funções. Os principais protocolos desta camada são o **UDP** e **TCP**.



2

Transporte – Portos e Sockets

Na comunicação entre dois processos há a necessidade de os identificar pois cada máquina pode estar a correr vários processos em simultâneo

Exemplo: se estivermos a receber um e-mail e a ouvir rádio online, a nossa máquina recebe pacotes destinados a cada um destes processos. Se apenas estivessem identificados pelo nosso endereço IP, a distinção entre eles não era possível

Surge então o conceito de **porto** e **socket**: permitem a identificação unívoca do canal lógico de comunicação estabelecido entre dois processos que estejam em comunicação

Transporte – Portos e Sockets

Cada processo que pretenda comunicar com outro deve identificar-se na arquitetura TCP/IP por um ou mais portos (ou portas):

- **Porto** (ou porta): número de 16 bits - valores entre 0 e 65535; Todos os pacotes trocados entre dois sistemas têm de identificar o endereço IP de destino e o porto de destino.
- O conjunto formado pelo Endereço IP + porto é o **socket**
- Os portos podem ser Bem-Conhecidos/well-know ou Efémeros/cliente

Portos Bem-conhecidos ou "well-know ports" [1 a 1023] (geridos e atribuídos pela IANA) - algumas aplicações utilizam sempre o mesmo porto. Alguns exemplos na tabela seguinte:

Portos bem conhecidos

Porta	TCP ou UDP	Nome do protocolo ou serviço	Nome do serviço	Usado por/informações adicionais
7	TCP/UDP	echo	echo	ping, traceroute
20	TCP	File Transport Protocol (FTP)	ftp-data	-
21	TCP	Controle de FTP	ftp	-
22	TCP	Secure Shell (SSH)	ssh	Acesso Remoto
23	TCP	Telnet	telnet	-
25	TCP	Simple Mail Transfer Protocol (Protocolo simples de transferência de e-mails - SMTP)	smtp	Mail (para enviar e-mail)
53	TCP/UDP	Sistema de nomes de domínio (DNS)	domínio	
67	TCP	File Transfer Protocol (FTP)	ftp	
68	TCP	File Transfer Protocol (FTP)	ftp	
69	UDP	Trivial File Transfer Protocol (TFTP)	tftp	
80	TCP	Hypertext Transfer Protocol (HTTP)	http	
110	TCP	Post Office Protocol (POP3)	pop3	Mail (para receber e-mail)
143	TCP	Internet Message Access Protocol (IMAP)	imap	Mail (para receber e-mail)
443	TCP	Secure Sockets Layer (SSL ou "HTTPS")	https	
993	TCP	SSL para IMAP no Mail	imaps	
995	TCP/UDP	SSL para POP no Mail	pop3s	-

5

Transporte – Portos Efémeros

Portos Efémeros/cliente : [1024, 65535]

Portos não alocados a uma aplicação ou serviço específico e que podem ser utilizados pelos clientes que iniciam a comunicação com o servidor.

Tipicamente são escolhidos pelo sistema operativo no início de uma comunicação.

Por exemplo, se desenvolverem uma nova aplicação poderão escolher um destes portos para a comunicação com o respetivo servidor.

Uma vez que ao nível do pacote IP se faz a distinção entre os protocolos de transporte, pode-se usar o mesmo número de porto para dois processos distintos, um que utilize o UDP e outro que utilize o TCP

6

Protocolos de Transporte

A camada de transporte tem dois principais protocolos: **UDP** e **TCP**

	UDP	TCP
Ligação	Connectionless – Ausência de ligação	Modo de ligação – exige estabelecimento de ligação prévia à troca de dados
Consumo de recursos	Mais leve	Mais complexo (pesado)
Controlo de fluxo	Não	Sim
Controlo de congestionamento	Não	Sim
Garante Fiabilidade?	Não	Sim (os pacotes chegam todos ao destino, por ordem e sem duplicações)
Nome dos pacotes	Datagramas	Segmentos
Broadcast	Sim	Não

Protocolo UDP

User Datagram Protocol

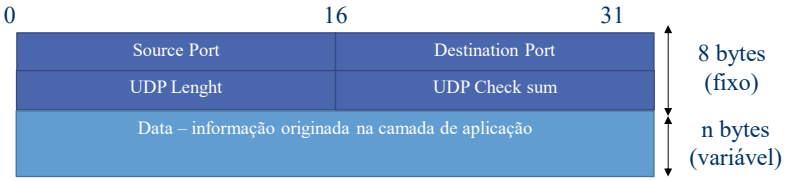
Fornece um serviço não orientado à conexão (Connectionless)

Não fiável (não há garantia de entrega do datagrama)

Simples (não introduz muito overhead na rede): Apenas acrescenta ao pacote IP a identificação do porto do processo de aplicação remetente e do porto do processo de aplicação destinatário (porto origem e porto destino)

Permite comunicação unicast, broadcast, multicast

Formato do datagrama UDP



- **Source Port (16b)**: número do porto que identifica o processo de aplicação remetente
- **Destination port (16b)**: número do porto que identifica o processo de aplicação destinatário (remoto)
- **UDP length (16b)**: Nº de bytes de todo o datagrama
- **UDP checksum (16b)**: (opcional) verifica integridade de todo o datagrama - cabeçalho+dados+pseudo cabeçalho (IP Origem, IP Destino, Protocolo, UDP lenght)
- **Data**: dados da aplicação a transportar (recebido da camada de aplicação)

Protocolo UDP

Quando uma comunicação utiliza o protocolo de transporte UDP, o UDP apenas identifica as aplicações que correm numa máquina através dos sockets (conjunto de endereço IP e porto).

- Não há contato prévio entre emissor e recetor para confirmar a disponibilidade de comunicação
- Os pacotes são enviados e o UDP não dará a indicação se chegaram ou não
- Se os pacotes chegarem fora de ordem, o UDP também não tem forma de identificá-los para poderem ser ordenados
- Se for necessário garantir entrega de pacotes e a sua ordem, terá de ser a aplicação a implementar algum mecanismo de controlo

Exemplos de protocolos de aplicação que usam UDP:

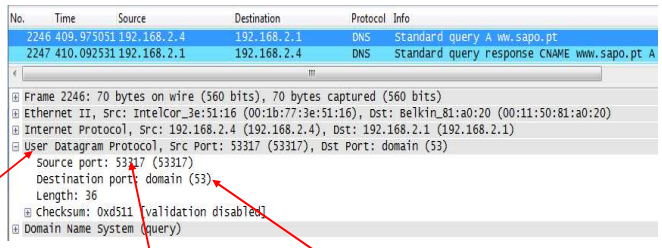
Domain Name System (DNS), Network File System (NFS), Simple Network Management Protocol (SNMP), Lightweight Directory Access Protocol (LDAP), NetBIOS, Trivial File Transfer Protocol (TFTP)

Protocolo UDP

Exemplo de uma aplicação que usa UDP: DNS

Com o wireshark ativo executem o seguinte comando: *nslookup www.sapo.pt*

Analisem o primeiro pacote do protocolo DNS:



Deverão verificar:

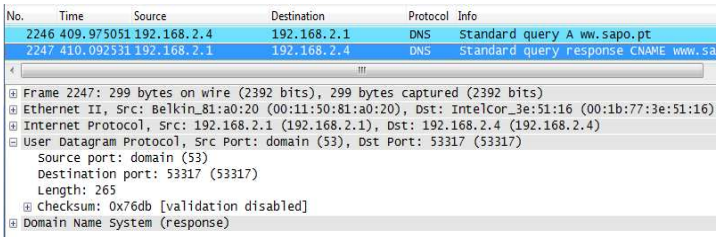
- Utilização do UDP (User Datagram Protocol)
- Não há estabelecimento de sessão (o pedido é enviado sem contacto prévio com o destino)
- O pedido é enviado de um porto efémero (no caso acima foi o 53317) para um well-known port (o porto 53 está reservado para a aplicação DNS)

11

Protocolo UDP

Exemplo de uma aplicação que usa UDP: DNS

O segundo pacote DNS é a resposta do servidor de DNS ao nosso pedido:



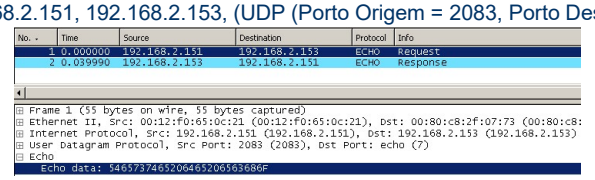
- Continua a ser via UDP
- O porto de origem é o 53, well-know port do DNS
- O porto de destino é o 53317 que, neste exemplo, foi o porto efémero que o nosso computador utilizou para identificar o pedido.

12

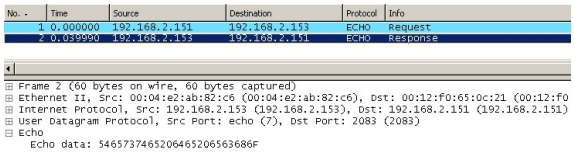
Protocolo UDP

Outro exemplo de uma aplicação com UDP: Echo (Ping), Porta Well-known 7 UDP

IP(192.168.2.151, 192.168.2.153, (UDP (Porto Origem = 2083, Porto Destino = 7)))



IP(192.168.2.153, 192.168.2.151, (UDP (Porto Origem = 7, Porto Destino = 2083)))



Protocolo TCP

Transmission Control Protocol

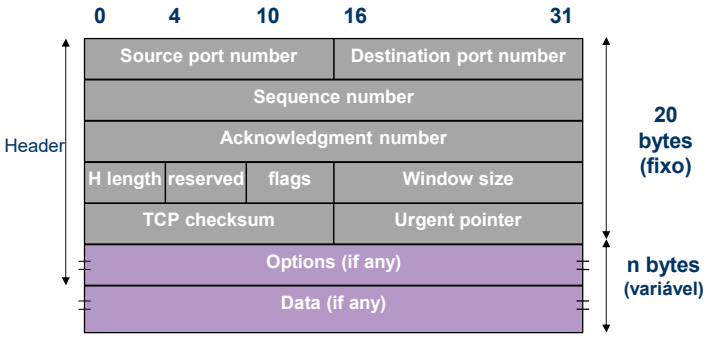
- Orientado à Ligação:
  - Necessidade de estabelecer uma ligação prévia antes do envio de dados. Confirma a disponibilidade do destino e evita o envio de pacotes se não houver forma de comunicar;
  - No fim da transmissão deve ser encerrada a ligação.
- Fiável:
  - Garante que os dados chegam ao destino e identifica a ordem pela qual devem ser colocados na aplicação de destino
  - Necessita de um mecanismo de confirmação de receção e de numeração dos dados
  - Faz o Controlo de erros

Protocolo TCP

Transmission Control Protocol

- Controlo do Fluxo:
  - Ambos os extremos do fluxo de dados da ligação utilizam mecanismos de controlo de fluxo
  - Adequação entre débitos de envio e de receção
- Controlo de Congestionamento:
  - Adequação às condições de débito do canal de comunicação
- Full Duplex:
  - Permite o fluxo concorrente de dados em ambos os sentidos da conexão

Segmento TCP

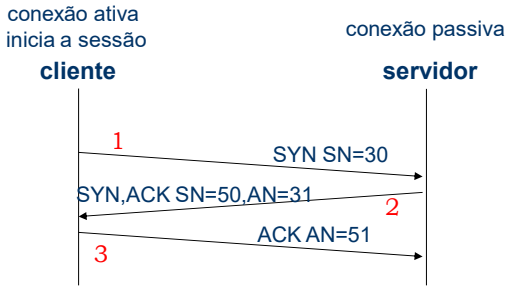


- *Source port* (16b) e *Destination port* (16b): números dos porto origem/destino
- SN – *Sequence number* (32b): identifica o número de sequência do primeiro byte do segmento de dados que está a ser enviado
- AN – *Acknowledgement number* (32b): contém o próximo número de sequência do byte que o remetente espera receber.
- *Window size* (16b): número de bytes que o remetente está disponível a receber no próximo segmento (utilizado para o controlo de fluxo)
- Data: dados de aplicação a transportar

TCP – Início de Sessão

Antes de qualquer troca de informação numa ligação com TCP, os dois sistemas têm de previamente estabelecer uma ligação.

É feito através do método **three way handshake**:



1 - O cliente envia um segmento SYN com o número de sequência inicial do cliente (ISNcli = SN=30 neste exemplo)

2 - O servidor responde com o seu segmento SYN, com o número de sequência inicial do servidor (ISNsrv = SN = 50, neste exemplo) e, ao mesmo tempo, confirma a receção do segmento SYN cliente ativando a flag ACK e colocando no campo número de confirmação (AN) o valor de ISNcli+1 = 31

3 - O cliente envia confirmação num segmento ACK com AN=ISNsrv+1 = 51

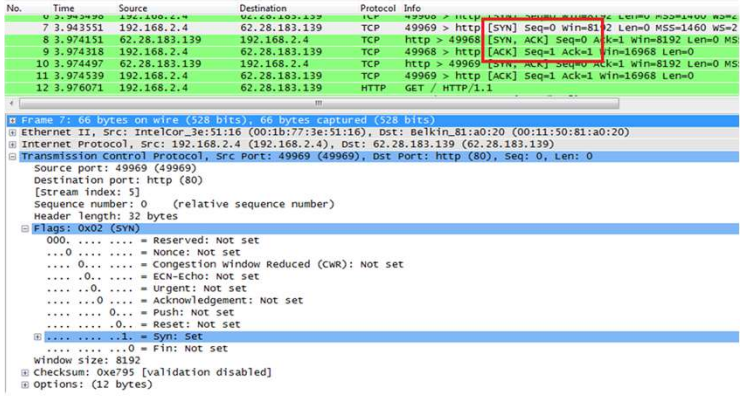
17

TCP – Início de Sessão

Exemplo de um serviço que usa TCP: HTTP

Com o wireshark ativo, acedam a uma página web no browser.

Antes de ser enviado o pedido é feito o three-way-handshake com o servidor onde está a página:

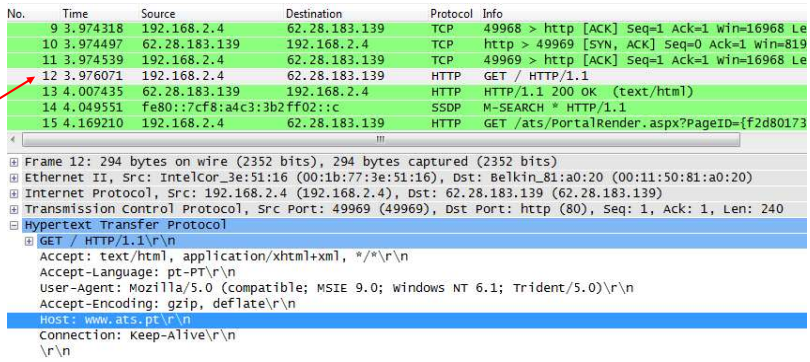


18

TCP – Início de Sessão

Só depois do three-way-handshake é que o vosso pedido é enviado ao servidor (neste caso, protocolo HTTP).

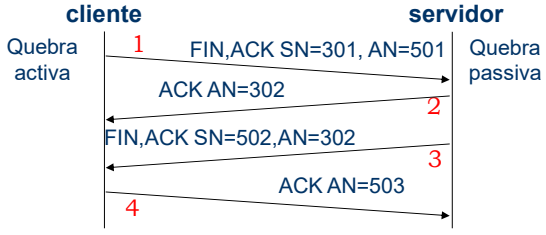
O conteúdo da página Web é enviado também em TCP.



19

TCP – Fecho de Sessão

Quando um sistema já não tem dados para enviar faz o half-close, enviando um segmento FIN:



Como a ligação é full-duplex, cada uma das partes necessita de fazer um half-close

Protocolo:

1- O cliente envia um segmento FIN com o seu número de sequência (SNcli) atual e com o respetivo número de confirmação

2 -O servidor envia um segmento ACK com AN =SNcli+1 (302)

3 -O servidor envia um segmento FIN com o seu número de sequência (SNsrv) atual e com o respetivo número de confirmação (SN=502, AN = 302)

4 - O cliente envia um segmento ACK com AN = SNsrv+1 (503)

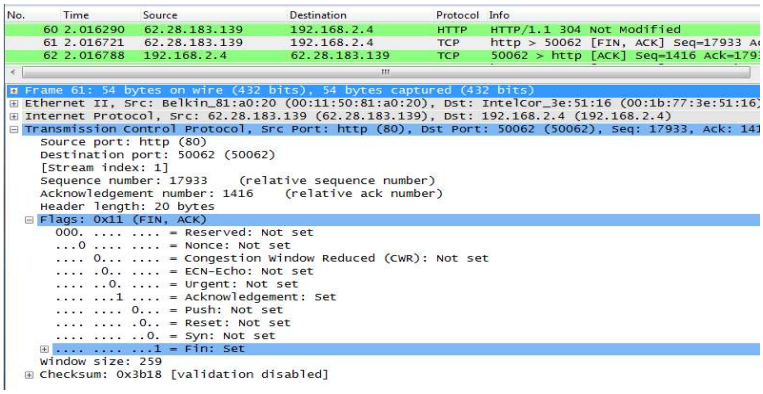
Entre os pontos 2 e 3, o servidor ainda pode enviar dados.

20

TCP – Fecho de Sessão

Exemplo de um serviço que usa TCP: HTTP

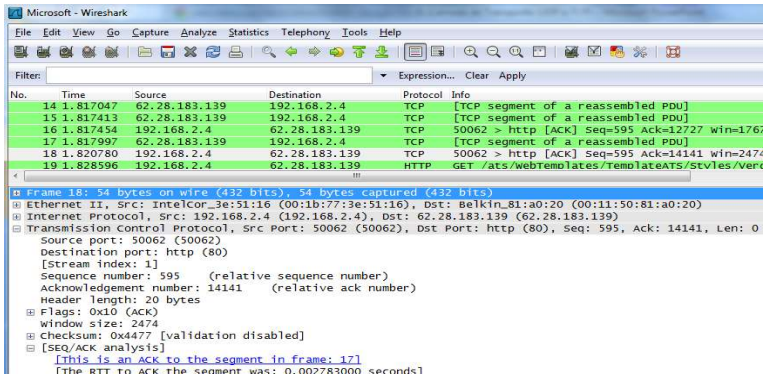
Depois da página Web ser enviada e recebida (às vezes são milhares de pacotes…) o servidor pede o fecho da sessão, o que significa que não tem mais nada a enviar:



TCP – Fiabilidade

Por cada pacote TCP enviado, o recetor tem de confirmar a sua chegada através de um ACK!

Desta forma, o emissor sabe quais os pacotes que foram entregues.



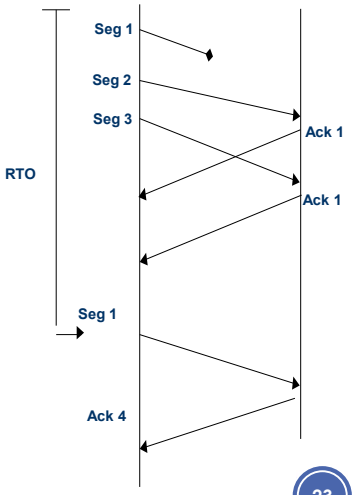
TCP – Controlo de Fluxo

O TCP coloca um **número de sequência (SN)** em todos os pacotes: permite ao recetor saber a ordem pelo qual deverão ser entregues à aplicação de destino.

O recetor tem de confirmar a chegada de cada pacote através de uma mensagem ACK.

No emissor, é ativado um **RTO - Retransmission Timeout** para cada segmento enviado.

Se não existir confirmação da receção (ACK) no período de tempo RTO, o emissor assume que o segmento se perdeu e é retransmitido (ver imagem).

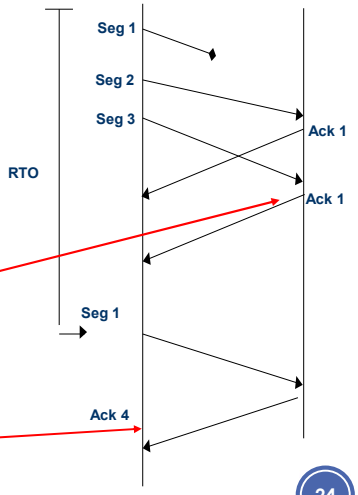


23

TCP – Controlo de Fluxo

Os Segmentos podem chegar fora de ordem pois são transportados por pacotes IP e seguir caminhos diferentes ou mesmo serem descartados pelos routers.

- Os ACK de confirmação de receção de segmento apontam para o primeiro segmento em falta (Imagem: recetor envia Ack1 mesmo quando o 2 e 3 já chegaram pois ainda não recebeu o 1). A receção de confirmações duplicadas indicia a ocorrência de problemas na transmissão.
- Um segmento de confirmação valida a correta receção de todos os segmentos enviados anteriormente (Imagem: o ACK 4 confirma que o 2 e 3 já foram recebidos).



24

TCP – Controlo de congestionamento

Objetivos do **controlo de congestionamento**:

- Maximizar o débito
- Adaptação dinâmica às condições de congestionamento
- Aumento da eficiência

O TCP utiliza o tempo que decorre entre o envio de um segmento e a receção do respetivo ACK para perceber se a rede está livre ou congestionada.

O recetor também pode enviar mensagens TCP ao emissor a informar que está sem capacidade de receber mais segmentos (**Window Size = Zero**).

Quando o tempo fica muito elevado, o TCP reduz o ritmo de envio de segmentos.

Este processo é contínuo, ajustando-se durante toda a sessão.

25

TCP vs UDP - Aplicações

Exemplos de **aplicações que usam UDP**: VoIP, Streaming de Vídeo ou de áudio, P2P e DNS

Preferencialmente as que têm mecanismos próprios de controlo da perda de pacotes e que privilegiam uma rede rápida, mesmo que com perda de alguns pacotes, ou seja, precisam de toda a velocidade que a rede possa dar (p.e. o vídeo em IP utiliza MPEG que disfarça as perdas de pacotes).

Também as que necessitam de transmissão para múltiplos recetores, tanto em multicast ou broadcast.

26

TCP vs UDP - Aplicações

Exemplos de **aplicações que utilizam o TCP**: E-mail, Transmissão de ficheiros de forma segura (FTP) ou Acesso remoto a máquinas (SSH ou telnet)

As que necessitam de um transporte de pacotes fiável e garantido mesmo que à custa de velocidade de transmissão.

Quando projetamos uma aplicação em IP temos de tomar uma **decisão acerca do protocolo de transporte**:

- Ou construímos uma aplicação que seja por si só capaz de detetar e colmatar as falhas da rede – usamos UDP
- Ou construímos uma aplicação sem esses mecanismos mas que depende da fiabilidade da rede para garantir a entrega de toda a informação mesmo que tenha de esperar por ela – usamos TCP

27

Protoloco QUIC

Evolução recente nas funções de controlo de sessão. Desenvolvido pela Google para reduzir a latência nas transmissões de média.

- Suportado em UDP para otimizar a velocidade: não há handshake, por exemplo
- Complementa o UDP com mecanismos avançados de deteção e recuperação de perdas de pacotes (em vez de recorrer ao TCP)
- Mantém a sessão quando se alterna de rede (por exemplo, quando se muda de uma rede wifi para outra)
- Melhor segurança: criptografia intergrada
- Outras vantagens:
  - Melhor desempenho na consulta a páginas web HTTP.
  - Maior eficiência para serviços como streaming e jogos online.

A maior parte das páginas Web passou a utilizar este protocolo.

28