N_COM_DAD_A4 - Texto de apoio

Índice

1. Introdução

2. Introdução e serviços da camada de transporte

3. Comunicação entre processos

4. User Datagram Protocol – UDP

5. Transmission Control Protocol – TCP

6. REFERENCIAS

1. Introdução

Entender a camada de transporte e a influência de seus protocolos nas aplicações de rede não é interesse somente dos profissionais ligados à infraestrutura de redes de computadores, mas de todos os profissionais de computação, incluindo os programadores.

Para desenvolver aplicações distribuídas e prover um serviço na Internet, será necessário habilidades de programação na linguagem de desenvolvimento, entender como sua aplicação interagirá com a rede e, principalmente, será importante entender os diferentes serviços oferecidos pela camada de transporte, bem como de que maneira esses serviços impactarão na aplicação.

Para iniciar nossos estudos sobre a camada de transporte, trabalharemos com o conceito de comunicação entre processos para, posteriormente, discutir sobre os dois principais protocolos da camada de Transporte, o Transmission Control Protocol (TCP) e o User Datagram Protocol (UDP).

2. Introdução e serviços da camada de transporte

Localizada entre a camada de Aplicação e a camada de Rede do modelo OSI, a camada de transporte dispõe de mecanismos que possibilitam a troca de dados fim-a-fim, ou seja, ela atua como uma ligação entre um processo em execução no cliente e um processo execução no servidor, portanto, uma conexão de processo para processo (Forouzan, 2010).

Quando uma mensagem é enviada para um destino qualquer, a camada de transporte é responsável por receber uma mensagem da camada de Aplicação, segmentar essa mensagem em pedaços menores, adicionar um cabeçalho a cada segmento e encaminhar os segmentos para a camada de Rede para serem transmitidos pela rede. Quando esses segmentos chegarem ao destino, a camada de Transporte do destino irá reconstruir a mensagem e a encaminhar para a camada de Aplicação.

A camada de Transporte da arquitetura TCP/IP poder oferecer um serviço de transporte confiável e orientado à conexão, por meio do protocolo (Transmission Control Protocol), bem como um serviço não confiável e não orientado à conexão, por meio do protocolo UDP (User Datagram Protocol).

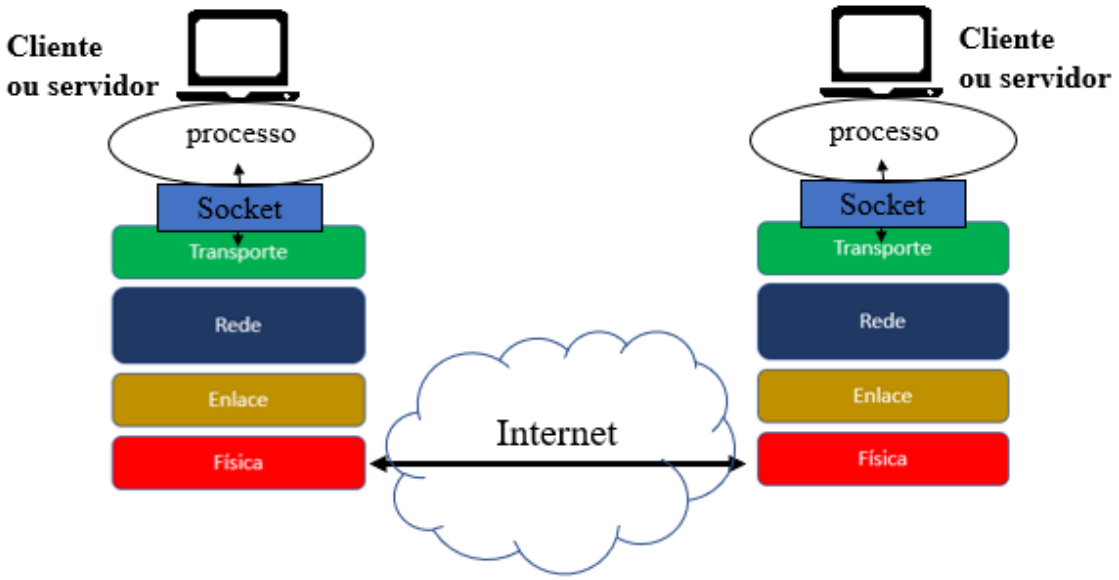
Detalhes sobre a comunicação entre processos e algumas características dos protocolos TCP e UDP serão discutidos nos tópicos subsequentes.

3. Comunicação entre processos

Segundo Silberschatz (2015), um processo é um programa em execução, incluindo todos os recursos necessários para sua execução (contator de programa, conteúdo de registradores etc.). Assim como processos distintos sendo executados em um mesmo sistema computacional podem compartilhar informações pela comunicação entre processos (Inter-Process Communication – IPC), as comunicações entre processos rodando em sistemas finais diferentes podem se comunicar pelo uso de Sockets.

Lançados na distribuição UNIX 4.2BSD de Berkeley em 1983 (Tanenbaum, 2013), o Socket é uma API (Application Program Interface) disponibilizada pelos sistemas operacionais que permite criar uma interface entre o processo e a camada de Transporte do modelo TCP/IP conforme representado no diagrama da figura 1.

Figura 1 – Representação do Socket, API que conectas processos remotos



Na Figura 1, podemos observar dois hosts conectados por meio de uma infraestrutura de rede. Nessa representação, o processo remetente enviará uma mensagem para o Socket que conectará esse processo a um protocolo específico da camada de transporte. Após segmentar e adicionar os cabeçalhos, a camada de transporte encaminhará os segmentos para a camada de rede.

A camada de rede será responsável por adicionar as informações de endereçamento e encaminhar essa mensagem até o computador de destino. Ao chegar no destino, a mensagem será encaminhada para a camada de transporte e direcionada pelo Socket ao processo a que ela foi endereçada.

Atualmente, os sistemas operacionais suportam tanto ambientes multiusuário como de multiprogramação (Forouzan, 2010), logo, para um processo estabelecer uma conexão lógica com outro processo remoto, faz-se necessário identificar ambos os processos, permitindo assim o encaminhamento correto das mensagens. Essa identificação dos processos é realizada por meio de identificadores chamados números de porta.

O uso de portas introduz em nosso processo de comunicação o serviço conhecido como multiplexação e demultiplexação. A multiplexação é caracterizada pela tarefa do host de origem em encapsular mensagens oriundas de diferentes aplicações (consequentemente, diferentes portas) com informações de cabeçalho que serão tratadas no destino da mensagem. O processo de demultiplexação, por sua vez, é a entrega do segmento para a porta correta a partir da análise do cabeçalho da camada de transporte.

Na pilha de protocolos TCP/IP o número de porta é um número de 16 bits representados por números inteiros entre 0 e 65.535 (16 bits). Quando um programa-cliente é executado, em geral, o sistema operacional atribui dinamicamente um número de porta (superior a 1023) a esse processo.

Assim como acontece com o endereço IP do servidor (assunto já discutido quando tratamos do paradigma cliente-servidor), a porta do processo que está executando no servidor deverá ser fixa. Com intuito de padronizar processos e portas, a IANA (organização mundial que supervisiona a atribuição global dos números na Internet) decidiu associar números de porta universais para os processos-servidor de aplicações (protocolos) tradicionais da Internet (Kurose, 2013). Na Tabela 1, temos uma relação de aplicações tradicionais da internet, seus protocolos e suas respectivas portas.

Porta	Descrição
22	SSH (Secure Shell) – Shell seguro para acesso remoto
23	Telnet – Comunicação de texto sem encriptação
53	DNS (Domain Name System) – Resolução de nomes
80	HTTP (HyperText Transfer Protocol) transferência de páginas Web
179	BGP (Border Gateway Protocol) – Protocolo de roteamento
389	LDAP (Lightweight Directory Access Protocol) – Protocolo de acesso a diretórios distribuídos
443	HTTPS – Protocolo HTTP sobre TLS/SSL (camada de segurança) oferecendo confidencialidade às aplicações
995	POP3 sobre SSL – Recebimento de e-mail, garantindo confidencialidade

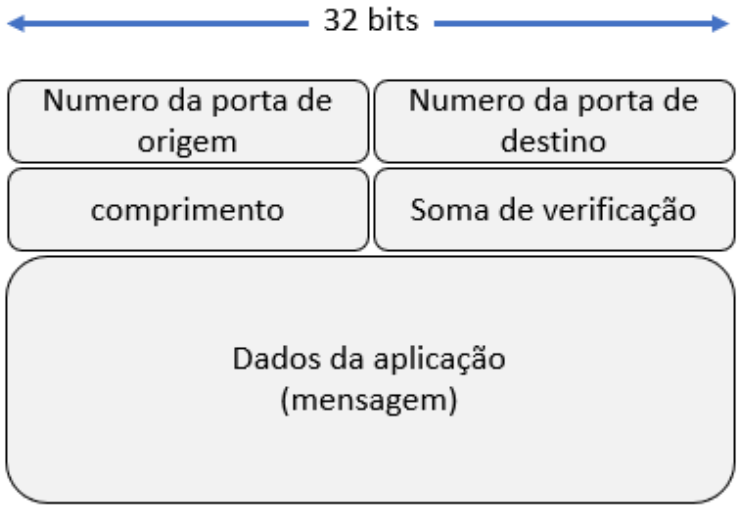
As portas atribuídas para as aplicações tradicionais da Internet são conhecidas como “porta bem conhecidas” (Well Known Ports) e a faixa de endereços reservados para essas portas é de 0 a 1023.

4. User Datagram Protocol – UDP

O protocolo UDP é um protocolo de transporte simples que fornece o mínimo de mecanismos para que aplicações enviem mensagens por meio de uma rede de computadores. Considerado um protocolo pouco confiável, pois não oferece controle de fluxo ou mecanismos de controle para garantir que as mensagens enviadas serão entregues, o UDP é também um serviço não orientado à conexão, ou seja, cada datagrama (nome dado à unidade de transmissão do UDP) enviado pelo usuário é independente (Forouzan, 2010).

Devido ao número reduzido de mecanismos de controle, o cabeçalho UDP é simples e sua função principal é executar a multiplexação das mensagens recebidas da camada de aplicação, bem como realizar a demultiplexação dos datagramas recebidos da camada de rede. Na Figura 2, temos uma representação do cabeçalho UDP.

Figura 2 – Formato do cabeçalho UDP



Conforme apresentado na Figura 2, o cabeçalho UDP é constituído pelos campos porta de origem e de destino, que serão responsáveis pela multiplexação/demultiplexação, pelo campo comprimento que conterá o comprimento total do datagrama UDP (cabeçalho e dados) e, por fim, temos a soma de verificação.

A soma de verificação é um processo matemático gerado a partir das informações do datagrama. Adicionar o resultado desse processo matemático ao cabeçalho UDP permitirá que o destino realize o mesmo processo matemático com o datagrama recebido e compare o valor calculado com o valor recebido. Essa comparação entre o valor calculado no destino da mensagem e o valor contido no campo “soma de verificação” permite ao protocolo assegurar a integridade das informações recebidas.

Devido à simplicidade do protocolo UDP, aplicações de fluxos de dados em tempo real sensíveis à latência e tolerantes à perda de pacotes como telefonia sobre IP, vídeo conferências e transmissões ao vivo, podem se beneficiar do uso desse protocolo.

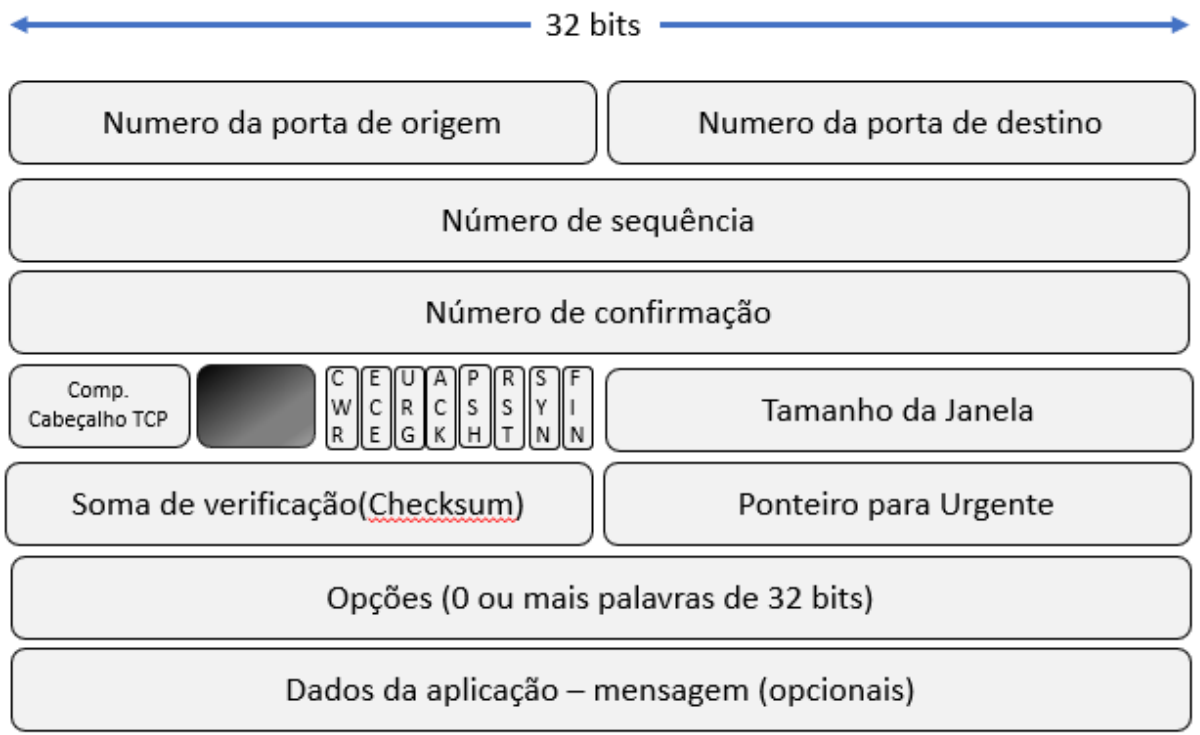
5. Transmission Control Protocol – TCP

O Protocolo de Controle da Transmissão (TCP) é um protocolo orientado à conexão, ou seja, possui um procedimento para estabelecer conexão (three-way handshake), transferir dados e encerrar a conexão, bem como ordenar as mensagens recebidas.

Outra característica do protocolo TCP é o uso de mecanismos de controle para detectar quando uma mensagem não é recebida pelo destino, bem como garantir a integridade das mensagens recebidas. Em caso de perdas ou falhas na transmissão, os pacotes são retransmitidos pela origem.

Com um tamanho de 20 bytes, a estrutura do cabeçalho TCP pode ser vista na Figura 3.

Figura 3 – Formato o Cabeçalho TCP



Na Figura 3, temos o formato do cabeçalho TCP, cujo significado de cada campo será discutido na sequência.

Porta de origem e de destino – Assim como no protocolo UDP, esse campo contém as informações dos números das portas responsáveis pela multiplexação/demultiplexação das mensagens.

Número de sequência – Um dos campos responsáveis pela transferência confiável de dados, o número de sequência identificará cada segmento em função do número de bytes transmitidos.

Número de confirmação – Campo responsável por indicar qual sequência do segmento está sendo confirmada pelo destino.

Comprimento do cabeçalho – Tamanho do cabeçalho do TCP.

CWR (Congestion Window Reduced) e ECE (Explicit Congestion Notification) – são dois Flags de 1 bit cada usados para sinalizar congestionamento entre transmissor e receptor, indicando que o transmissor deve reduzir o fluxo de dados que está sendo enviado.

URG (Urgent) – Flag que indica o envio de segmentos urgentes.

ACK (Acknowledgement) – Esse flag indica que o segmento contém uma confirmação de recebimento e o campo “número de confirmação” deve ser analisado.

PSH (PUSH) – Solicitação para o receptor entregar os dados para aplicação.

RST (Reset) – Utilizada para reiniciar a conexão devido a uma eventual falha ou para recusar a tentativa de conexão.

SYN (Synchronize) – Flag utilizado para iniciar uma conexão entre dois hosts.

FIN (Finalize) – Flag utilizado para encerrar uma conexão entre dois hosts.

Tamanho da Janela – Usado para o controle de fluxo, este flag indica os buffers (memória) disponíveis no receptor.

Soma de verificação (Checksum) – A soma de verificação permite eventuais problemas nos segmentos aumenta a confiabilidade do TCP.

Ponteiro de urgente – Identifica o número de sequência do octeto seguinte ao dado de urgência.

Opções: para recursos não previstos originalmente.

Dados da aplicação: Dados da camada superior (mensagem).

- **Estabelecer conexão**

Como já mencionado, o TCP é um protocolo orientado à conexão, ou seja, o cliente envia um pedido de conexão para o destino, criando um caminho lógico entre ambos que facilitará o processo de verificação de segmentos, bem como a retransmissão de quadros danificados ou perdidos (Forouzan, 2010).

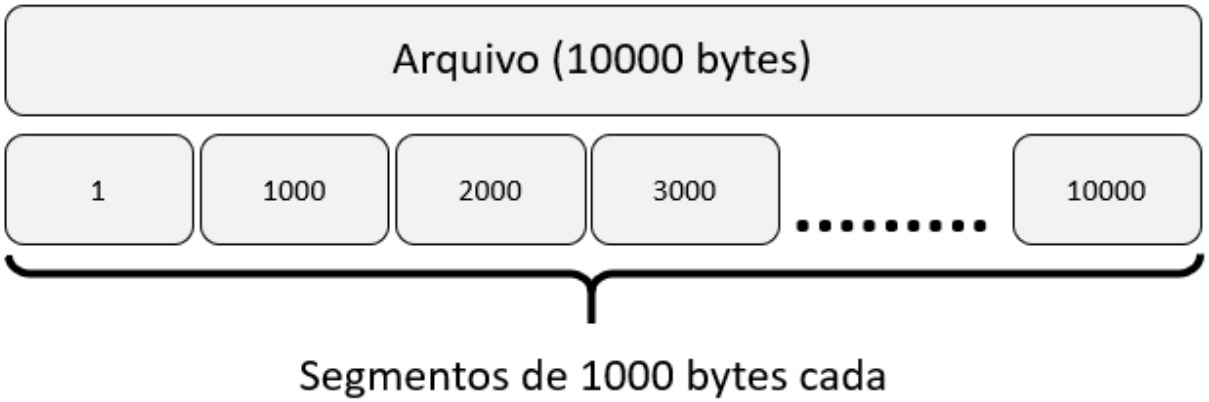
No TCP, a conexão é estabelecida por meio do handshake em três vias (three-way handshake) que tem esse nome em alusão às três mensagens trocadas entre os hosts que estão estabelecendo comunicação.

Assim como o TCP estabelece conexão entre hosts, após a troca de mensagens entre esses hosts, o TCP também tem um procedimento para finalizar a conexão.

- **Transferência de dados confiável**

O TCP vê os dados como uma cadeia de bytes não estruturadas, mas ordenada (Kurose, 2013). Esses bytes são segmentados com um tamanho específico e o campo do cabeçalho TCP " número de segmento" contém como informação o número do primeiro byte do segmento conforme apresentado na Figura 4.

Figura 4 – Divisão dos dados em segmentos TCP



Na Figura 4, temos um exemplo hipotético no qual o arquivo oriundo da camada de aplicação com um tamanho 10k bytes foi bem segmentado na camada de transporte. O número dentro de cada segmento representa o "número de sequência" do protocolo TCP que serve para identificar um pacote específico. Com base no número de sequência, o TCP consegue confirmar para o remetente quais pacotes foram recebidos, permitindo, assim, a retransmissão de pacotes que não chegaram ao destino ou chegaram danificados.

- **Entrega ordenada de segmentos**

Devido a características das redes comutadas que serão abordadas em outras aulas, os segmentos enviados podem não chegar de maneira ordenada ao destino. O número de segmento, além de garantir a entrega dos segmentos, também permite que estes sejam ordenados no destino.

- **Controle de fluxo**

O TCP oferece um controle de fluxo que tem como objetivo equalizar o envio de dados entre o transmissor e o receptor. Controlada pelo campo “tamanho da janela”, transmissor e receptor compartilham informações sobre o tamanho do buffer disponível no receptor, determinando a quantidade máxima de bytes que o transmissor pode enviar sem sobrecarregar o receptor.

- **Controle de congestionamento**

Além do controle de fluxo usado para não sobrecarregar o receptor, o TCP analisa as mensagens de confirmação (Acknowledgement), bem como as informações de RTT (Round Trip Time) dessas confirmações, para estimar o congestionamento da rede. Quando o TCP detecta algum tipo de congestionamento, ele reduz a taxa de transmissão com o intuito de não congestionar ainda mais a rede e reduzir a perda de pacotes.

6. REFERENCIAS

FOROUZAN, B. A. Comunicação de dados e Redes de computadores. 3. ed. Porto Alegre: Bookman, 2010.

KUROSE, J. F.; ROSS, K. W. Redes de computadores e a internet: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2013 | Biblioteca Virtual Universitária — Pearson.

SILBERSCHATZ, A.; GALVIN, P.B.; GAGNE, G. Fundamentos de Sistemas Operacionais: princípios básicos. São Paulo: LTC, 2015.

TANENBAUM, A. Redes de computadores. 5. ed. São Paulo: Pearson, 2010 | Biblioteca Virtual Universitária — Pearson.