Descrição da tarefa de programação a ser feita na disciplina de Redes de Alto Desempenho (RAD) SSC-144. Turmas A e B.

A tarefa de programação é referente ao Capítulo 7 do Livro: Redes de Computadores e a Internet: Uma abordagem top-down. (Kurose & Rosss, 2006).

Streaming vídeo com RTSP e RTP

1. O código

Neste laboratório, você implementará um servidor de streaming vídeo e cliente que se comunica usando o protocolo de fluxo contínuo em tempo real (RTSP) e envia dados usando o protocolo de tempo real (RTP). Sua tarefa é implementar o protocolo RTSP no cliente e implementar o empacotamento RTP no servidor.

Forneceremos o código que implementa o protocolo RSTP no servidor, o desempacotamento RTP no cliente e trataremos de exibir o vídeo transmitido. Você não precisa mexer neste código.

2. Classes

Existem quatro classes nesta tarefa.

Client

> Esta classe implementa o cliente e a interface de usuário que você usará para enviar comandos RTSP e que será utilizada para exibir o vídeo. Abaixo vemos como é a interface. *Você deverá implementar as ações que são tomadas quando os botões são pressionados.*

Server

> Esta classe implementa o servidor que responde às requisições RTSP e encaminha o vídeo de volta. A interação RTSP já está implementada e o servidor chama as rotinas na classe `RTPpacket` para empacotar os dados de vídeo. Você não precisa mexer nesta classe.

RTPpacket

> Esta classe é usada para manipular os pacotes RTP. Ela possui rotinas separadas para tratar os pacotes recebidos no lado cliente que já é dado e você não precisa modificá-lo (mas veja os Exercícios opcionais). Você deverá completar o primeiro construtor desta classe para implementar o empacotamento RTP dos

dados de vídeo. O segundo construtor é usado pelo cliente para desempacotar os dados. Você não precisa modificá-lo também.

VideoStream

Esta classe é usada para ler os dados de vídeo do arquivo em disco. Você não precisa modificar esta classe.

3. Executando o código

Após completar o código, você pode executá-lo da seguinte forma:

Primeiro, inicie o servidor com o comando:

```
java Server server_port
```

onde `server_port` é a porta onde seu servidor escuta as conexões RTSP que chegam. A porta RTSP padrão é a 554, mas você deve escolher um número de porta maior que 1024.

Então, inicie o cliente com o comando:

```
java Client server_name server_port video_file
```

onde `server_name` é o nome da máquina onde o servidor está sendo executado, `server_port` é a porta que o servidor está escutando, e `video_file` é o nome do arquivo que você quer requisitar (fornecemos um arquivo de exemplo `movie.Mjpeg`). O formato do arquivo está descrito no Apêndice.

O cliente abre uma conexão com o servidor e abre uma janela como esta:

Você pode enviar comandos RTSP para o servidor pressionando os botões. Uma interação normal RTSP acontece assim:

1. O cliente envia SETUP. Esse comando é usado para ajustar os parâmetros de sessão e de transporte.
2. O cliente envia PLAY. Isso inicia a reprodução.
3. O cliente pode enviar PAUSE se ele quiser pausar durante a reprodução.
4. O cliente envia TEARDOWN. Isso termina a sessão e fecha a conexão.

O servidor sempre responde a todas as mensagens que o cliente envia. Os códigos de resposta são exatamente os mesmos do HTTP. O código 200 indica que a requisição foi bem-sucedida. Neste laboratório, você não precisa implementar nenhum outro código de resposta. Para mais informações sobre o RTSP, veja a RFC-2326.

4. Cliente

Sua primeira tarefa é implementar o RTSP do lado cliente. Para fazer isso, você deve completar as funções que são chamadas quando o usuário clica nos botões da interface de usuário. Para cada botão na interface, há uma função manipuladora do código. Você deve implementar as seguintes ações em cada função manipuladora.

Quando o cliente é iniciado, ele também abre o socket RTSP para o servidor. Use este socket para enviar todas as requisições RTSP.

SETUP

- Crie um socket para receber os dados RTP e ajustar o tempo de expiração no socket para 5 milissegundos.
- Envie uma requisição SETUP para o servidor. Você deve inserir o cabeçalho de Transporte onde você especificará a porta para o socket de dados RTP que você criou.
- Leia a resposta do servidor e analise o cabeçalho de Sessão na resposta para obter o ID da sessão.

PLAY

- Envie uma requisição PLAY. Você deve inserir o cabeçalho de sessão e usar o ID de sessão fornecido na resposta ao SETUP. Não coloque cabeçalho de Transporte nesta requisição.
- Leia a resposta do servidor.

PAUSE

- Envie uma requisição PAUSE. Você deve inserir o cabeçalho de Sessão e usar o ID de sessão fornecido na resposta ao SETUP. Não coloque cabeçalho de Transporte nesta requisição.
- Leia a resposta do servidor.

TEARDOWN

- Envie uma requisição TEARDOWN. Você deve inserir o cabeçalho de Sessão e usar o ID de sessão fornecido na resposta ao SETUP. Não é preciso colocar cabeçalho de Transporte neste requisição.
- Leia a resposta do servidor.

Nota: Você deve inserir o cabeçalho CSeq em cada requisição que você enviar. O valor do cabeçalho CSeq é um numero que será incrementado de 1 a cada requisição que você enviar.

Exemplo

Aqui está uma interação de amostra entre cliente e servidor. As requisições dos clientes são marcadas com C: e as respostas dos servidores com S:. Neste laboratório, tanto o cliente quanto o servidor são bem simples. Eles não precisam ter rotinas de análise sofisticadas e esperam sempre encontrar os campos do cabeçalho na ordem que você verá abaixo, ou seja, numa requisição, o primeiro cabeçalho é o CSeq, e o segundo é ou o de Transporte (para SETUP) ou o de Sessão (para todas as outras requisições). Na resposta, CSeq é novamente o primeiro e de Sessão é o segundo.

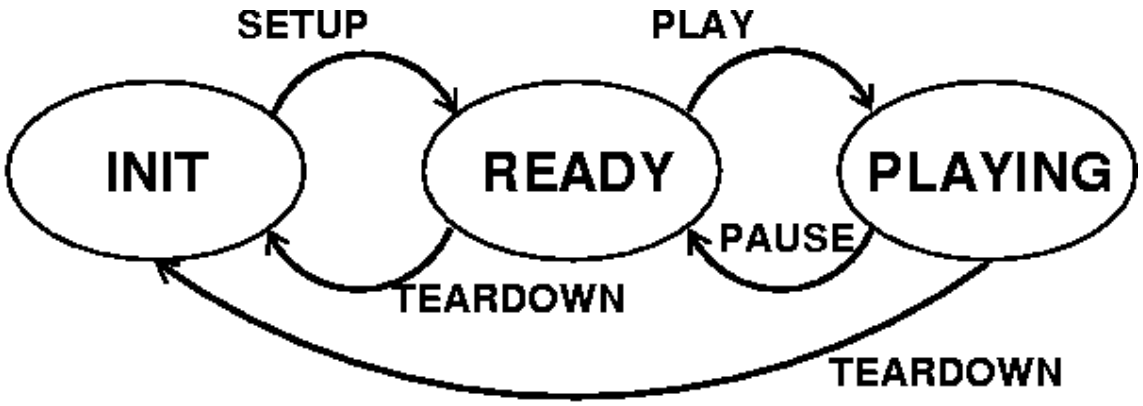
```
C: SETUP movie.Mjpeg RTSP/1.0
C: CSeq: 1
C: Transport: RTP/UDP; client_port= 25000
```

```
S: RTSP/1.0 200 OK
S: CSeq: 1
S: Session: 123456

C: PLAY movie.Mjpeg RTSP/1.0
C: CSeq: 2
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 2
S: Session: 123456

C: PAUSE movie.Mjpeg RTSP/1.0
C: CSeq: 3
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 3
S: Session: 123456

C: PLAY movie.Mjpeg RTSP/1.0
C: CSeq: 4
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 4
S: Session: 123456

C: TEARDOWN movie.Mjpeg RTSP/1.0
C: CSeq: 5
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 5
S: Session: 123456
```

Estado do cliente

Uma das diferenças entre HTTP e RTSP é que no RTSP cada sessão possui um estado. Neste laboratório, você precisará manter o estado do cliente atualizado. O cliente muda

de estado quando ele recebe uma resposta do servidor de acordo com o seguinte diagrama.



5. Servidor

No servidor, você precisará implementar o empacotamento dos dados de vídeo em pacotes RTP. Para isso, será necessário criar o pacote, ajustar os campos no cabeçalho do pacote e copiar a carga útil (exemplo: um quadro de vídeo) dentro do pacote.

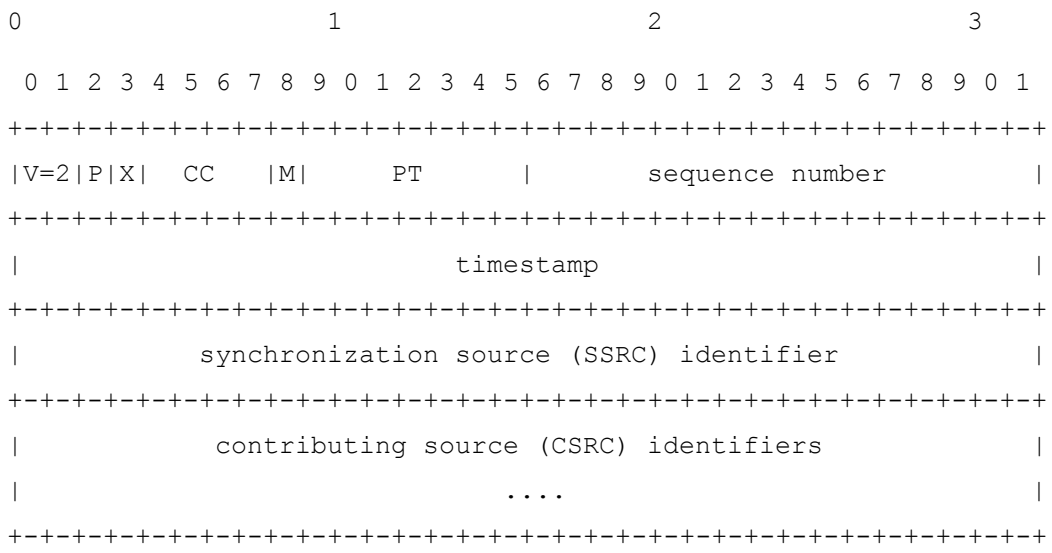
Quando o servidor recebe a requisição PLAY do cliente, ele aciona um temporizador que é ativado a cada 100ms. Nesses tempos, o servidor lerá um quadro de vídeo do arquivo e o enviará para o cliente. O servidor cria um objeto RTPpacket, que é o encapsulamento RTP do quadro de vídeo.

O servidor chama o primeiro construtor da classe RTPpacket para realizar o encapsulamento. Sua tarefa é escrever essa função. Você precisará fazer o seguinte: (as letras em parênteses se referem aos campos no formato do pacote RTP abaixo)

1. Ajuste o campo RTP-version (V). Você deve ajustá-lo para 2.
2. Ajuste os campos padding (P), extension (X), number of contributing sources (CC), e marker (M). Todos eles são ajustados em zero neste laboratório.
3. Ajuste o campo carga útil (PT). Neste laboratório, usamos MJPEG e o tipo para ele é 26.
4. Ajuste o número de seqüência. O servidor fornece esse número de seqüência como argumento `Framenb` para o construtor.
5. Ajuste a marca de tempo. O servidor fornece este número como argumento `Time` para o construtor.

6. Ajuste o identificador da fonte (SSRC). Este campo identifica o servidor. Você pode usar o valor inteiro que desejar.

Como não temos nenhuma outra fonte de contribuição (campo CC == 0), o campo CSRC não existe. O comprimento do cabeçalho do pacote é de 12 bytes, ou as três primeiras linhas do diagrama abaixo.



Você deve preencher o cabeçalho na disposição `header` da classe RTPpacket. Você também precisará copiar a carga útil (fornecida como argumento `data`) para a variável `payload`. O comprimento da carga útil é dado no argumento `data_length`.

O diagrama acima está na ordem de byte de rede (também conhecido como big-endian). A Java Virtual Machine usa a mesma ordem de byte, então você não precisa transformar seu cabeçalho de pacote na ordem de byte de rede.

Para mais detalhes sobre RTP, veja a RFC-3550.

Manipulando os bits

Aqui estão alguns exemplos de como ajustar e verificar bits individuais ou grupo de bits. Note que no formato do cabeçalho do pacote RTP, os menores números de bit se referem a maiores ordens de bits, ou seja, o bit número 0 de um byte é 2^7, e o bit número 7 é 1 (ou 2^0). Nos exemplos abaixo, os números de bit se referem aos números no diagrama acima.

Como o campo `header` da classe `RTPpacket` é um vetor do tipo `byte`, você precisará ajustar o cabeçalho de um byte por vez, que é um grupo de 8 bits. O primeiro byte possui os bits 0-7, o segundo byte possui os bits 8-15, e assim por diante. Em Java, um int tem 32 bits ou 4 bytes.

Para ajustar o número *n* na variável `mybyte` do tipo byte:

```
mybyte = mybyte | 1 << (7 – n);
```

Para ajustar os bits *n* e *n+1* para o valor de `foo` na variável `mybyte`:

```
mybyte = mybyte | foo << (7 – n);
```

Note que `foo` deve ter um valor que possa ser expresso com 2 bits, ou seja, 0, 1, 2 ou 3.

Para copiar um `foo` inteiro de 16-bits em 2-bytes, `b1` e `b2`:

```
b1 = foo >> 8;
b2 = foo & 0xFF;
```

Após fazer isso, `b1` terá 8 bits de maior ordem de `foo` e `b2` terá 8 bits de menor ordem de `foo`.

Você pode copiar um inteiro de 32-bits em 4-bytes de maneira similar.

Se você não se sente confortável com o ajuste de bits, pode encontrar mais informações no Java Tutorial.

Exemplo de bit

Suponha que desejamos preencher o primeiro byte do cabeçalho do pacote RTP com os seguintes valores:

- V = 2
- P = 0
- X = 0
- CC = 3

Em binário, isso poderia ser representado como

```
1 0 | 0 | 0 | 0 0 1 1
V=2   P   X   CC = 3

2^7 . . . . . . . 2^0
```

Exercícios:

- Calcule as estatísticas sobre a sessão. Você precisará calcular a taxa de perda de pacotes RTP, a taxa de dados de vídeo (em bits ou bytes por segundo) e qualquer outra estatística interessante que você conseguir encontrar.

- A interface de usuário no cliente possui 4 botões para as 4 ações. Se você compará-la a um transdutor padrão, tal como RealPlayer ou transdutor Windows, você verá que eles possuem apenas 3 botões para as mesmas ações, chamadas, PLAY, PAUSE e STOP (correspondendo exatamente ao TEARDOWN). Não há nenhum botão de SETUP disponível para o usuário. Dado que o SETUP é mandatório numa interação RTSP, como você implementaria isso? É apropriado enviar TEARDOWN quando o usuário clica no botão *stop*?

- Até aqui, o cliente e o servidor implementam apenas o mínimo necessário de interações RTSP e PAUSE. Implemente o método DESCRIBE, que é usado para passar informações sobre o *media stream*. Quando o servidor recebe uma requisição DESCRIBE, ele envia de volta um arquivo de descrição de sessão que diz ao cliente que tipos de streams estão na sessão e quais codificações estão sendo utilizadas.

Apêndice

Formato do MJPEG (Motion JPEG) proprietário do laboratório.

Neste laboratório, o servidor encaminha um vídeo codificado com um formato de arquivo proprietário MJPEG. Este formato armazena o vídeo como concatenação de imagens codificadas em JPEG, com cada imagem sendo precedida por um cabeçalho de 5-bytes que indica o tamanho em bits da imagem. O servidor analisa o bitstream do arquivo MJPEG para extrair as imagens JPEG no caminho. O servidor envia as imagens ao cliente em intervalos periódicos. O cliente então exibe as imagens JPEG individuais conforme elas vão chegando do servidor.

REFERÊNCIA:

(Kurose & Ross, 2006) Kurose, James F. Ross, Keith W. – Redes de Computadores e a Internet: uma abordagem top-down. 3ed. – São Paulo, Pearson Addison Wesley, 2006.