Especificação do projeto 2 **Protocolo de transporte sobre UDP**
Redes de Computadores 2018.1
Prof. Filipe

Visão geral

Neste projeto você implementará uma versão básica de um protocolo de transferência de dados confiável, incluindo estabelecimento de conexão e controle de congestionamento. Você implementará o protocolo no contexto de aplicações cliente/servidor, onde o cliente transmite um arquivo tão logo uma conexão esteja estabelecida.

A implementação é na linguagem de sua escolha. Porém, não devem ser utilizados frameworks de programação em rede. Devem ser utilizadas somente abstrações não relacionadas a rede, como parser, multi-thread e etc.

O objetivo deste projeto é aprofundar seu conhecimento do TCP, especificamente como ele manipula perda de pacotes e reordenamento.

Descrição

O projeto consiste de duas partes: um cliente e um servidor.

- O servidor abre um socket UDP e gerencia o estabelecimento, bem como as conexões requisitadas pelos clientes. Para cada conexão, o servidor salva os dados recebidos de um cliente em um arquivo.
- O cliente abre uma conexão UDP, implementa o gerenciamento da conexão, e conecta a um servidor. Desde que a conexão está estabelecida, envia o conteúdo de um arquivo para um servidor.

Ambos cliente e servidor devem implementar a transferência de dados confiável utilizando a camada de transporte não confiável UDP, incluindo: sequenciamento de dados, reconhecimento cumulativo e uma versão básica de controle de congestionamento.

Especificação do protocolo

0.1 Cabeçalho

O payload (carga útil, parte de dados) de cada segmento UDP enviado pelo servidor e cliente DEVE iniciar com o cabeçalho de 12 bytes seguinte:

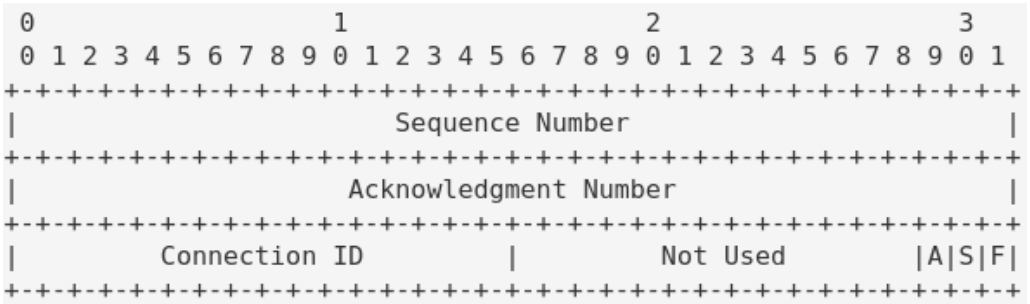


Figura 1: Cabeçalho.

Onde

Sequence Number (32 bits): O número de sequencia do primeiro octeto de dados no pacote, exceto quando SYN estiver presente.

Se SYN estiver presente, o número de sequencia é o número de sequencia inicial e o primeiro octeto de dados é SYN + 1.

Ack Number (32 bits): Se o bit ACK estiver definido, este campo contém o valor do próximo número de sequencia que o receptor do segmento está esperando. Desde que a conecão está estabelecida, isso sempre é enviado.

Connection ID (16 bits): Um número representando o identificado de conexão.

Not used (13 bits): Deve ser zero.

A (ACK, 1 bit): Indica que o valor do campo Acknowledgement Number é válido.

S (SYN, 1 bit): Sincroniza os números de sequencia, estabelecimento da conexão.

F (FIN, 1 bit): Sem mais dados do emissor, término da conexão.

Requisitos:

- O tamanho máximo do pacote UDP é 524, incluindo o cabeçalho (512 bytes de payload).
- O número máximo de sequencia e de reconhecimento deve ser 102400 e redefinido para zero toda vez que alcançar seu máximo.
- Retransmissão de pacotes, e as ações de controle de congestionamento apropriadas, devem ser engatilhadas quando nenhum dado é reconhecido por mais de 0.5 segundos.
- Tamanho da janela de congestionamento (CWND) inicial e mínimo deve ser 512.
- Limite de partida lenta inicial (SS-THRESH) deve ser 10000.
- Se o campo ACK não estiver definido, o número de reconhecimento deve ser zero.
- FIN deve ter um byte de comprimento.
- FIN e FINACK não devem carregar dados.

0.2 Especificação da aplicação servidora

A aplicação servidora DEVE ser compilada no aplicativo server, aceitando dois argumentos:

```
$ server <PORT> <FILE-DIR>
```

- <PORT>: número da porta em que o servidor escutará conexões.
- <FILE-DIR>: nome do diretório onde salvará os arquivos a serem recebidos.

Por exemplo, o comando abaixo inicia o servidor escutando na porta 5000 e salva os arquivos recebidos no diretório /save.

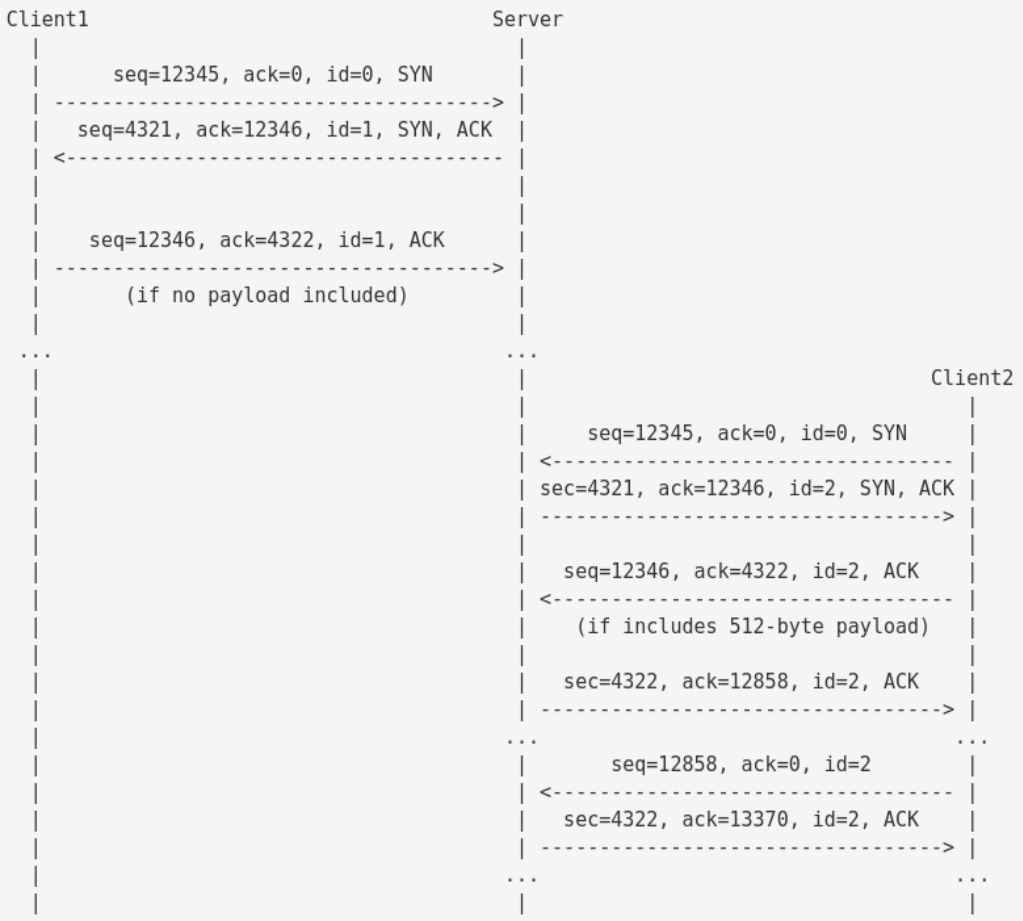
```
$ server 5000 /save
```

Requisitos:

- Servidor deve abrir um socket UDP no número de porta especificado.
- Servidor deve processar números de porta incorretos e terminar a execução com uma mensagem de erro.
- Servidor deve terminar CORRETAMENTE quando receber um sinal de término do sistema operacional.
- Servidor deve contar todas as conexões estabelecidas (1 para o primeiro a se conectar, 2 para o segundo, etc.). O arquivo recebido sob a conexão deve ser salvo em <FILE-DIR>/<CONNECTION-ID>.file (ex., /save/1.file, /save/2.file).
- O servidor deve aceitar e gerenciar múltiplas conexões de clientes ao mesmo tempo.

  Após recebe o pacote com flag SYN, o servidor deve criar estado para Connection ID e proceder com o handshake de 3 vias. Servidor deve usar 4321 como número de sequencia inicial.

  Depois de receber um pacote sem SYN, o servidor deve buscar a conexão utilizando Connection ID e proceder com a ação apropriada para a conexão.



- O servidor deve assumir erro se nenhum dado for recebido de um cliente durante 10 segundos. Ele deve abortar a conexão e escrever a string ERROR no arquivo correspondente.
- O servidor deve aceitar e salvar arquivos de até 100MB.

0.3 Especificação da aplicação cliente

A aplicação cliente deve ser compilada no executável client, aceitando três argumentos da linha de comando.

```
$ client <HOSTNAME-OR-IP> <PORT> <FILENAME>
```

- <HOSTNAME-OR-IP>: hostname ou IP do servidor ao qual conectar-se-á.
- <PORT>: número da porta no servidor ao qual conectar-se-á.
- <FILENAME>: nome do arquivo a ser transferido ao servidor após a conexão está estabelecida.

Por exemplo, o comando abaixo deve resultar na conexão ao servidor executando na mesma máquina, escutando na porta 5000 e transferindo o conteúdo de file.txt.

```
$ client localhost 5000 file.txt
```

Requisitos:

- O cliente deve abrir um socket UDP e iniciar o handshake de 3 vias para o hostname/ip e porta especificados.

  Enviar o pacote UDP src-ip=DEFAULT, src-port=DEFAULT, dst-ip=HOSTNAME-OR-IP, dst-port=PORT com flag SYN definido, CONNECTION-ID inicializada com 0, SEQUENCE NUMBER inicializada como 12345 e Acknowledgement Number definido como 0.

  Esperar respostas do servidor com flags SYN-ACK. O cliente deve registrar o CONNECTION-ID retornado e utilizá-lo em todos os pacotes subsequentes.

  Enviar pacote UDP com flag ACK incluindo a primeira parte do arquivo especificado.

- O cliente deve processar um hostname/ip incorreto, bem como número da porta, e terminar com uma mensagem de erro relacionada.

- Depois do arquivo ter sido transferido com sucesso, o cliente deve encerrar a conexão.

  Enviar pacote UDP com flag FIN definida.

  Esperar pacote com flag ACK.

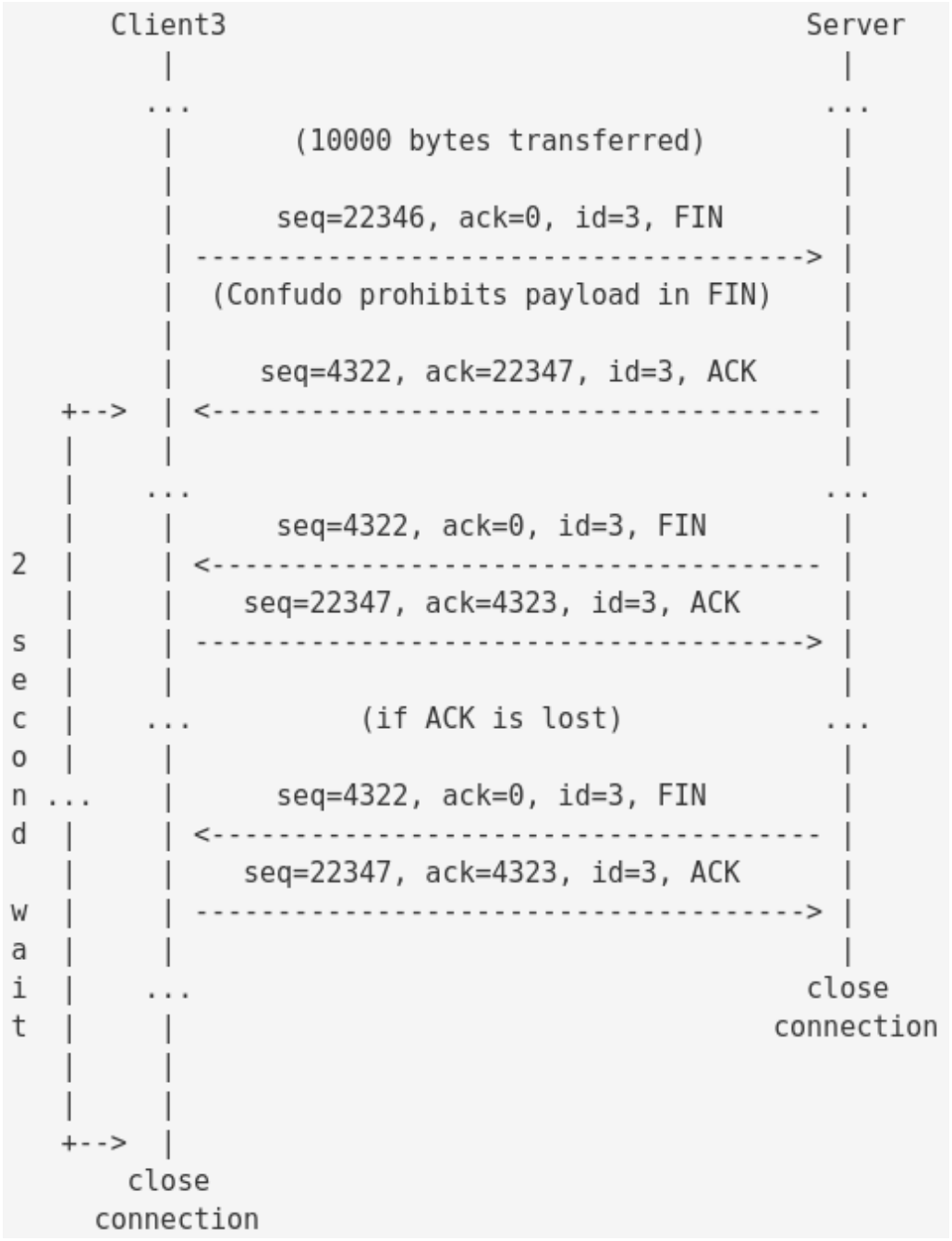
  Esperar por 2 segundos por pacotes com a flag FIN (FIN-WAIT)

  Durante a espera, responder cada FIN que chega com um pacote ACK; descartar qualquer outro pacote não FIN.

  Encerrar conexão e terminar execução normalmente.

  Cliente deve suportar transferência de arquivo de até 100MB.

  Toda vez que o cliente não receber pacotes do servidor por 10 segundos, ele deve abortar a conexão (fechar socket e sair com mensagem de erro.)


Client3
Server
(10000 bytes transferred)
seq=22346, ack=0, id=3, FIN
(Confudo prohibits payload in FIN)
seq=4322, ack=22347, id=3, ACK
seq=4322, ack=0, id=3, FIN
seq=22347, ack=4323, id=3, ACK
(if ACK is lost)
seq=4322, ack=0, id=3, FIN
seq=22347, ack=4323, id=3, ACK
close connection
close connection
2 second wait

0.4 Requisitos de controle de congestionamento

A lógica de controle de congestionamento a ser implementada é a do TCP Tahoe:

- Depois da conexão estabelecida, o cliente deve enviar até CWND bytes de dados sem ter de esperar por ACKs.
- Para cada ACK recebido:

  (Partida lenta) Se CWND $<$ SS-THRESH : CWND $+=$ 512.

  (Prevenção de congestionamento) Se CWND $>=$ SS-THRESH : CWND $+=$ $(512 * 512)/\text{CWND}$.
- Depois de estouro de tempo, defina SS-THRESH como CWND, e CWND como 512, retransmita os dados após o último byte reconhecido.
- Para cada pacote válido da conexão, exceto pacotes com somente a flag ACK e payload vazio, o servidor responde com pacotes ACK.

0.5 Dicas

Para emular perda de pacotes e outros comportamentos na rede, siga o tutorial: Linux - https://wiki.linuxfoundation.org/networking/netem ; Windows - http://wanem.sourceforge.net/ ou http://jagt.github.io/clumsy/index.html .