Avaliação de um Sistema Distribuído de Criação e Apresentação de Documentos Multimídia

Roberta Lima Gomes, Eduardo Carneiro da Cunha,
Luiz Fernando Rust da Costa Carmo, Luci Pirmez

beta@posgrad.nce.ufrj.br
{educc,rust,luci}@nce.ufrj.br

Núcleo de Computação Eletrônica – UFRJ
Caixa Postal 2324, 20001-970
Rio de Janeiro – RJ – Brasil

Resumo

A apresentação de um documento multimídia distribuído envolve a recuperação de objetos a partir de um ou mais servidores e a apresentação destes nos sistemas clientes. A recuperação desses objetos é influenciada pela quantidade de recursos oferecidos pela rede e deve ser realizada de acordo com a especificação dos relacionamentos entre os objetos. Este trabalho apresenta um estudo sobre o desempenho de um ambiente propício para a criação e recuperação de documentos multimídia adaptativos em redes corporativas. O objetivo final consiste em possibilitar uma apresentação coordenada de um documento multimídia, garantindo sempre a preservação da coerência entre os diferentes fluxos de mídias, mesmo quando o processamento é confrontado com uma insuficiência temporária de recursos oferecidos pela rede de comunicação.

Palavras-chave: Sistemas multimídia distribuídos, sistemas adaptativos, educação a distância.

Abstract

A distributed multimedia document presentation involves retrieval of objects from one or more document servers and their presentation at the client systems. The retrieval of these objects is influenced by the amount of resources offered by the network, and has to be carried out in accordance with the specification of object relationships. This work presents performance analyses of an appropriate environment for creation and retrieval of adaptive multimedia documents in corporate networks. The final purpose consists of providing a coordinate multimedia document presentation, always guaranteeing the preservation of coherence between the different media, even when the process is confronted with a temporary lack of communication resources.

Keywords: Distributed multimedia systems, adaptive systems, distance learning.

1 INTRODUÇÃO

O surgimento das redes de computadores de alta velocidade, das tecnologias de armazenamento e de comunicação, e dos equipamentos periféricos de áudio e vídeo digitais incentivou o rápido crescimento da pesquisa e desenvolvimento de sistemas multimídias. Os sistemas multimídia são sistemas computacionais que manipulam de forma integrada vários tipos de mídias de representação da informação. Frequentemente, sistemas multimídia são distribuídos (SMD), isto é, seus componentes estão localizados em diferentes nós de processamento numa rede local ou de longa distância.

A apresentação de um documento multimídia distribuído envolve a recuperação de objetos a partir de um ou mais servidores e a apresentação destes nos sistemas clientes. A recuperação dos objetos multimídia a partir dos servidores de documentos é influenciada por fatores dinâmicos, como atraso, variação de atraso e perdas de pacotes, que são verificados no interior da rede de comunicação, ao longo do caminho que interconecta os clientes e servidores. Devido à esta natureza heterogênea e variável das infra-estruturas de redes globais, satisfazer completamente os requisitos de Qualidade de Serviço (QoS) e Qualidade de Percepção (QoP) [Ghin99] de uma apresentação multimídia pode ser uma tarefa complexa. A reserva de recursos e as técnicas de admissão convencionais não são capazes de garantir uma qualidade de serviço sem um considerável desperdício de recursos e uma ineficiente utilização dos mesmos [Ferg98]. Uma forma de se resolver este problema e evitar qualquer impacto adverso sobre o usuário final é tornar as aplicações distribuídas e suas infra-estruturas adaptativas. Desta forma, as aplicações passam a tolerar flutuações na disponibilidade dos recursos ou a infra-estrutura de rede torna-se capaz de se moldar às mudanças dinâmicas nos requisitos das aplicações.

Cristina Aurrecoechea [Aurr98] descreve uma variedade de mecanismos de adaptação da qualidade de serviço. Por exemplo, filtros de QoS localizados em pontos específicos na rede que transcodificam os fluxos que passam por eles de acordo com a capacidade do *link* e os recursos do cliente; mecanismos de adaptação da taxa de envio do emissor, e camadas incrementais de QoS enviadas por *multicast* (*Layered Multicast*). Entretanto, todos esses mecanismos se preocupam apenas com parâmetros de tráfego específicos e com a percepção direta do usuário com relação a mídias isoladas. A apresentação de um documento multimídia, considerando todos os seus fluxos de mídias distintas e suas relações semânticas, não é tratada como um todo.

O objetivo deste trabalho é avaliar o desempenho da estratégia de adaptação para documentos multimídia distribuídos proposta pelo projeto ServiMídia [Cunh99a, Cunh99c]. Basicamente, este projeto trata da autoria e armazenamento de documentos multimídia e da infra-estrutura de rede de comunicação para realizar a recuperação adaptativa destes documentos. O objetivo básico está na necessidade de se acomodar em um mesmo ambiente multimídia de ensino à distância, baseado no modelo cliente/servidor, usuários com diferenças substanciais na disponibilidade dos recursos de comunicação. O ponto chave nesta estratégia é que, para cada adaptação do documento ao longo de sua apresentação, as propriedades semânticas das diversas mídias que o compõem devem ser preservadas. Em outras palavras, o usuário que estiver assistindo à sua apresentação deve ser capaz de absorver o conteúdo informacional contido no documento, mesmo quando este sofrer alguma degradação de qualidade devido à falta de recursos verificada na rede.

O artigo está estruturado em 5 sessões. A segunda sessão descreve a proposta de autoria e apresentação de documentos multimídia adaptativos. Na terceira sessão é discutido o ambiente e as características dos testes implementados. Na quarta sessão são apresentados os resultados obtidos com o ambiente de testes. Por último, na quinta seção são apontadas algumas conclusões e possibilidades de trabalhos futuros.

2 COMPOSIÇÃO E APRESENTAÇÃO DE DOCUMENTOS MULTIMÍDIA ADAPTATIVOS

A maioria das propostas de adaptabilidade trata os objetos de uma mesma apresentação de forma independente. Essas adaptações ao nível dos objeto, ou ao nível da codificação, são limitadas a uma estrutura lógica estática do documento. Além disso, da maneira como estas

alternativas têm sido especificadas, a adaptação é realizada ao se apresentar uma ou mais alternativas de um objeto, mas não é possível interromper outros objetos como parte desta mesma adaptação. A estratégia de autoria adaptativa proposta em [Cunh99a, Cunh99b] tem como objetivo principal descrever formas alternativas para a apresentação do documento e os critérios de seleção, baseados em relacionamentos semânticos, que definem os momentos de ativá-las. Esses relacionamentos indicam como deverá ocorrer o processo de adaptação, de forma a manter a consistência desejada para o documento multimídia adaptado.

2.1 A Metodologia de Adaptação

Com o objetivo de suavizar o impacto das adaptações sobre os usuários e, ao mesmo tempo, garantir que a grande variedade de alternativas disponíveis para cada objeto possam ser utilizadas, foram adotados os dois níveis de adaptação descritos a seguir. No primeiro nível (adaptação suave) ocorre uma adaptação ao longo do eixo das fidelidades (resoluções) de um certo objeto. Neste nível, durante a apresentação do documento, um objeto de mídia codificado em um conjunto de diferentes resoluções pode ter a sua apresentação chaveada entre as resoluções deste conjunto. Neste nível de adaptação, os usuários não são incomodados pelo processo de adaptação pois a estrutura original do documento se mantém e apenas a qualidade das mídias é alterada.

No segundo nível (adaptação forte) ocorre uma adaptação ao longo do eixo das modalidades (tipos de mídia) de um certo objeto. Neste nível de adaptação, os usuários são capazes de perceber a ocorrência da adaptação, pois, ao se alterar o tipo de mídia de uma informação, dificilmente é mantida a estrutura original do documento. Porém, para que ocorra uma adaptação coerente da estrutura do documento, as relações de dependência condicional devem ser especificadas entre as diferentes alternativas.

As dependências condicionais (ou relacionamentos semânticos) [Court96] foram propostas para que se pudesse explorar o conhecimento de relações semânticas entre os diversos objetos de mídias. As dependências condicionais expressam relações de causalidade entre os objetos, ajudando a descrever os requisitos de seleção de um objeto com relação a outros objetos. A descrição desses relacionamentos permite que durante a adaptação de um documento multimídia seja realizada uma reestruturação coerente no formato do documento (de forma a preservar a QoP do mesmo).

2.2 Arquivos de Controle e a Linguagem SCL

A estratégia de autoria de documentos adaptativos adotada no Projeto ServiMídia baseia-se na especificação de dois arquivos: (i) um documento multimídia original, contendo uma apresentação com os maiores requisitos de QoS (do ponto de vista do autor); (ii) e um arquivo de controle (ou de anotação), possuindo informações necessárias para a realização do processo de adaptação.

O arquivo original é especificado na linguagem SMIL 1.0 [Smil98], uma linguagem declarativa para especificação de apresentações multimídia. Para a especificação do arquivo de controle, foi definida a linguagem *Smil Control Language* (SCL) [Gome00], baseada no XML [Bray98].

No arquivo de controle são descritos os requisitos de QoS dos elementos do documento original e as alternativas para estes elementos. Também são especificas as dependências condicionais entre esses objetos. As dependências condicionais são especificadas através da definição de *links*, formando uma malha de causalidade que descreve a consistência desejada

para o documento. SCL define dois tipos de *links*: *startlink* e *stoplink* (para apresentar e interromper, respectivamente).

A abordagem adotada (informações de controle externas ao arquivo SMIL) garante a interoperabilidade do ambiente com sistemas multimídia já existentes, além de simplificar o desenvolvimento de um *player* para apresentar os documentos adaptativos. Esta abordagem baseou-se no conceito de acesso universal (Universal Multimedia Access – UMA) adotado pelo grupo de desenvolvimento do MPEG-7 [MPEG7a, MPEG7b]. O objetivo dos sistemas de acesso universal é criar diferentes apresentações da mesma informação para atender diferentes formatos, equipamentos e redes, a partir de uma mesma base de informação.

2.3 Exemplo de Aplicação

Para ilustrar a execução dos dois mecanismos de adaptação, considere o documento multimídia descrito na figura 1.a. Neste documento SMIL o autor define a apresentação de um vídeo (“V1”). No respectivo arquivo SCL (figura 1.b), o autor descreve as seguintes informações necessárias para a realização dos dois níveis de adaptação: (i) a faixa de fidelidade para a apresentação do vídeo “V1” (adaptação suave); (ii) além de mídias alternativas (“anim1” e “text1”) e as condições para que as mesmas tornem-se ativas (adaptação forte).

Sob congestionamento, a apresentação deste documento é, inicialmente, submetida à adaptação suave, implicando na diminuição progressiva da taxa de apresentação do vídeo “V1” até um mínimo especificado pelo autor. Se o mecanismo de monitoramento do cliente continuar detectando o congestionamento no fluxo de “V1” (através de parâmetros do protocolo RTP/RTCP) será ativada a adaptação forte, implicando (i) na interrupção da apresentação do documento, (ii) criação (em tempo real) de um novo documento SMIL com base nas informações do arquivo SCL (figura 1.c), e apresentação deste novo documento a partir do mesmo instante em que o documento original foi interrompido.

a) Arquivo SMIL original

```
<smil>
...
 <body>
   <video id="V1" region="r_v"...
 </body>
</smil>
```

b) Arquivo SCL

```
<smal>
 <controls>
   <description about="V1">
     <minRequirements bw="64kbps">
      <maxRequirements bw="128kbps">
  </description>
 </controls>
 <alternatives>
  <replace target="V1">
    <startlink expr="(V1:stopped)">
    <regionToAdd>    ...
    </regionToAdd>
    <resourceToSubstitute>
       <par>
         <animation id="anim1" ... />
         <text id="text1" ... />
       </par>
     </resourceToSubstitute>
  </replace>
 </alternatives>
</smal>
```

c) Arquivo SMIL adaptado

```
<smil>
...
 <body>
   <par>
     <animation id="anim1" ... />
     <text id="text1" ... />
   </par>
 </body>
</smil>
```

Figura 1- Arquivos SMIL e SCL.

3 ARQUITETURA E CARACTERÍSTICAS DO SISTEMA PARA TESTES DA ESTRATÉGIA DE ADAPTAÇÃO

Com o intuito de verificar o comportamento geral da estratégia proposta em [Cunh99a, Cunh99c] e, em particular, o desempenho do mecanismo de adaptação da estrutura de um documento multimídia, foram realizados testes práticos em um ambiente de rede corporativa.

A implementação e os testes realizados se concentraram na avaliação do tempo de resposta do sistema a uma solicitação de adaptação de conteúdo (chaveamento de um fluxo). Assim, foram implementadas uma aplicação servidora, responsável pela transmissão dos fluxos de vídeo, e uma aplicação cliente, responsável pela apresentação dos vídeos e pela solicitação da adaptação das mídias. Esta solicitação corresponde, neste contexto de simulações e testes, tanto a uma solicitação para adaptação de fidelidade (nível 1) quanto a uma solicitação para adaptação da estrutura do documento (nível 2).

A decisão de simular o sistema com o cliente solicitando os dois níveis de adaptação foi tomada para que os intervalos e tempos de resposta do sistema pudessem ser registrados adequadamente e para que a simulação ocorresse de forma controlada. O sistema foi desenvolvido utilizando o ambiente de desenvolvimento de aplicações JBuilder 3.0 em conjunto com a plataforma Java 2 e com JMF 2.0 (*Java Media Framework 2.0* [JMF98]).

3.1 Características do JMF2.0

O JMF é uma API que permite aos programadores desenvolver aplicações em Java para capturar, processar e apresentar dados multimídia, além de prover suporte à transmissão e à recepção destes dados através da rede com o uso do protocolo RTP [Schu96].

No JMF, o processo de apresentação é modelado pela interface *Controller* (controlador), que define o mecanismo de controle de um objeto para processar, apresentar ou capturar uma mídia. São definidos dois tipos de objetos C*ontrollers*: *Players* e *Processors*. Ambos são construídos a partir de uma fonte de dados (*DataSource*) particular. Um *Player*, assim como um *Processor*, sinaliza os eventos de transição (*TransitionEvents*) conforme ele se move de um estado para outro. A interface *ControllerListener* provê uma forma de determinar em que estado um objeto *Player* se encontra.

Os objetos *Player* e *Processor* são utilizados para apresentar e processar os dados enviados pelos fluxos RTP. No JMF, um gerente de sessão (*SessionManager*) é utilizado para coordenar uma sessão RTP, como mostrado na figura 2. Este gerente da sessão mantém controle sobre seus participantes e sobre todos os fluxos estabelecidos dentro dela.

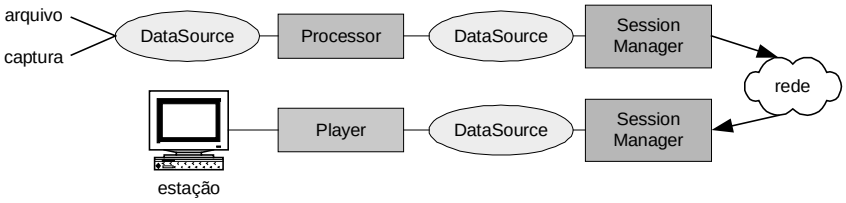


Figura 2 - Transmissão e recepção de fluxos RTP no JMF 2.0

No momento da criação e inicialização de um objeto *SessionManager*, os endereços local e remoto da sessão RTP são definidos. O gerente de sessão (*SessionManager*) suporta o estabelecimento de sessões *unicast* ou *multicast*. No momento da inicialização do objeto *SessionManager*, também é fornecido um nome canônico (*CNAME*) e um identificador de fonte de sincronização (*SSRC*). *CNAME* identifica o usuário local e *SSRC*, que é único na sessão, identifica o fluxo transmitido pelo objeto *SessionManager* se o *host* for um emissor.

O objeto *SessionManager* controla todos os participantes da sessão. Cada participante pode ser a origem de um ou mais fluxos RTP (cada fluxo possuindo um *SSRC*). Como o SSRC é associado ao objeto *SessionManager* no momento da inicialização, um participante que queira enviar dois fluxos simultâneos na mesma sessão RTP deve criar e inicializar dois objetos *SessionManager*s distintos, porém, com o mesmo endereço da sessão RTP.

O objeto *SessionManager* mantém um objeto *RTPStream* para cada fluxo de dados RTP que ele percebe na sessão. Existem dois tipos de objetos *RTPStream*: o objeto *ReceiveStream* representa o fluxo que está sendo recebido de um participante remoto; e o objeto *SendStream* representa o fluxo de dados vindo de um objeto *Processor* ou *DataSource* e que está sendo enviado para a rede se o *host* for um emissor.

Vários eventos relacionados à comunicação RTP são utilizados para reportar o estado da sessão RTP e dos seus fluxos. Para receber as notificações destes eventos (*RTPEvents*), a interface *RTPListener* apropriada deve ser implementada e registrada no gerente da sessão (*SessionManager*).

3.2 O Servidor

A aplicação servidora é composta por 4 classes principais.

(i) *VideoServer* – é a primeira classe a ser instanciada. Ele inicia sua execução criando um objeto *ServerSocket* na porta 1554 e aguarda por pedidos de conexão dos clientes. Quando um cliente solicita uma conexão, um novo socket é criado para se comunicar exclusivamente com o cliente. Esta conexão é utilizada como um canal de controle, similar a uma sessão RTSP [Schu98], através do qual o cliente faz solicitações de transmissões ou de adaptações.

(ii) *RTPTransmitter* – trata-se de uma *thread* criada pelo *VideoServer* a cada vez este aceita uma conexão com um novo cliente. O *RTPTransmitter* é responsável por receber e tratar as mensagens do canal de controle.

(iii) *SMWrapper* – é instanciada por um objeto *RTPTransmitter* sempre que há a necessidade de estabelecer uma sessão RTP entre o servidor e um cliente. A classe *SMWrapper* encapsula o *SessionManager* e implementa as interfaces *ReceiveStreamListener* e *SendStreamListener* para receber as sinalizações dos eventos relacionados à sessão RTP.

(iv) *VideoTransmit* – é instanciada por um objeto *RTPTransmitter* para transmitir um fluxo de vídeo em uma sessão RTP. Esta classe cria um objeto *Processor* para receber e processar o arquivo de vídeo antes de enviar os dados para o objeto *SessionManager*, o qual cria um objeto *SendStream* na sessão RTP representando esses dados. Além disso, a classe *VideoTransmit* implementa a interface *ControllerListener* para receber as notificações de transição e gerenciar os estados do objeto *Processor* criado.

3.3 O Cliente

A aplicação cliente também é composta por 4 classes principais.

(i) *RTPPlayer* – é a primeira classe a ser instanciada e entrar em execução. Quando o usuário decide receber um vídeo, a classe *RTPPlayer* cria um *socket* e solicita a conexão do canal de controle à porta 1554 do servidor.

(ii) *SMWrapper* – é instanciada por um objeto *RTPlayer* sempre que o cliente quiser estabelecer uma sessão RTP com o servidor. A classe *SMWrapper* encapsula o *SessionManager* e implementa as interfaces *ReceiveStreamListener* e *SendStreamListener* para receber as sinalizações dos eventos relacionados à sessão RTP.

(iii) *PlayerWindow* – é instanciada por um objeto *SMWrapper* quando este recebe uma sinalização do evento *NewReceiveStreamEvent*, indicando que um novo fluxo foi detectado na sessão RTP. Quando o objeto *SMWrapper* detecta este novo fluxo, ele cria um objeto *Player* para processar e apresentar este novo fluxo. O *PlayerWindow*

corresponde a uma janela onde os componentes visuais *Player* são adicionados e onde a apresentação do vídeo acontece. Além disso, a classe *PayerWindow* implementa a interface *ControllerListener* para receber as notificações dos eventos relacionados ao objeto *Player* em questão.

(iv) *QDM* – é instanciada por um objeto *PlayerWindow* para monitorar os pacotes de feedback RTCP. Um gráfico com os valores de *jitter* e da porcentagem de perda de pacotes é atualizado continuamente. Além disso, os valores dos campos das mensagens *RTCP* são registrados em um *log*.

3.4 Características dos Testes

Os equipamentos utilizados foram dois PCs com processador Pentium-II 250MHz, 128MB de memória, e sistema operacional Windows NT Workstation 4.0 SP6. Estes dois computadores estavam conectados a um barramento Ethernet (100Mbits) compartilhado por mais três outros computadores, sendo um deles o gateway para outra subrede IP. As outras duas estações eram utilizadas por outros usuários executando aplicações web padrões (http, email, ftp, compartilhamento no ambiente windows, etc)(figura 3). Os relógios dos dois PCs (servidor e cliente) foram mantidos sincronizados utilizando-se uma ferramenta de sincronização.

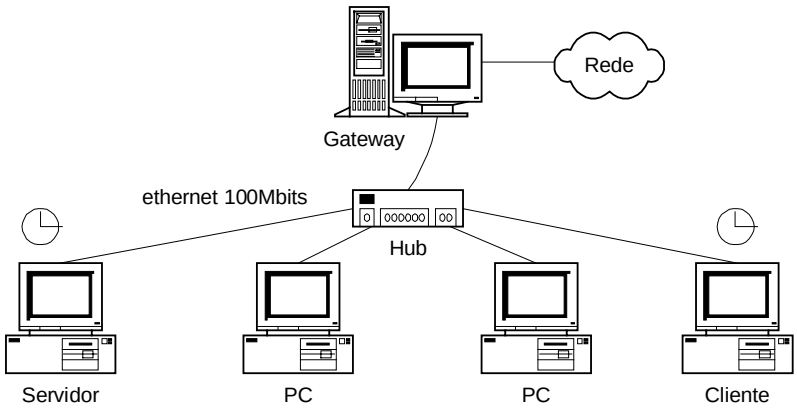
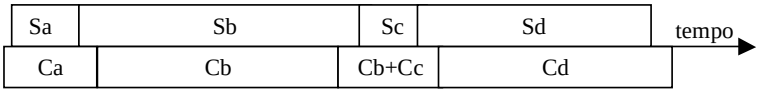


Figura 3 - Ambiente de simulação



Servidor :
Sa – processa pedido de *start* do video inicial
Sb – envia dados do vídeo inicial
Sc – processa pedido de *switch* para o vídeo final
Sd – envia dados do vídeo final

Cliente :
Ca – envia pedido de *start* e aguarda
Cb – recebe e apresenta dados do vídeo inicial
Cc – envia pedido de switch e aguarda
Cd – recebe e apresenta dados do vídeo final

Figura 4 - Etapas de uma rodada de apresentação

Os testes se baseiam na avaliação dos intervalos de tempo entre cada um dos diversos eventos registrados no *log* para uma rodada de apresentação. Como mostrado na figura 4, uma rodada de apresentação compreende todo o processamento realizado desde a solicitação de um vídeo pelo cliente, o atendimento deste pedido pelo servidor, a apresentação deste vídeo pelo cliente, o pedido de chaveamento do vídeo inicial pelo cliente, o chaveamento propriamente dito feito pelo servidor, a apresentação do novo vídeo pelo cliente, até o encerramento da aplicação cliente.

Mensagem	Descrição
START (sp) session_IP (sp) session_PORT (sp) media_LOCATOR (sp) media_TIME	Solicita o início da transmissão do arquivo de vídeo identificado por *media_LOCATOR* na sessão RTP especificada.
SWITCH (sp) session_IP (sp) session_PORT (sp) flow_SSRC (sp) media_LOCATOR (sp) media_TIME	Solicita a substituição do fluxo identificado por *flow_SSRC* pelo arquivo de vídeo identificado por *media_locator* na sessão RTP especificada.
STOP (sp) SSRC (sp) media_TIME	Solicita a interrupção da transmissão do fluxo identificado por *flow_SSRC*.

(sp) = espaço

Tabela 1 - Mensagens de controle da transmissão

Evento registrado	Descrição do evento	Local
Creating Manager… / Manager created.	**Instantes de início e conclusão da criação de um *SessionManager*. A diferença entre estes dois eventos é o tempo gasto pelo sistema para criar uma nova sessão RTP.**	**Servidor e Cliente**
NewSendStreamEvent	Indica o início da transmissão de um novo fluxo na sessão RTP.	Servidor
NewReceiveStreamEvent	Indica o início do recebimento de um novo fluxo na sessão RTP.	Cliente
TimeoutEvent	Indica que um participante deixou de participar da sessão RTP.	Servidor e Cliente
Creating VideoTransmit.../ VideoTransmit created.	Instantes de início e conclusão da criação de um *VideoTransmit*. A diferença entre estes dois eventos é o tempo gasto pelo sistema para criar um novo *VideoTransmit*.	Servidor
Creating Player… / Player create.	Instantes de início e conclusão da criação de um *Player*. A diferença entre estes dois eventos é o tempo gasto pelo sistema para criar um novo *Player*.	Cliente
Realizing Player...	Indica o início do processo de realização do *Player*.	Cliente
RealizeCompleteEvent	Indica a conclusão do processo de realização do *Player* e o início do processo de *prefetch*.	Cliente
PrefetchCompleteEvent	Indica a conclusão do processo de prefetch do *Player* e o instante de chamada do método *start()* do *Player* para que a apresentação do fluxo seja iniciada.	Cliente
StartEvent	Indica que o *Player* iniciou com sucesso a apresentação do fluxo recebido na sessão RTP.	Cliente
KillOldPlayer	Indica o instante em que, durante o chaveamento, é iniciado o processo de destruição do antigo *Player*.	Cliente
SetNewPlayer	**Indica o instante em que, durante o chaveamento, é iniciado o processo de substituição do antigo *Player*.**	**Cliente**
StopEvent	**Indica que a atividade do *Controller* foi paralizada.**	**Servidor e Cliente**
ClosedEvent	Indica que o *Controller* foi fechado (não pode mais ser utilizado)	Servidor e Cliente
START,STOP e SWITCH	Indica o instante de recebimento (servidor) ou envio (cliente) de uma mensagem de controle.	Servidor e Cliente

Tabela 2 - Eventos registrados pelas aplicações

Durante uma rodada de apresentação, as aplicações cliente e servidora utilizam um protocolo de comunicação para controlar o andamento da comunicação. Este protocolo é baseado em mensagens simples de requisição no formato texto (tabela 1) que transportam os comandos do cliente para o servidor através do canal de controle. Este protocolo implementa um subconjunto das funcionalidades existentes no RTSP que são necessárias ao desenvolvimento deste experimento. As versões futuras da API JMF incluirão suporte ao RTSP, que poderá ser adicionado ao sistema sem muita dificuldade.

Os eventos registrados nos *logs* são aqueles relacionados tanto com as sessões RTP (*SessionManagers*) quanto com os objetos *Players* e *Processors* envolvidos. Os eventos relacionados com as sessões RTP são monitorados, no servidor e no cliente, pela classe *SMWrapper* implementada em cada um. Os eventos relacionados aos objetos *Controllers* são monitorados no servidor pela classe *VideoTransmit* e no cliente pela classe *PlayerWindow*, pois são elas que implementam a interface *ControllerListener*. A tabela 2 identifica cada um dos eventos que são registrados nos *logs* e os seus significados.

Durante as transmissões de mídias sob demanda, as sessões entre servidores e clientes devem ser sessões *unicast* para que cada cliente possa ter o controle sobre os fluxos que ele solicitou aos servidores. O chaveamento de dois vídeos neste cenário pode ocorrer na mesma sessão RTP ou em duas sessões RTP distintas. Pode-se notar que essas duas opções estão diretamente relacionadas aos dois níveis de adaptação apresentados ateriormente. No primeiro nível de adaptação, o servidor adapta o fluxo ao longo do eixo de fidelidades do mesmo objeto de mídia, sendo interessante que a transmissão do novo fluxo ocorra na mesma sessão RTP usada pelo fluxo anterior. Já no segundo nível, a adaptação de um fluxo pode gerar a substituição deste por outros fluxos ou a simples adição de outros fluxos simultaneamente. Neste caso, os fluxos simultâneos devem trafegar, obrigatoriamente, em sessões RTP distintas. A simulação do segundo nível trata do caso mais complexo, onde uma sessão RTP é substituída por outra.

Para simular o comportamento do sistema no primeiro nível de adaptação, o cliente envia uma mensagem *SWITCH* para o servidor solicitando que este troque o vídeo corrente por outro, como se o novo vídeo representasse uma variante de fidelidade do primeiro. A sessão RTP especificada na mensagem deve ser a mesma do fluxo que está sendo substituído. O servidor faz, então, a troca dos fluxos na mesma sessão RTP. Para simular o comportamento do sistema no segundo nível de adaptação, o cliente envia uma mensagem *SWITCH* para o servidor solicitando que ele inicie a transmissão de um outro vídeo no lugar do primeiro, como se o novo vídeo fosse uma mídia alternativa ao primeiro. A sessão RTP especificada na mensagem deve ser diferente da sessão em que o primeiro fluxo está sendo transmitido.

Nos dois tipos de simulação descritos, (veja tabela 1) o fluxo que será substituído é identificado por *flow_SSRC* enquanto o novo fluxo é identificado por *media_LOCATOR*. O campo *media_TIME* pode ser usado para fornecer o valor para o qual o relógio do novo fluxo deve ser ajustado antes que a transmissão inicie. Da mesma forma, o primeiro fluxo deve ser interrompido quando o seu relógio atingir o valor de *media_TIME*. Se o valor de *media_TIME* não é fornecido, o servidor utiliza o tempo corrente do relógio e faz o chaveamento imediatamente.

Como um SSRC (identificador do fluxo) é associado ao objeto *SessionManager* no momento da sua inicialização, as duas opções acima devem ser implementadas utilizando, respectivamente, o mesmo objeto *SessionManager* ou objetos *SessionManagers* distintos.

4 RESULTADOS DOS TESTES

Durante as simulações, a avaliação do tempo de resposta do sistema está baseada na percepção do chaveamento dos fluxos por parte do usuário. Ou seja, considera-se como tempo de resposta do sistema o intervalo de tempo entre a paralisação do primeiro vídeo na interface gráfica e o início da apresentação do segundo vídeo (eventos percebidos pelo usuário).

Além dos arquivos de *log*, também foi utilizada uma ferramenta de análise do tráfego da rede [Ana98]. Desta forma, foi possível analisar todos os campos dos protocolos envolvidos, além de acompanhar os instantes de recepção dos pacotes de dados RTP e identificar o tempo de chaveamento dos fluxos. Este procedimento também permitiu identificar quais eventos, registrados nos logs das aplicações, correspondiam à parada no recebimento dos pacotes RTP do primeiro fluxo e ao início da recepção dos pacotes do segundo fluxo.

4.1 Caso 1: Adaptação de Nível 1 (mesma sessão)

No primeiro caso, ilustrado na figura 5, quando o chaveamento é solicitado pelo cliente (mensagem *SWITCH*), o servidor cria um objeto *VideoTransmit* para processar e transmitir o novo vídeo na mesma sessão RTP. Este novo vídeo é enviado para o cliente utilizando o mesmo *SSRC*, já que ele está sendo enviado pelo mesmo objeto *SessionManager*. Para o cliente, tudo se passa como se o fluxo que ele está recebendo fosse o mesmo. O mesmo objeto *Player* que estava apresentando o primeiro vídeo passa a apresentar o novo vídeo. Como não ocorre a criação de um novo objeto *Player* o tempo de chaveamento reduz notadamente.

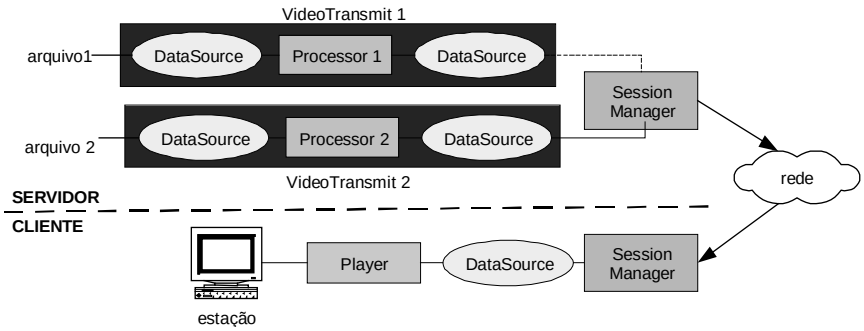


Figura 5 - Chaveamento usando o mesmo objeto *SessionManager*

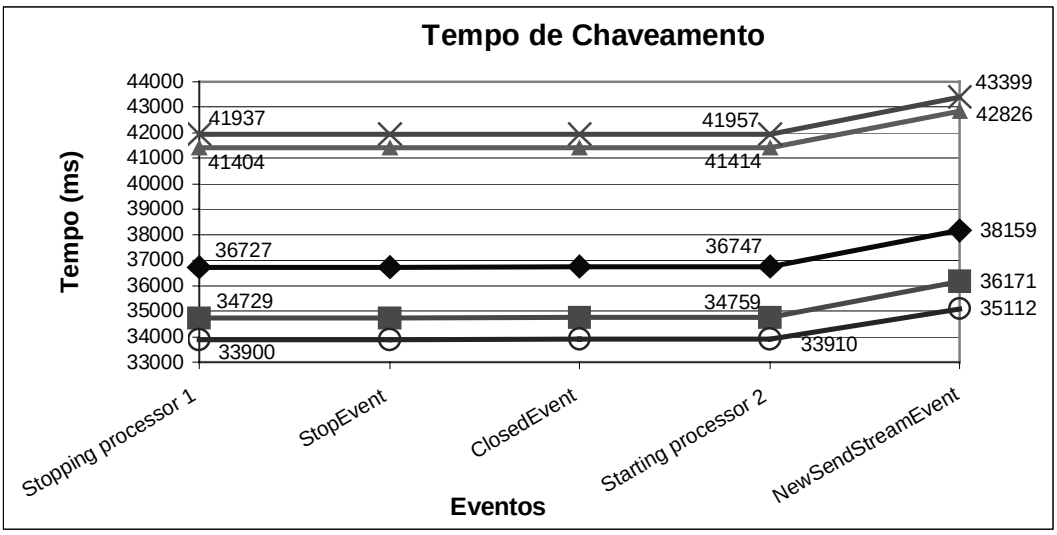


Figura 6 - Tempo de chaveamento (nível 1) para 5 rodadas distintas

O gráfico apresentado na figura 6 mostra as curvas do tempo de chaveamento para 5 rodadas de simulação do primeiro nível de adaptação. Neste gráfico são registrados os tempos de ocorrência dos eventos sinalizados durante o processo de chaveamento. Os valores registrados a partir do evento de recepção da mensagem SWITCH.

4.2 Caso 2: Adaptação de Nível 2 (sessões diferentes)

No segundo caso, ilustrado na figura 7, o servidor cria um segundo objeto *SessionManager* com o endereço da nova sessão RTP fornecido pelo cliente (mensagem *START*). Assim, o servidor começa a participar da nova sessão RTP com um *SSRC* diferente do primeiro. Em seguida, o servidor cria um objeto *VideoTransmit* para processar e transmitir o novo vídeo na nova sessão RTP utilizando o segundo objeto *SessionManager*. Quando os dados enviados chegam ao cliente, este detecta o novo fluxo identificado pelo *SSRC* do segundo objeto *SessionManager* e sinaliza o evento *NewReceiveStream*. Com isso, um novo objeto *Player* é criado para apresentar os dados do segundo vídeo.

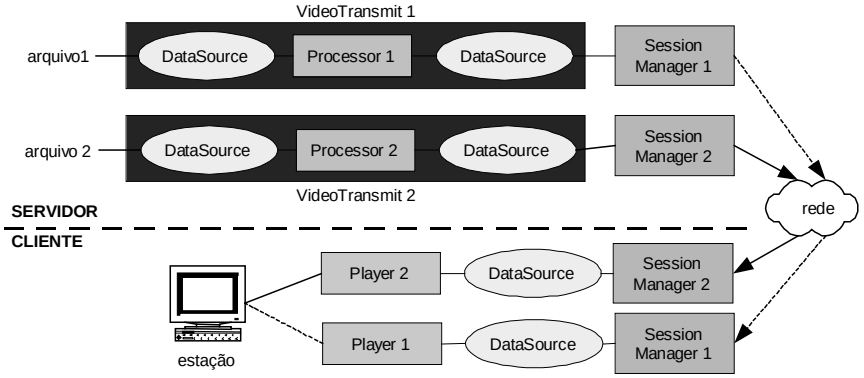


Figura 7 - Chaveamento usando dois objetos *SessionManager*

Esse objeto *Player* passa por todos os estados preparatórios de um objeto *Controller* antes de iniciar a apresentação do vídeo, o que produz um certo atraso. Neste ponto, foi verificada uma limitação da implementação da API JMF2.0: um objeto *Player* só pode ser criado após obter-se a referência do fluxo que ele irá apresentar. Isto é, não é possível criar um *Player* a priori (antes que os dados do fluxo cheguem ao cliente). Assim, mesmo sabendo-se qual fluxo será transmitido pelo servidor (com base na descrição da apresentação), o *Player* só pode ser criado após o novo fluxo atingir o cliente.

Entretanto, uma forma de minimizar o efeito deste atraso é interromper o primeiro fluxo somente após o instante em que o *Player* do segundo fluxo esteja pronto para apresentá-lo. Desta forma, a percepção do tempo gasto na preparação do segundo *Player* por parte do usuário pode ser praticamente nula (o segundo fluxo é estabelecido enquanto o primeiro ainda está ativo). No entanto, uma vez que o chaveamento do primeiro fluxo é gerado por consequência de uma falta de recursos na rede, iniciar o segundo fluxo mantendo o primeiro ativo consumirá ainda mais recursos e prejudicará ainda mais a qualidade de serviço fornecida.

Outro problema que surge é o da sincronização entre os dois fluxos. Uma solução seria o servidor descontar o tempo de preparação do *Player* e iniciar a transmissão do segundo fluxo com um avanço no relógio em relação ao relógio do primeiro fluxo. Este avanço deve corresponder ao tempo de preparação do *Player* para que, quando o segundo fluxo começar a ser apresentado, os relógios dos dois fluxos estejam sincronizados. Por exemplo,

considerando o tempo de preparação igual a TP segundos, se o servidor recebe uma solicitação de chaveamento e o relógio do primeiro fluxo marca T segundos, o relógio do segundo fluxo deve ser ajustado para (T+TP) segundos antes de iniciar a transmissão. Da mesma forma, o primeiro fluxo deve ser interrompido quando o seu relógio atingir (T+TP) segundos. Além desta solução poder sofrer com erros na sincronização (TP não é determinístico), as informações do primeiro fluxo continuam sendo apresentadas com qualidade inferior à aceitável enquanto o novo *Player* não estiver pronto.

Outra solução seria utilizar técnicas de predição para avaliar o instante em que o limite de qualidade especificado seria violado. Assim, antes que ocorra uma violação do limite de qualidade de serviço, uma solicitação de chaveamento é enviada. O tempo de preparação, como no caso anterior, também deve ser descontado pelo servidor para que os fluxos sejam sincronizados. Esta solução é melhor que a anterior pois evita que informações continuem sendo transmitidas após a violação da qualidade de serviço, porém, continua vulnerável às mesmas variações no tempo de preparação. Além disso, a previsão do instante de violação da qualidade pode ser alterada pela própria transmissão do segundo fluxo em paralelo com o primeiro.

Desta forma, com o objetivo de preservar a qualidade de percepção e simplificar o mecanismo, é melhor iniciar o chaveamento pela interrupção do primeiro fluxo e imediatamente começar a transmitir o segundo fluxo na nova sessão. Embora o tempo de chaveamento percebido pelo usuário seja maior, ocorre uma liberação imediata dos recursos da rede, evitando-se que os dois fluxos tornem-se concorrentes. Além disso, os problemas de sincronização são minimizados pois o relógio do segundo fluxo é ajustado, antes da transmiss, para o instante exato em que o primeiro fluxo é interrompido.

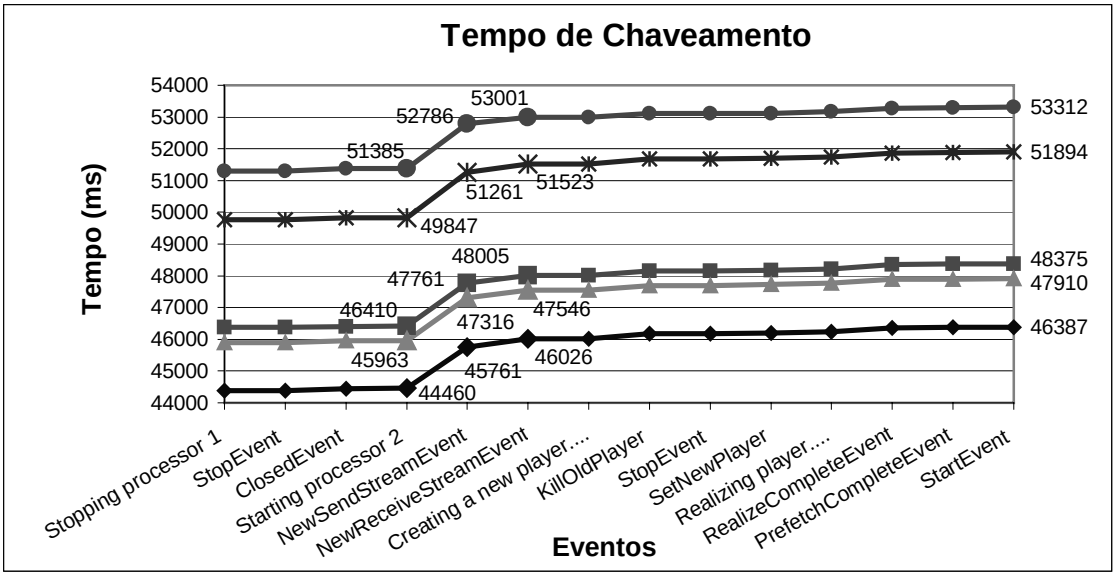


Figura 8 - Tempo de chaveamento (nível 2) para 5 rodadas distintas

Os resultados para este tipo de abordagem mostram que todas as rodadas apresentaram um comportamento bastante semelhante, tanto no primeiro caso (mesma sessão – figura 6) quanto no segundo (sessões diferentes – figura 8).

4.3 Análise dos Resultados

No primeiro caso, o tempo de chaveamento está bem marcado entre os eventos *"Starting Processor 2"* e *"NewSendStreamEvent"*, e corresponde ao tempo gasto pelo servidor para iniciar a transmissão do fluxo. O tempo médio de chaveamento neste caso foi de 1394ms.

No segundo caso, nota-se que, até o quinto evento (*"NewSendStreamEvent"*), as curvas apresentam as mesmas características das curvas do primeiro caso. O tempo gasto até o quinto evento responde por 2/3 do tempo total de chaveamento. A outra parcela (1/3) do tempo total fica distribuída ao longo dos eventos registrados pelo cliente durante a criação e preparação do novo objeto *Player*. Pode-se observar que, nesta parcela, os principais incrementos no tempo de chaveamento ocorrem em dois momentos: entre os eventos *"Creating a new player..."* e *"KillOldPlayer"* (tempo gasto para criar o novo *Player*); e entre os eventos *"Realizing player..."* e *"PrefetchCompleteEvent"* (tempo gasto para o novo *Player* atingir o estado *prefetched*). O tempo médio de chaveamento para o segundo caso foi de 2044ms.

Através destes resultados, pode-se notar que o tempo de resposta obtido se mostra satisfatório quando considerada a sua percepção por parte do usuário. Um usuário habituado a assistir apresentações multimídia em rede geralmente se defronta com paralisações constantes e transmissões com baixa qualidade, o que muitas vezes inviabiliza o objetivo da apresentação. Nos testes realizados neste trabalho, os usuários concordaram que um intervalo de 2000ms entre a paralisação e a retomada da apresentação é bastante aceitável, principalmente quando a continuidade da apresentação é garantida e a semântica inicial do documento é preservada.

5 CONCLUSÕES

Neste artigo foi avaliado o comportamento da estratégia de adaptação para documentos multimídia distribuídos, proposta pelo projeto ServiMídia. Esta estratégia possibilita a utilização de dois mecanismos combinados de adaptação para a apresentação de documentos multimídia em sistemas de rede de computadores: (i) um mecanismo "suave" (nível 1), que efetua uma adaptação a nível de codificação (fidelidade) de mídias independentes de forma transparente ao usuário; e (ii) um mecanismo "forte" (nível 2), que leva o conceito de adaptabilidade ao nível do documento, gerando uma nova estrutura lógica e temporal em tempo real com base na avaliação dos relacionamentos condicionais entre as mídias. Espera-se que a maioria das aplicações realizem adaptações de nível 1, porém, uma vez esgotadas as possibilidades de adaptação no primeiro nível, esta estratégia permite que uma adaptação mais severa seja realizada no documento de maneira consistente, preservando sua coerência.

Com o objetivo de simular o comportamento do sistema em relação ao tempo de resposta às solicitações de adaptação dos fluxos multimídia, foram realizadas implementações simulando os dois níveis de adaptação propostos. A plataforma *Java Media Framework 2.0* foi utilizada nas implementações.

Como mostrado pelos resultados das simulações, o primeiro nível de adaptação possui uma latência menor que o segundo nível, pois o chaveamento dos fluxos pode ser realizado na mesma sessão RTP. Neste caso, o tempo de chaveamento é todo gerado pelo servidor quando este processa a troca dos fluxos na sessão RTP. No segundo nível de adaptação, quando o chaveamento é feito entre sessões RTP diferentes, o cliente também contribui para o tempo de chaveamento, pois, além do tempo gasto pelo servidor, é acrescentado o tempo gasto com a criação e preparação de um novo *Player* para a apresentação do novo fluxo no cliente.

Apesar do tempo de resposta da adaptação "forte" ser implicitamente superior ao tempo de adaptação do mecanismo "suave", o tempo total de chaveamento não representa um fator de desagrado por parte dos usuários. Em um ambiente de ensino à distância este atraso tende a ser aceitável, pois as dimensões temporais das mídias acessadas neste tipo de aplicação são da ordem de alguns minutos. Além disso, quando lidamos com aplicações voltadas ao ensino é melhor a convivência com um pequeno atraso de chaveamento do que a possibilidade de uma interpretação equivocada das informações. A adaptação de segundo nível deve ser ativada nas situações em que o primeiro mecanismo não garantir os requisitos de QoS especificados.

Como trabalhos futuros, no que diz respeito aos mecanismos de adaptação, um estudo pode ser realizado buscando avaliar a possibilidade de se adaptar o documento seguindo os relacionamentos condicionais no sentido inverso, quando a QoS verificada na rede sofrer uma melhoria. Em conjunto, deve ser feito um estudo sobre o momento de solicitação da adaptação, com base no momento da violação da QoS, buscando eliminar possíveis oscilações de adaptação propondo um refinamento na sincronização do reinício do documento adaptado.

6 REFERÊNCIAS

[Ana98] Analyzer: a public domain protocol analyzer, http://netgroup-serv.polito.it/analyzer/

[Aurr98] Aurrecoechea, C., Campbell, A.T., Hauw, L. "A survey of QoS architectures". *ACM Multimedia Systems*, n.6, pp.138-151, 1998.

[Bray98] Bray, T., Paoli, J., Sperberg-McQueen, C. M. "Extensible Markup Language (XML) 1.0". *W3C Recommendation, REC-xml-19980210*, Feb. 1998.

[Court96] Courtiat, J.P., Carmo, L.F.R.C, Oliveira, R.C "A General-purpose Multimedia Synchronization Mechanism Based on Causal Relations", *IEEE Journal on Selected Areas in Communications*, v.14, n.1, pp. 185-195, Jan. 1996.

[Cunh99a] Cunha, E.C.C., Carmo, L.F.R.C., Pirmez, L., "Design of an Integrated Invironment for Adaptive Multimedia Document", *6th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services, LNCS-1718, Springer-Verlag*, pp. 191-204, 1999.

[Cunh99b] Cunha, E.C.C., Carmo, L.F.R.C., Pirmez, L., "A Multimedia Document Authoring Strategy for Adaptive Retrieval in a Distributed System", *Proc. of 6th International Conference on Distributed Multimedia Systems*, University of Aizu-Japan, Jul. 1999.

[Cunh99c] Cunha, E.C.C., Carmo, L.F.R.C., Pirmez, L., "A Stream Relationship Monitor for Adaptive Multimedia Document Retrieval", *Globecom'99*, Rio de Janeiro, Dezembro 1999.

[Ferg98] Ferguson, P., Huston, G. "Quality of Service on the Internet: Fact, Fiction or Compromise?". *Internet Conference*, 1998, Genebra, Suiça.

[Ghin99] Ghinea,G., Thomas, J.P., Fish, R.S. "Quality of Perception to Quality of Service Mapping Using a Dynamically Reconfigurable Communication System", *IEEE Global Telecommunications Conference*, pp. 2061-2065 , Brazil, 1999.

[Gome00] Gomes, R.L., Menezes, H., Carmo, L.F.R.C., Pirmez, L. "Suporte à Apresentação Adaptativa de Aplicações Multimídia em Sistemas Distribuídos", Simpósio Brasileiro de Redes de Computadores, pp. 489-504, Brasil, 2000.

[JMF98] Java Media Framework API, http://java.sun.com/marketing/collateral/jmf_ds.html

[MPEG7a] MPEG 7 Requirements document V.7. ISO/IEC JTC1/SC29/WG11/N2461. Atlantic City, USA, October 98.

[MPEG7b] MPEG 7 Applications document V.7. ISO/IEC JTC1/SC29/WG11/N2462. Atlantic City, USA, October 98.

[Schu96] Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V. "RTP: a transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force, Jan. 1996.

[Schu98] Shulzrinne, H., Rao, A., Lanphier, R. "Real Time Streaming Protocol (RTSP)", RFC 2326, Apr. 1998.

[Smil98] W3C, "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", W3C Recommentation, Jun. 1998.