

Introdução

Este projeto avaliará o domínio da API socket. A linguagem a ser utilizada no trabalho é C. O trabalho é, no máximo, em dupla. O trabalho está dividido em duas partes, independentes.

Parte 1.

Na primeira parte você criará um cliente que se comunicará com um servidor segundo um protocolo específico. A tarefa do servidor é validar que o cliente está seguindo o protocolo. Qualquer desvio que o cliente fizer, do protocolo estabelecido, o servidor fechará a conexão. O cliente e o servidor comunicar-se-ão via sockets UDP e TCP. A seguir a descrição do protocolo, em estágios. Lembre-se, o protocolo deve ser seguido à risca.

Protocolo

O servidor escutará por pacotes UDP na porta 12235. O servidor espera receber e irá enviar somente:

- Dados que contenham cabeçalho, veja abaixo.
- Dados com bytes em ordem da rede (big-endian).
- Inteiros sem sinal de 4 bytes(`uint32_t`) ou de 2 bytes(`uint16_t`).
- Caracteres (1 byte).
- Strings, que são sequências de caracteres finalizando com caractere `'\0'`.
- Pacotes devem ser preenchidos até o comprimento ser divisível por 4.

O servidor fechará qualquer socket aberto com um cliente e/ou falhará em responder ao cliente se:

- Número inesperado de pacotes foi recebido.
- Dados inesperados, comprimento do pacote ou dos dados inesperado.
- O servidor não receber qualquer pacote do cliente em 3 segundos.

Cada dado, enviado por TCP ou UDP, ao/do servidor deve ter o cabeçalho. Este cabeçalho deve estar prefixado aos dados. O cabeçalho tem comprimento, constante, de 12 bytes. Os primeiros 4 bytes do cabeçalho contém o comprimento dos dados que o pacote carrega, excluindo-se qualquer preenchimento necessário para tornar o comprimento do pacote divisível por 4. Os próximos 4 bytes contém o segredo de **estágio prévio** do protocolo. Os próximos dois bytes contém um inteiro indicativo do estágio corrente do protocolo. Por exemplo, para o passo c1, os primeiros 4 bytes do cabeçalho conterão o comprimento do pacote, os próximos 4 bytes conterão `secretB` e os dois bytes seguintes terão o valor 1. **Para o lado cliente, o passo sempre será o número 1, para o servidor será 2.** Para o estágio a, `psecret` é definido 0. Os últimos dois bytes do cabeçalho deve ser preenchido com um inteiro formado pelos últimos 3 números da matrícula de um aluno. Diagramas de pacotes serão utilizados através da descrição do protocolo. Abaixo o formato do cabeçalho do pacote para a primeira parte:

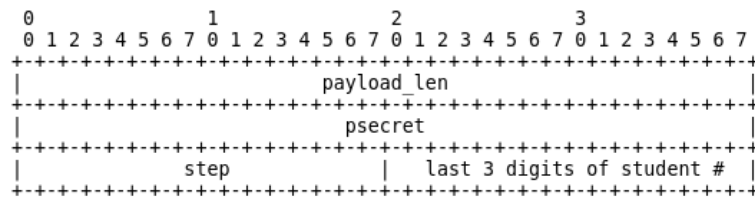


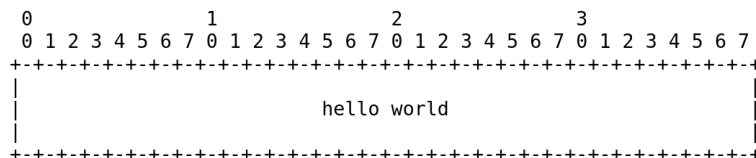
Figura 1: Cabeçalho.

O cabeçalho é omitido nos pacotes seguintes, para eliminar redundância, **mas lembre-se todo pacote tem o cabeçalho acima.**

Estágio a.

Passo a1.

O cliente envia um único pacote UDP contendo a string "hello world", sem as aspas, para o servidor que estará escutando em 12235.



Obs: hello world não ocupa 4 bytes.

Passo a2.

O servidor responde com um pacote UDP contendo quatro inteiros: num, len, udp_port, secretA:



Estágio b.

Passo b1.

O cliente envia **num** pacotes UDP para o servidor na porta **udp_port**. Cada um desses pacotes de dados tem comprimento **len + 4**. Lembre-se que a carga de dados deve estar alinhada a 4 bytes. Os primeiros 4 bytes de dados de cada pacote deve ser um inteiro identificando o pacote. O primeiro pacote deve ter esse inteiro definido como 0, enquanto o último pacote deve ter esse inteiro definido como -1 . O restante dos bytes no pacote(len), devem ser 0s.

Para cada pacote de dados recebido, o servidor irá reconhecer(ack) respondendo com um pacote "ack" que contém como dado o identificador do pacote a ser reconhecido.

Para completar este passo, o cliente **deve** receber pacotes ack do servidor para todos num pacotes gerados.

Estágio d.

Passo d1.

O cliente envia num2 pacotes, cada com dados de comprimento len2, com todos os bytes definidos como o caractere informado pelo servidor no pacote enviado no passo c2.

```

      0             1             2             3
      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-----+-----+-----+-----+
|                                     |
|      payload of length len2 filled with char c      |
|                                     |
+-----+-----+-----+-----+
```

Passo d2.

O servidor responde com um inteiro: secretD.

```

      0             1             2             3
      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-----+-----+-----+-----+
|                                     |
|               secretD               |
|                                     |
+-----+-----+-----+-----+
```

Parte 2.

Na parte 2 você escreverá um servidor que manipula a comunicação com clientes. A tarefa do servidor é verificar se os clientes seguem o protocolo de comunicação. Ele deve estar haptó a se comunicar com diversos clientes ao mesmo tempo.

Protocolo

Seguiremos o mesmo protocolo da parte 1 e, portanto, você poderá usar seu cliente da parte 1 para verificar seu servidor. O servidor deve receber e enviar apenas:

- Dados que contêm cabeçalho, veja abaixo.
- Dados com bytes em ordem da rede (big-endian).
- Inteiros sem sinal de 4 bytes(`uint32_t`) ou de 2 bytes(`uint16_t`).
- Caracteres (1 byte).
- Strings, que são sequências de caracteres finalizando com caractere `'\0'`.
- Pacotes devem ser preenchidos até o comprimento ser divisível por 4.

Cada dado, enviado por TCP ou UDP, ao/do servidor deve ter o cabeçalho. Este cabeçalho deve estar prefixado aos dados. O cabeçalho tem comprimento, constante, de 12 bytes. Os primeiros 4 bytes do cabeçalho contém o comprimento dos dados que o pacote carrega, excluindo-se qualquer preenchimento necessário para tornar o comprimento do pacote divisível por 4. Os próximos 4 bytes contém o segredo de **estágio prévio** do protocolo. Os próximos dois bytes contém um inteiro indicativo do estágio corrente do protocolo. Por exemplo, para o passo c2, os primeiros 4 bytes do cabeçalho conterão o comprimento do pacote, os próximos 4 bytes conterão secretB e os dois bytes seguintes terão o valor 2. **Para o lado cliente, o passo sempre será**

o número 1, para o servidor será 2. Para o estágio a, psecret é definido 0. Os últimos dois bytes do cabeçalho deve ser preenchido com um inteiro formado pelos últimos 3 números da matrícula de um aluno. Diagramas de pacotes serão utilizados através da descrição do protocolo. Abaixo o formato do cabeçalho do pacote para a segunda parte, igual ao da primeira:

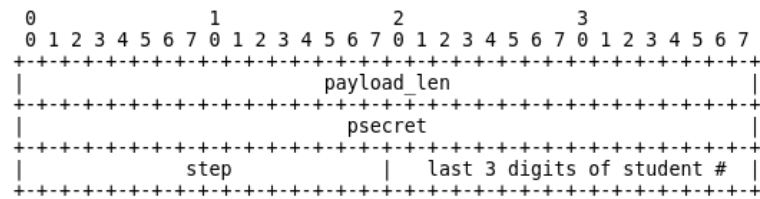


Figura 2: Cabeçalho.

O cabeçalho é omitido nos pacotes seguintes, para eliminar redundância, **mas lembre-se todo pacote tem o cabeçalho acima.**

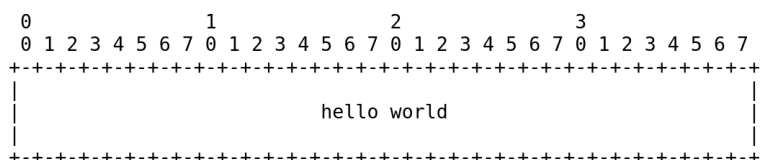
Servidor

1. O servidor deve escutar conexões pela porta 12235.
2. O servidor deve manipular diversos clientes ao mesmo tempo. Isso pode ser feito através de threads. Crie uma nova thread para cada novo cliente, mate-a quando finalizar a comunicação com o cliente.
3. O servidor deve verificar cada cabeçalho recebido e fechar qualquer conexão com o cliente ou deixar de responder se:
 - Número inesperado de pacotes recebido.
 - Dados inesperados, comprimento do pacote ou dos dados inesperado.
 - O servidor não receber qualquer pacote do cliente em 3 segundos.
 - O servidor não recebe o segredo correto.
4. O servidor responde ao cliente em quatro estágios. Em cada estágio, o servidor deve gerar aleatoriamente um segredo a ser enviado ao cliente.

Estágio a.

Passo a1.

O cliente envia um único pacote UDP contendo a string "hello world", sem as aspas, para o servidor que estará escutando em 12235.



Obs: hello world não ocupa 4 bytes.

O servidor deve verificar os dados e responder com um pacote UDP contendo quatro inteiros: num, len, udp_port, secretA. Todos esses números devem ser gerados aleatoriamente. O servidor deve aguardar por pacotes do cliente na porta udp_port.

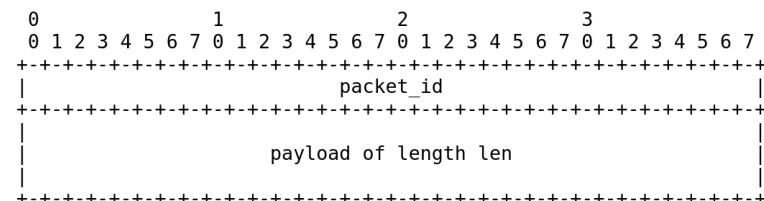


Estágio b.

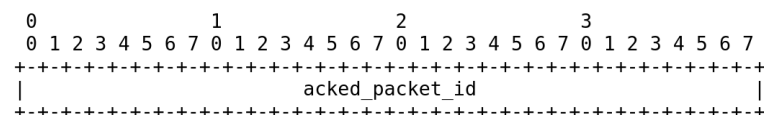
Passo b1.

O cliente transmitirá num pacotes UDP para o servidor escutando da porta `udp_port`. O servidor tem que verificar:

- Cada um dos pacotes tem comprimento $\text{len} + 4$.
- Os primeiros 4 bytes de dados de cada pacote deve ser um inteiro identificando o pacote.
- O primeiro pacote deve ter esse inteiro definido como 0, enquanto o último pacote deve ter esse inteiro definido como -1 .
- Os pacotes chegam em ordem, com cada ACK enviado antes de receber um novo pacote.
- O restante dos bytes no pacote(len), devem ser 0s.



Para cada pacote de dados recebido, o servidor decide aleatoriamente se confirma (ack) aquele pacote respondendo com um pacote “ack” que contém como dado o identificador do pacote reconhecido:



O servidor deve receber o mesmo pacote novamente se decidir não enviar uma confirmação (certifique-se de que seu servidor não envie um ack pelo menos uma vez para toda a transação). Esta etapa será concluída após o servidor receber num pacotes corretamente em ordem.

Passo b2.

Uma vez que o servidor recebe todos os num pacotes, ele deve enviar um pacote UDP contendo dois inteiros: um número de porta TCP, secretB. Agora o servidor deve esperar por uma conexão TCP do cliente no número da porta TCP.

```

      0           1           2           3
      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     tcp_port                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     secretB                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Estágio c.

Passo c1.

O cliente abrirá uma conexão TCP com o servidor na porta `tcp_port` recebida do seu servidor na etapa b2.

Passo c2.

Seu servidor deve enviar três inteiros: `num2`, `len2`, `secretC` e um caractere: `c`, escolhido de forma aleatória no alfabeto.

```

      0           1           2           3
      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     num2                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     len2                             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     secretC                         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          c          |
+---+---+---+---+---+

```

Estágio d.

Passo d1.

Os clientes enviam `num2` pacotes, cada um com carga útil de comprimento `len2` e cada carga útil contendo todos os bytes configurados para o caractere `c`.

```

      0           1           2           3
      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     |
|                                     |
|          payload of length len2 filled with char c          |
|                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Passo d2.

O servidor responde com um inteiro: `secretD`.

```

      0           1           2           3
      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     secretD                         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Considerações finais.

No arquivo enviado pelo sigaa coloque o nome e matrícula dos integrantes da dupla. Cada parte deve estar programada em arquivos fonte distinto e em pastas distintas. Tudo deve ser enviado em um arquivo compactado com .zip.

As notas serão dadas para cada estágio completo, de ambas as partes.