Apresentação

Você já sabe que a camada de transporte pode multiplexar várias conversas entre diferentes aplicações. Mas como ela identifica para qual aplicação deve ser entregue um segmento específico?

1. Endereço de Servicos (Portas)

Tanto o TCP como o UDP usam números de porta armazenados em campos de cabeçalho, onde o número da porta de origem está associado ao aplicativo de origem no host local, enquanto o número da porta de destino está associado ao aplicativo de destino no host remoto. A figura 1 mostra uma conversa entre dois hosts e suas respectivas portas de origem e destino.

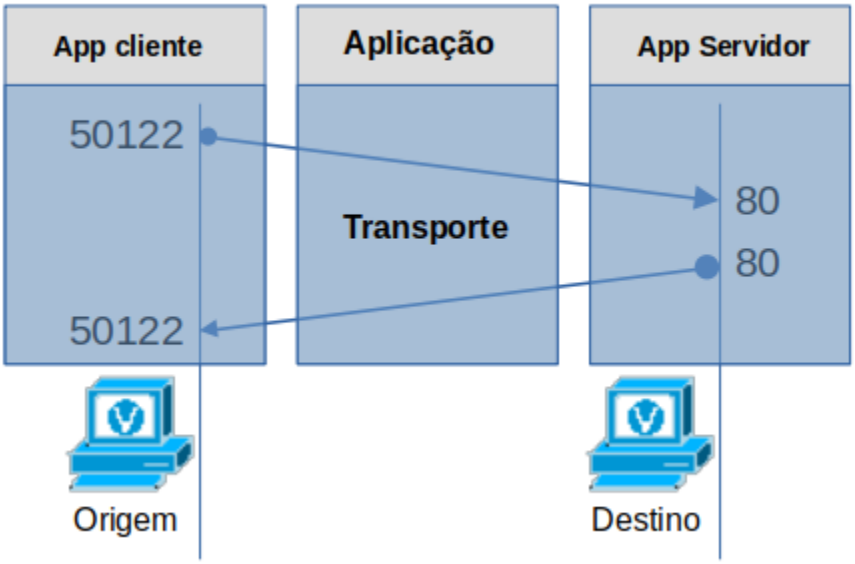

Fig. 1 - Conversação na camada de transporte Fonte: o autor

Por exemplo, imagine que um host está iniciando uma solicitação de página a um servidor Web. Quando o host inicia a solicitação, um número da porta de origem é gerado dinamicamente para identificar exclusivamente esta conversa. O número da porta de destino identifica qual serviço está sendo solicitado ao servidor de destino. No caso, quando um cliente especifica a porta 80 na porta de destino, o servidor que recebe a mensagem sabe que é o serviço Web que está sendo solicitado.

Destaque: Cada solicitação gerada por um host usará um número de porta de origem diferente, criado dinamicamente, diferente, isto permite que várias conversações

ocorram simultaneamente. Já os servidores utilizam endereços de porta fixos de forma a poderem ser facilmente identificados.

1.1. Sockets

As portas de origem e destino são colocadas no cabeçalho dos segmentos TCP ou datagramas UDP, e estes encapsulados em um pacote IP, que por sua vez contém o endereço IP de origem e destino. A combinação do endereço IP de origem e o número de porta de origem, ou do endereço IP de destino e o número de porta de destino é conhecida como um socket. A figura 2 representa esquematicamente uma PDU da camada de transporte, encapsulada em um pacote IP e os respectivos sockets formados.

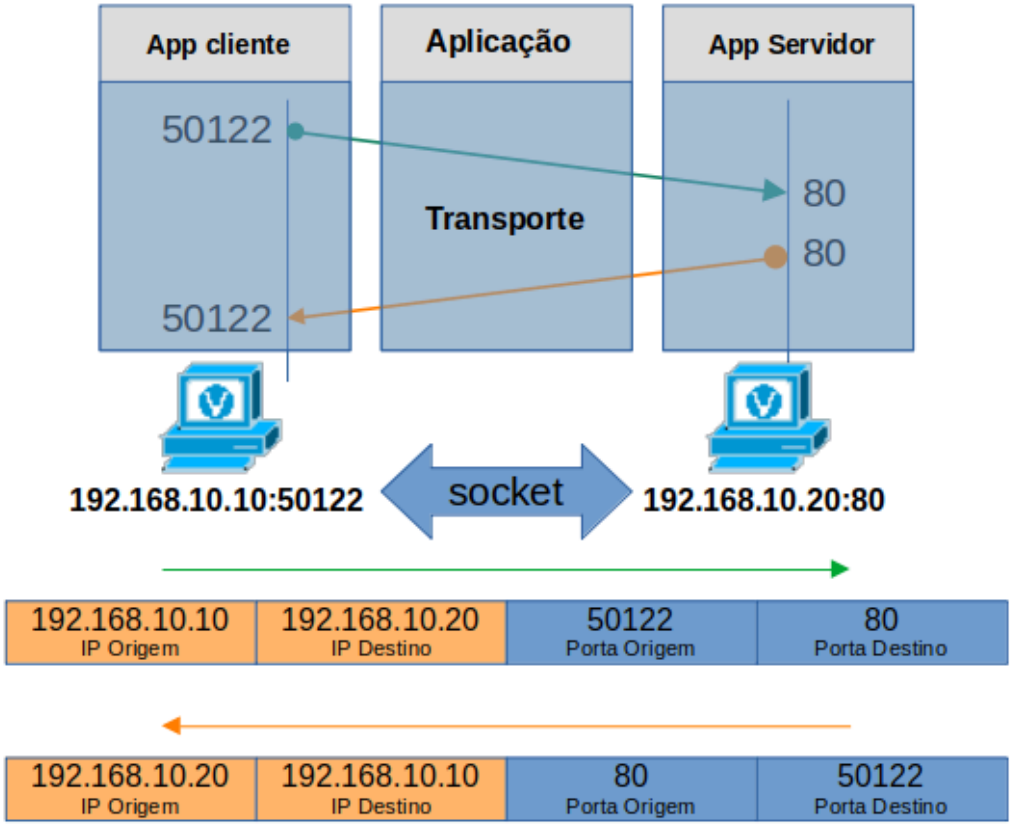

Fig. 2 - Sockets Fonte: o autor

Na solicitação o socket é usado para identificar o servidor e o serviço desejado, assim como o cliente e o serviço solicitante. Na figura 7 o socket do cliente pode ser representado pelo seu endereço IP de origem e o número da porta de origem. isto é, 192.168.10.10:50122 e o socket do servidor da web pode ser o endereço IP de destino e a porta de desino , ou seja, 192.168.10.20:80. Esses dois sockets se combinam para

formar um par de sockets: 192.168.10.10:50122 <-> 192.168.10.20:80. Na resposta os sockets de origem e destino se invertem. Os sockets permitem que os vários processos em execução em um cliente se diferenciem uns dos outros, e várias conexões com um processo no servidor sejam também diferentes umas das outras.

1.2. Grupos de portas

A Internet Assigned Numbers Authority (IANA) é a organização internacional responsável por atribuir os números de porta TCP/UDP de 16 bits, logo fornecem um intervalo de portas de 0 a 65535 para cada protocolo. Esta padronização está definida na RFC 6335 que dividiu a gama de números nos três grupos de portas apresentadas na tabela 1.

Tabela 1 - Grupos de portas TCP/UDP definidas pelo IANA

Grupo de Portas	Intervalo de números	Descrição
Portas Baixas	**0 a 1.023**	● Também chamadas de portas bem conhecidas(Well Known ports) definidas para aplicativos comuns de servidor que permitem identificar facilmente o serviço associado desejado, como servidores WEB, e-Mail, etc.
Portas registradas	**1.024 a 49.151**	● Esses números de porta são atribuídos pela IANA a uma entidade solicitante, para serem usados com processos ou aplicativos específicos, que por algum motivo não podem, ou não se deseja, utilizar as portas bem conhecidas. Por exemplo, a Cisco registrou a porta 1812 para seu servidor RADIUS de autenticação.

Tabela 1 - Grupos de portas TCP/UDP definidas pelo IANA		
Portas dinâmicas	**49.152 a 65.535**	• Também são conhecidas como portas *efêmeras*, sendo atribuídas dinamicamente pelo sistema operacional do cliente quando uma conexão a um serviço é iniciada, usada para identificar o aplicativo cliente durante a comunicação.

Destaque: Alguns sistemas operacionais clientes podem usar números de porta registrados em vez de números de porta dinâmicos para atribuir portas de origem.

As chamadas portas baixas e as portas registradas são utilizadas para identificar os serviços instalados nos servidores e que devem ser conhecidas para serem acessadas. A tabela 2 mostra alguns números de porta mais utilizados e seus aplicativos associados.

Tabela 2 - Números de portas comumente utilizados		
Número da Porta	Protocolo	Aplicação
20	TCP	Protocolo de transferência de arquivos (FTP) - Dados
21	TCP	Protocolo de transferência de arquivos (FTP) - Controle
22	TCP	Secure Shell (SSH - Secure Shell) - Emulação de terminal segura
23	TCP	Telnet - Emulação de Terminal
25	TCP	Protocolo SMTP (Simple Message Transfer Protocol) - Envio de e-Mail.
53	UDP, TCP	Protocolo DNS (Domain Name Service) - Resolução de Nomes de Domínio
67	UDP	Protocolo de Configuração Dinâmica de Host (DHCP) - Servidor

68	UDP	Protocolo de configuração dinâmica de host (DHCP) - cliente
69	UDP	Protocolo de Transferência Trivial de Arquivo (TFTP)
80	TCP	Protocolo HTTP (Hyper text Transfer Protocol) - WEB
110	TCP	Protocolo POP3 (Post Office Protocol) - Protocolo de recuperação de mensagens de e-mail)
143	TCP	Protocolo IMAP (Internet Mail Application Protocol) - Protocolo de recuperação de mensagens de e-mail)
161	UDP	Protocolo de Gerenciamento de Rede (SNMP - Simple Network Management Protocol)
443	TCP	HTTPS (Secure Hypertext Transfer Protocol - Protocolo de Transferência de Hipertexto Seguro)

Destaque: Algumas aplicações podem usar tanto o protocolo TCP quanto UDP. Por exemplo, o DNS usa o protocolo UDP quando os clientes enviam requisições a um servidor DNS, porém, a comunicação entre dois servidores DNS sempre usa o protocolo TCP.

Praticando: Identificando os sockets - O netstat é um utilitário de rede que pode ser usado para verificar os pares de sockets estabelecidos entre as aplicações clientes e servidoras. A figura 8 apresenta a saída do comando netstat -na

```
C:\netstat -na

Conexões ativas

Proto  Endereço local         Endereço externo       Estado
TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING
...
TCP    10.200.130.190:51612   52.226.139.180:443     ESTABLISHED
TCP    10.200.130.190:51643   13.107.42.16:443       CLOSE_WAIT
TCP    10.200.130.190:51666   172.217.192.188:5228   ESTABLISHED

...
UDP    0.0.0.0:5050           *:*
UDP    0.0.0.0:5353           *:*
UDP    0.0.0.0:5353           *:*
...
Saída Omitida
```

Fig. 8 - Saída do comando netstat Fonte: o autor

Veja o endereço local e os números de porta, o endereço externo e os números de porta, o protocolo e o estado da conexão. Um host não pode ter dois serviços atribuídos ao mesmo número de porta, por exemplo, um host executando um aplicativo web e um aplicativo de transferência de arquivos não podem ser configurados para usar a mesma porta.

Um aplicativo de servidor ativo atribuído a uma porta específica é considerado aberto, diz-se que ele está escutando a rede (LISTENING), o que significa que a camada de transporte aceita e processa os segmentos endereçados a essa porta, por exemplo, a porta TCP 135 ou 445 da figura. Qualquer solicitação de cliente que chega endereçada ao soquete correto é aceita e os dados são transmitidos à aplicação servidora. Também podemos ter várias portas abertas ao mesmo tempo em um servidor, uma para cada aplicação de servidor ativa.

Apresentação

No modelo TCP/IP a camada de transporte usa dois protocolos: TCP, orientado à conexão e o UDP, não orientado à conexão. Podemos comparar o TCP com serviço de carta registrada do correio, onde o remetente recebe um aviso de que a carta foi entregue com sucesso no destino. Já o UDP é mais como uma carta simples, um datagrama, onde o correio não informa ao remetente se a correspondência chegou ou não. Nessa aula vamos conhecer os principais detalhes e aplicações destes protocolos.

1. Protocolo TCP

O protocolo TCP é um protocolo da camada de transporte, do modelo TCP/IP, orientado à conexão, definido pela IETF na RFC 793, que além de suportar as funções básicas de segmentação e remontagem de dados, fornece os serviços de estabelecimento da conexão, entrega confiável, controle de erro e controle de fluxo para as conversações entre os protocolos da camada de aplicação.

1.1. O cabeçalho TCP

O cabeçalho TCP possui 20 bytes (160 bits) e encapsula a PDU da camada de aplicação. A figura 1 apresenta os campos em um cabeçalho TCP.

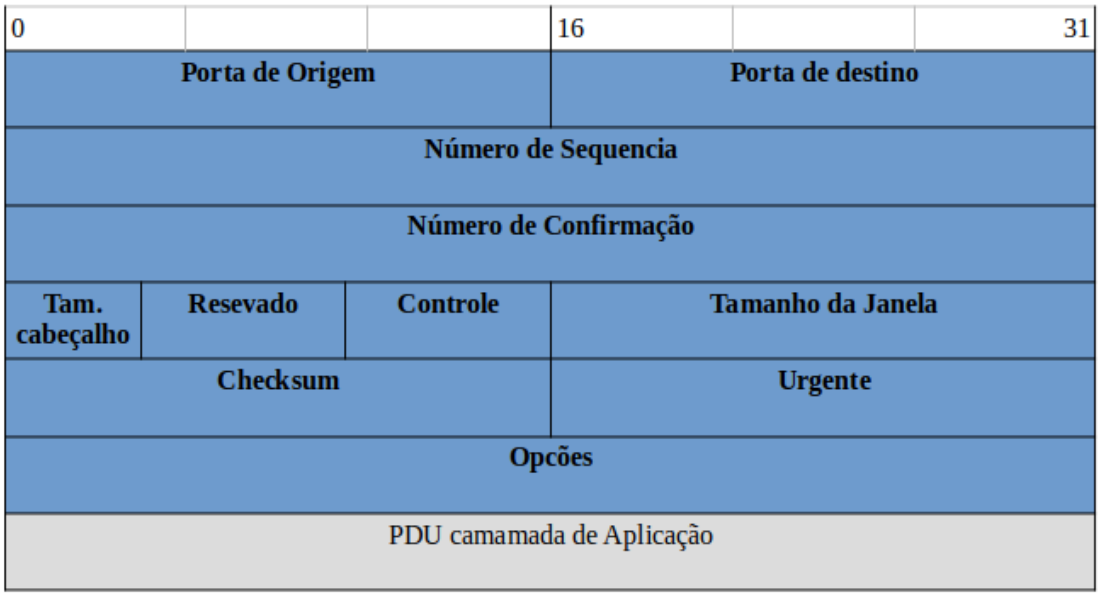

Figura 1 - Cabeçalho TCP Fonte: o autor

A tabela 1 apresenta os principais campos do cabeçalho TCP

Tabela 1 - Principais campos do cabeçalho TCP	
Campo de cabeçalho TCP	Descrição
Porta de origem (16 bits)	Usado para identificar o aplicativo de origem por número de porta.
Porta de destino (16 bits)	Usado para identificar o aplicativo de destino por porta número.
Num. de Sequência (32 bits)	Usado para fins de remontagem de dados.
Número de Confirmação (32	Usado para indicar que os dados foram recebidos e o

bits)	próximo byte esperado da fonte.
Tamanho do cabeçalho (4 bits)	Conhecido como 'offset' que indica o comprimento do cabeçalho do segmento TCP.
Reservado (6 bits)	Reservado para uso futuro.
Controle (6 bits)	Inclui os códigos de bits, ou sinalizadores, que indicam a finalidade e função do segmento TCP. Os seis bits de controle sinalizadores são os seguintes: ● URG - Ativa o campo de Urgência. ● ACK - Indicador de confirmação usado no estabelecimento de conexão e encerramento de sessão. ● PSH - Função Push. ● RST - Redefina a conexão quando ocorrer um erro ou tempo limite. ● SYN - Sincronizar números de sequência usados no estabelecimento de conexão. ● FIN - Não há mais dados do remetente e usados no encerramento da sessão.
Tamanho da janela (16 bits)	Usado para indicar o número de bytes que podem ser aceitos de uma só vez.
Checksum (16 bits)	Usado para verificação de erros do cabeçalho e dos dados do segmento.
Urgente (16 bits)	Usado para indicar o tipo de urgência dos dados..

2. Funcionalidades de protocolo TCP

2.1. Estabelecimento de conexão TCP

Assim como nas conversas entre nós humanos, quando duas aplicações trocam dados é preciso que antes do diálogo seja estabelecida uma conexão, e para isso é feito um handshake (aperto de mãos) Nas conexões TCP, o host cliente estabelece a conexão com o servidor usando o processo de handshake de três vias (Three Way Handshake). A figura 2 apresenta esquematicamente esse processo.

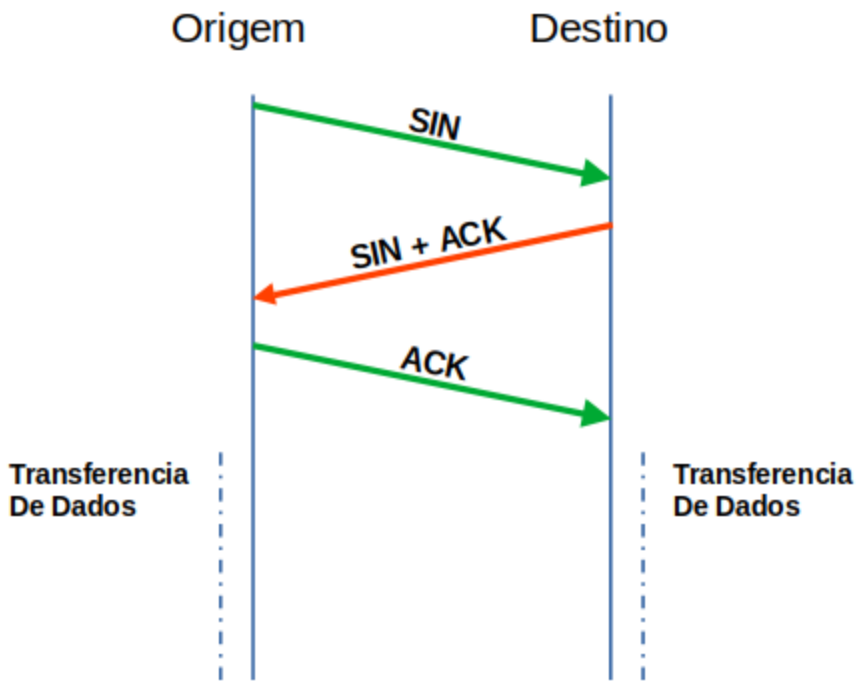

Fig. 2 - Abertura de uma conexão TCP

O cliente requisita uma sessão de comunicação cliente-servidor com o servidor enviando um segmento com o flag de controle SYN ativado, o servidor confirma a sessão de comunicação cliente-servidor e requisita uma sessão de comunicação de servidor-cliente com um SYN + ACK(Acknowledgment), O cliente iniciador confirma a sessão de comunicação de servidor-cliente, enviando um ACK. O handshake de três vias permite a validação mútua, garantindo que ambos se encontram disponíveis para comunicação.

2.2. Encerramento da conexão

Para terminar uma sessão TCP utiliza-se um handshake duplo, que consiste de um segmento FIN e um segmento ACK (Acknowledgment) para cada um dos participantes. Portanto, para terminar uma conversação única, quatro trocas são necessárias que são apresentadas na figura 3.

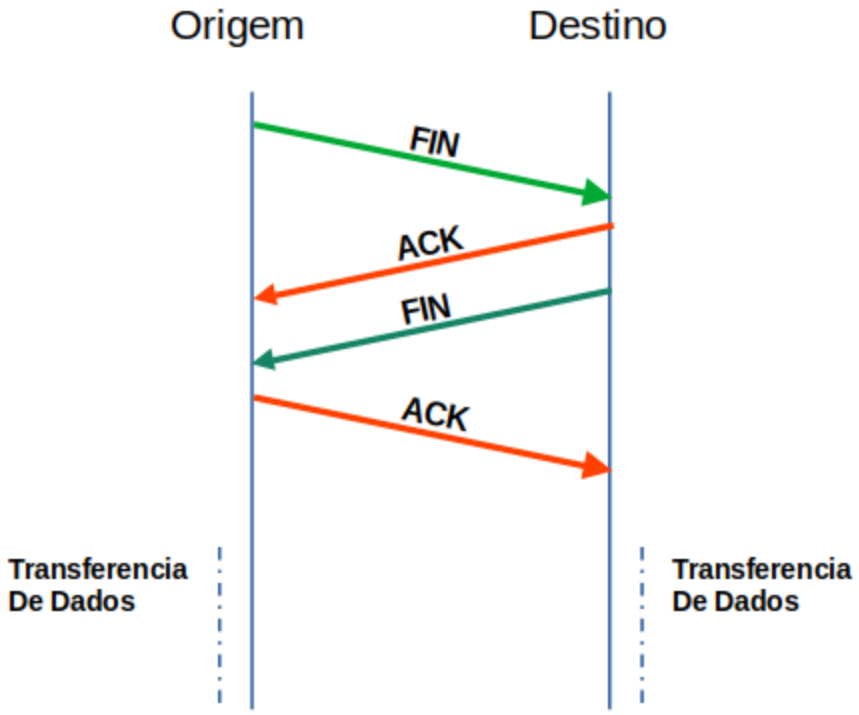

Figura 3 - Encerramento da conexão TCP

Tanto o cliente quanto o servidor podem solicitar o encerramento enviando um segmento com o flag de controle Finish (FIN) ligado no cabeçalho do segmento, aquele que recebe a solicitação responde com um ACK para confirmar o recebimento de FIN e encerrar a seu lado da conexão, em seguida envia um FIN ao seu par para encerrar a sessão do outro lado, que responde com um ACK para reconhecer ao FIN e encerrar sua própria conexão.

2.3. Números de sequência

O TCP é um protocolo confiável porque ele reenvia segmentos descartados e os numera indicar sua ordem correta antes da entrega, já que estes podem não chegar ao seu destino, ou às vezes chegar fora de ordem, devido a comutação de pacotes da camada 3. Para que a mensagem original seja entendida pelo destinatário, todos os dados devem ser recebidos e remontados na ordem original. Os números de sequência são atribuídos no cabeçalho de cada pacote para alcançar esse objetivo.

Após o estabelecimento da conexão, um número de sequência inicial (ISN - Initial Sequence Number) é definido indicando o valor inicial para os bytes que são transmitidos ao aplicativo receptor. A figura 4 apresenta um esquema para incremento do ISN.

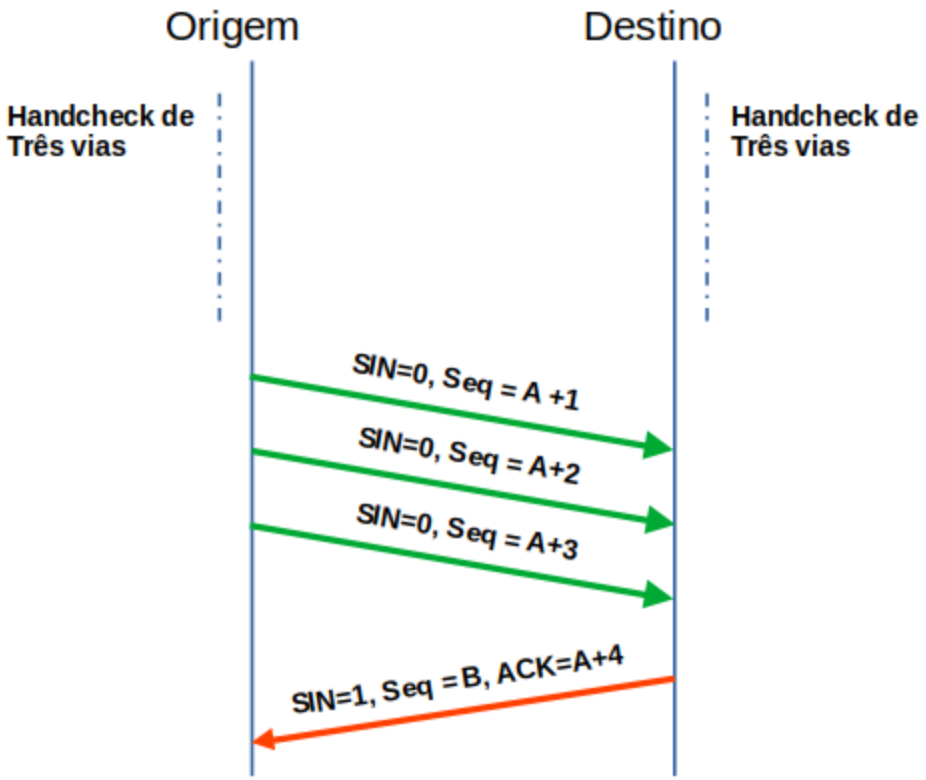

Fig. 4 - Incremento dos ISN Fonte: o autor

À medida que os dados são transmitidos durante a sessão, o número de sequência “A” é incrementado do número de bytes que serão transmitidos, gerando para cada segmento um número único para aquela conexão, permitindo que cada um seja identificado e confirmado de forma única. Segmentos perdidos podem, com isso, ser identificados.

Destaque: O ISN começa com um número aleatório, isso impede tipos de ataques maliciosos.

2.4. controle de erro

O número de sequência (SEQ) e o número de confirmação (ACK) são usados para confirmar o recebimento dos segmentos. O número SEQ identifica o byte que está sendo transmitido, e o número de confirmação (ACK) enviado de volta à origem indica o próximo byte que o destino espera receber. Isto é chamado de confirmação

antecipatória. Inicialmente o TCP só podia reconhecer o próximo byte esperado. Por exemplo, a figura 5 apresenta um esquematicamente um processo de transmissão e confirmação de segmentos utilizando ACK.

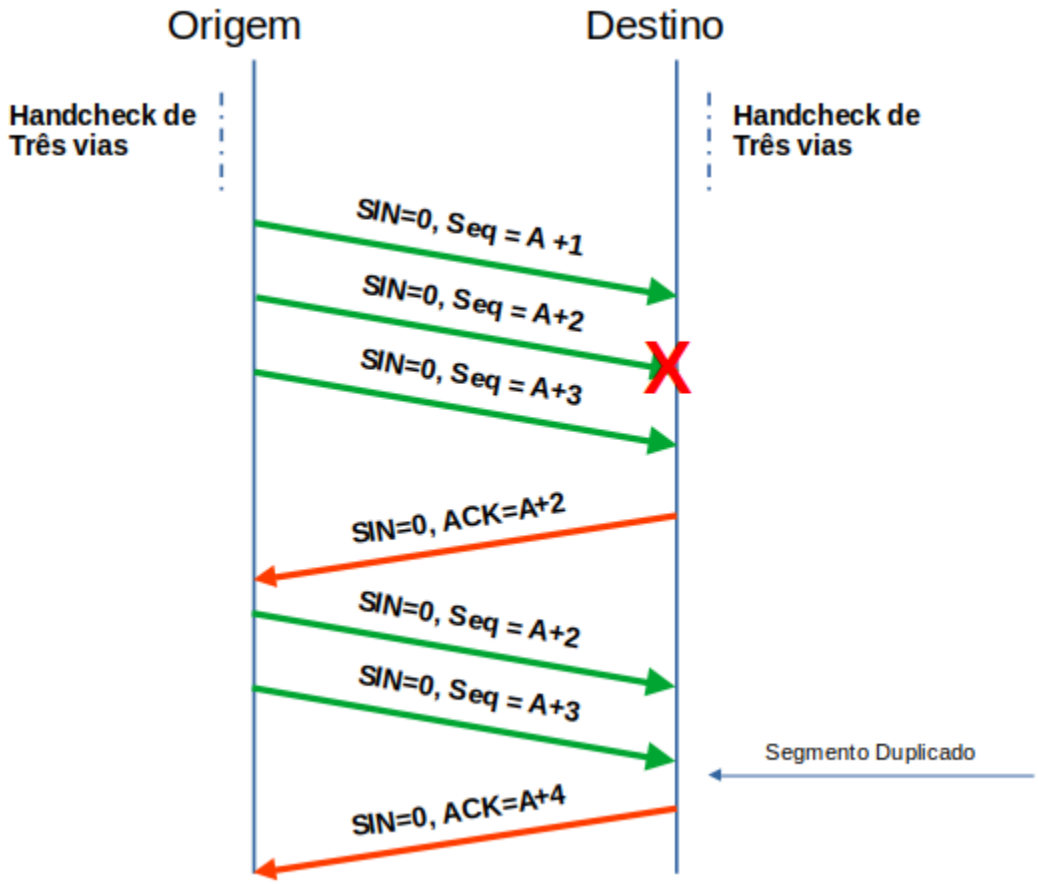

Fig. 5 - Transmissão e confirmação de segmentos TCP utilizando ACK Fonte: o autor

O host origem envia os segmentos de A+1 a A+3. Se todos os segmentos chegarem, exceto o segmento A+2, o host de destino responderia com um ACK indicando que o próximo segmento esperado é o A+3. Como o host de origem não sabe se outros segmentos chegaram, reenviaria os segmentos A+2 e A+3. Isso acarreta a transmissão de segmentos duplicados, ocasionando desperdício de banda e problemas de congestionamento.

Atualmente os sistemas operacionais utilizam um recurso chamado reconhecimento seletivo (SACK), negociado durante o handshake de três vias. Se ambos os hosts suportarem SACK, o receptor pode reconhecer explicitamente quais segmentos (bytes) foram recebidos e quais não, e o emissor só precisa transmitir os segmentos ausentes. figura 6 apresenta este esquema de transmissão utilizando SACK.

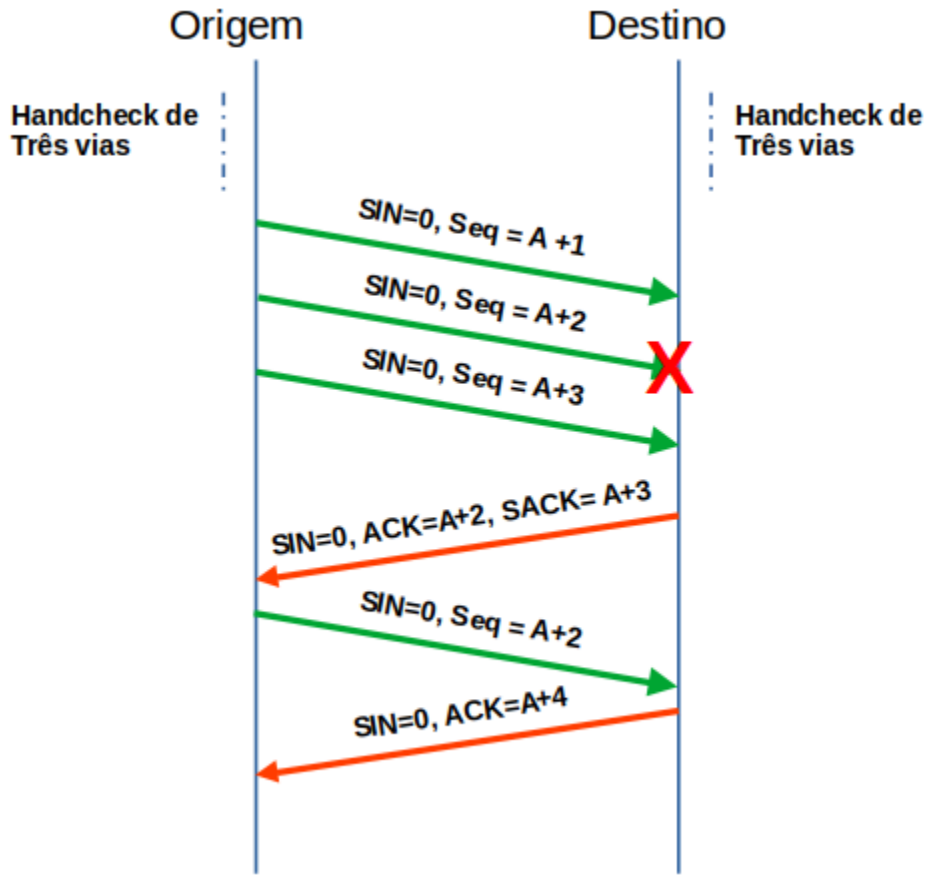

Fig. 6 - Transmissão e confirmação de segmentos TCP utilizando SACK Fonte: o autor

Aqui o host origem envia os segmentos de A+1 a A+3 para o host de destino. Se todos os segmentos chegarem, exceto o segmento A+2, o host de destino responderia com confirmação especificando que o próximo segmento esperado é o segmento A+2 (ACK=A+2) e que recebeu seletivamente o segmento A+3 (SACK=A+3). O host de origem então reenviaria apenas o segmento A+2. Isso evita segmentos duplicados e desperdício de banda.

2.5. Controle de fluxo

O TCP utiliza o campo tamanho da janela, que determina o número de bytes que podem ser enviados antes de esperar uma confirmação e o tamanho máximo da janela (MSS - Maximum segment Size) trocado no campo opções fornecer controle de fluxo, regulando a quantidade de dados que o destino pode receber e processar de forma confiável. O número de confirmação (ACK) corresponde ao próximo byte esperado. A figura 7 mostra o controle de fluxo entre dois processos utilizando o protocolo TCP.

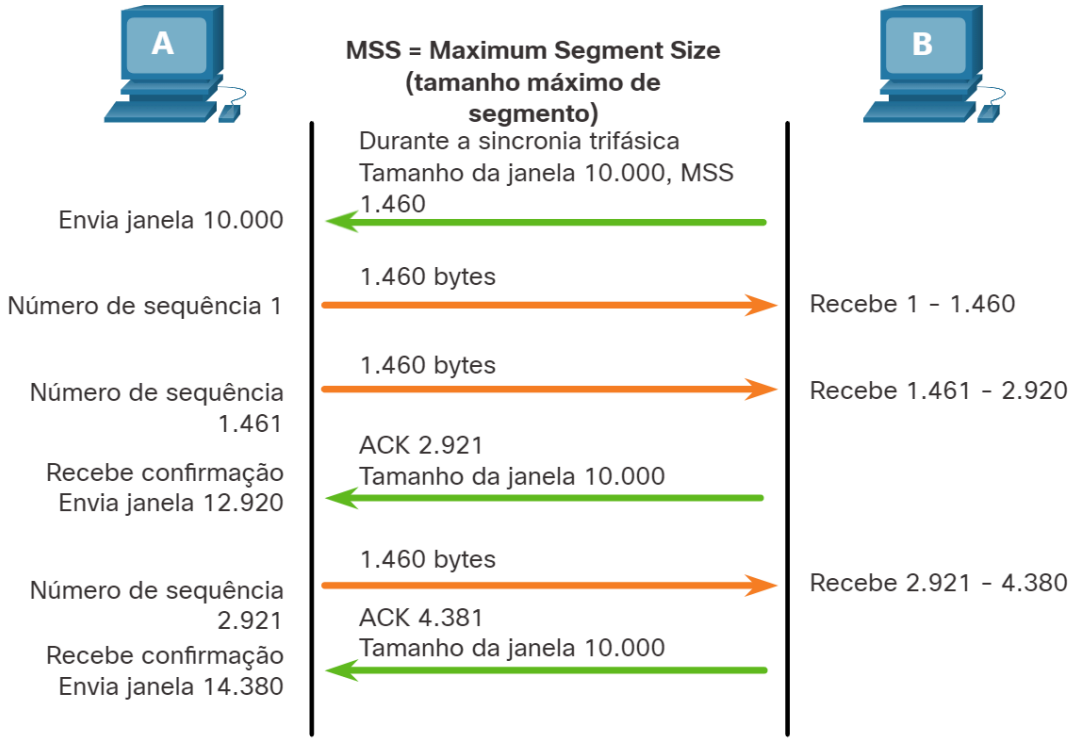

Fig. 7 - Controle de fluxo TCP fonte: CCNA 1 - Introduction to Network v7.02 item 14.6.6

Neste exemplo, o tamanho da janela inicial do PCB para a sessão TCP é de 10.000 bytes e o MSS é de 1460 bytes. Tomando o primeiro byte igual a 1, o último byte que PC A pode enviar sem receber uma confirmação é o byte 10.000. Isso é conhecido como janela de envio do PC A. O tamanho da janela pode ser fixo, isto é, não pode ser alterado naquela conexão ou deslizante, que pode ser alterado ao longo da mesma, nesse caso o tamanho da janela é incluído em todos os segmentos TCP. A vantagem de janelas deslizantes é que permite que o emissor transmita continuamente segmentos, desde que o receptor esteja reconhecendo segmentos anteriores.

3. Protocolo UDP

O protocolo UDP, definido pela RFC 768, é um protocolo não orientado à conexão, leve, que oferece a mesma segmentação de dados e remontagem que o TCP, porém não faz estabelecimento de conexão e é sem confiabilidade, o que significa que pacotes perdidos não são retransmitidos e não faz controle de fluxo.

3.1. Datagrama UDP

As PDUs de comunicação no UDP são chamados de datagramas e enviados como o melhor esforço pelo protocolo da camada de transporte, seu cabeçalho é muito mais simples do que o do TCP, só tem quatro campos e requer 8 bytes (ou seja, 64 bits)

apenas. A figura 8 apresenta a estrutura de uma datagrama UDP.

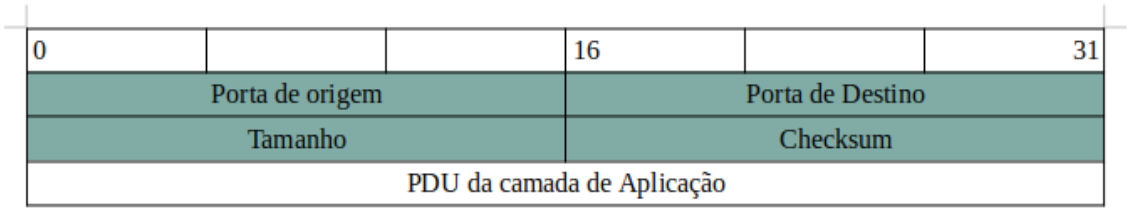

Figura 8 Datagrama UDP Fonte: o autor

A tabela 2 descreve os campos do cabeçalho UDP.

Tabela 2 - Datagrama UDP	
Campo do Cabeçalho UDP	**Descrição**
Porta de origem (16 bits)	Usado para identificar a porta de origem..
Porta de destino (16 bits)	Usado para identificar a porta de destino.
Tamanho (16 bits)	Indica o comprimento do cabeçalho do datagrama UDP.
Checksum (16 bits)	Usado para detecção de erros do cabeçalho e dos dados do datagrama.

Por ser muito mais simples que o TCP é muito mais rápido sendo ideal para as aplicações que precisam que os dados fluam rapidamente e podem tolerar alguma perda de dados com efeito mínimo no resultado final. Há três tipos de aplicações que são mais adequadas para o UDP:

- **Aplicativos de vídeo e multimídia ao vivo** - Esses aplicativos podem tolerar a perda de dados, mas requerem pouco ou nenhum atraso. Os exemplos incluem VoIP e transmissão de vídeo ao vivo.
- **Solicitações simples e aplicativos de resposta** - aplicativos com transações simples em que um host envia uma solicitação e pode ou não receber uma resposta. Os exemplos incluem DNS e DHCP.
- **Aplicativos que lidam com a confiabilidade** - Comunicações unidirecionais em que o controle de fluxo, a detecção de erros não são necessários ou podem ser gerenciados pela camada de aplicação. Os exemplos incluem SNMP e

TFTP.