

UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COLEGIADO DE INFORMÁTICA

Disciplina: Redes de Computadores
Professor: Luiz Antonio

Data Entrega: 30/junho/2008
Realização: em duplas.

Trabalho do 4º Bimestre - Streaming vídeo com RTSP e RTP

O código

Este trabalho consiste na implementação de um servidor de streaming de vídeo e um cliente que se comunicam usando o protocolo de fluxo contínuo em tempo real (RTSP) e enviam dados usando o protocolo de tempo real (RTP). Sua tarefa é implementar o protocolo RTSP no cliente e adicionar algumas funcionalidades descritas no final da especificação.

Classes

São sugeridas quatro classes nesta tarefa.

`Cliente.java`

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.*

`Servidor.java`

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 `PacoteRTP` para empacotar os dados de vídeo.

`PacoteRTP.java`

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.

VideoStream.java

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

Executando o código

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

Primeiro inicie o servidor com o comando:

```
java Servidor 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 Cliente 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 (o arquivo exemplo `movie.Mjpeg` foi disponibilizado para teste). 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.

1. 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 UDP 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 nesta 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:. 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.mpeg 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

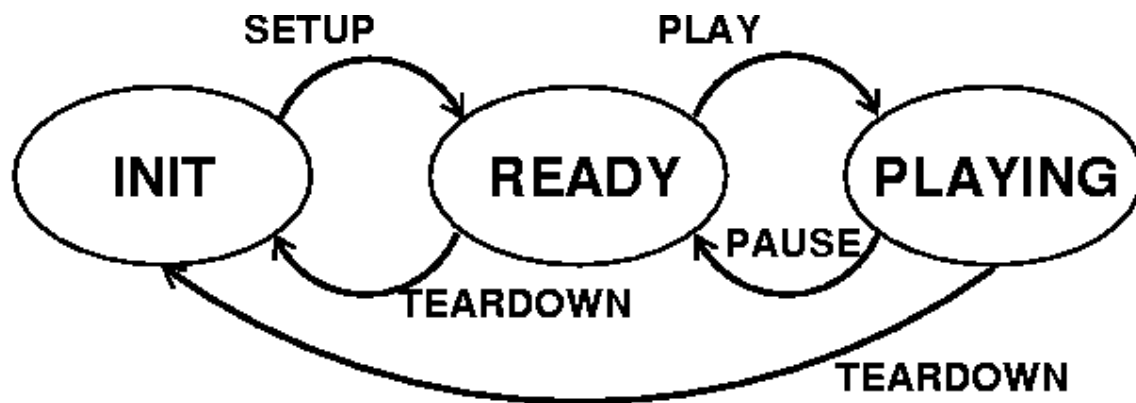
S: RTSP/1.0 200 OK

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. No trabalho, 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.



2. Servidor

No servidor está implementado o empacotamento dos dados de vídeo em pacotes RTP. Para isso, foi criado o pacote, ajustados os campos no cabeçalho do pacote e copiada a carga útil (exemplo: um quadro de vídeo) dentro do pacote.

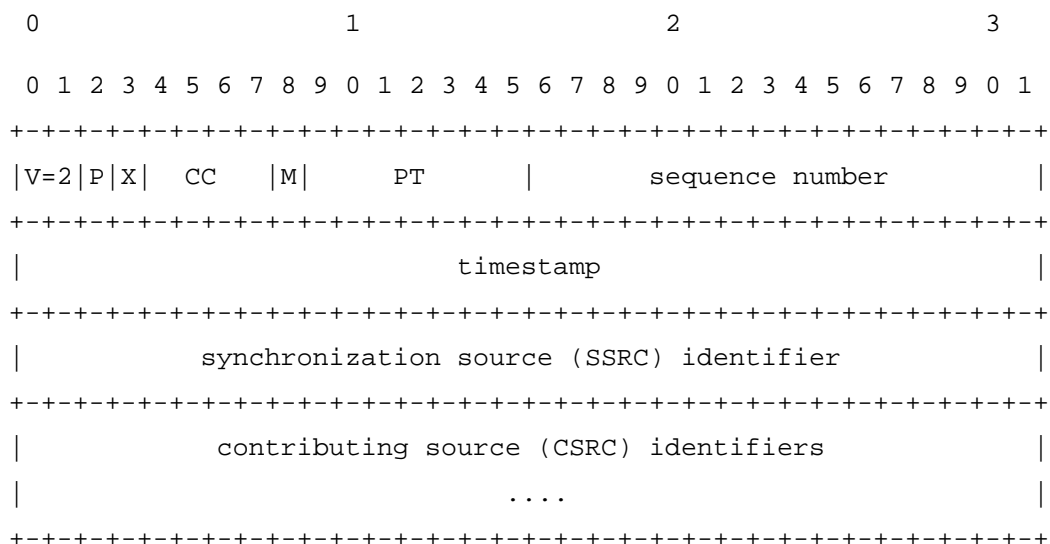
Quando o servidor recebe a requisição **PLAY** do cliente, ele aciona um temporizador que é ativado a cada 100ms. Nesses tempos, o servidor lê um quadro de vídeo do arquivo e o envia para o cliente. O servidor cria um objeto **PacoteRTP**, que é o encapsulamento RTP do quadro de vídeo.

O servidor chama o primeiro construtor da classe **PacoteRTP** para realizar o encapsulamento. Os seguintes campos são ajustados (veja a RFC 1889 para maiores detalhes):

1. RTP-version (V) com valor 2.

2. Padding (P), extension (X), number of contributing sources (CC), e marker (M). Todos eles são ajustados em zero.
3. Carga útil (PT). Neste trabalho usamos MJPEG e o tipo para ele é 26.
4. Número de sequência. O servidor fornece esse número de sequência 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.

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.



A carga útil (fornecida como argumento `data`) é copiada 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-1889.

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

Objetivos do Trabalho

- Converta o Servidor para atender a vários clientes simultaneamente.
- 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, 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, ajuste o Cliente para disponibilizar ao usuário apenas os botões PLAY, PAUSE e 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.