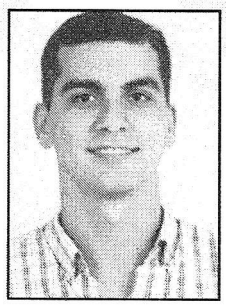
UM PROTOCOLO DE COMUNICAÇÃO HIPERMÍDIA COM QOS.



Antonio Benedito Coimbra Sampaio Júnior.

Mestre em Informática (PUC-RIO), Especialista em Redes de Computadores (UFPA), Engenheiro Civil (UFPA), Tecnólogo em Processamento de Dados (UNAMA) e Professor do Curso de Ciência da Computação (UNAMA).

UM PROTOCOLO DE COMUNICAÇÃO HIPERMÍDIA COM QOS

RESUMO: Nos últimos anos, tem-se observado uma crescente demanda pela transmissão de informações multimídia com garantias de Qualidade de Serviço (QoS – *Quality of Service*) na Internet. Esta demanda tem sido motivada, principalmente, pelo fato de que a rede mundial de computadores, desde a sua concepção, oferece apenas o serviço de melhor-esforço, sem garantias de entrega dos pacotes de dados transmitidos pelas aplicações. Em situações de congestionamento, os referidos pacotes podem ser perdidos ou sofrer atrasos indeterminados, o que não é desejável para as aplicações multimídia. Motivado pelo fato de que o protocolo de comunicação hipermídia utilizado na WWW, o HTTP, não oferece o suporte adequado à transmissão multimídia com garantias de QoS, este trabalho foi desenvolvido tendo como objetivo apresentar um novo protocolo de comunicação hipermídia que oferece tratamento de qualidade de serviço (QoS). Mostra-se como, a partir da definição de um *framework* genérico, pode-se especializar pontos de flexibilização para implementar QoS no protocolo proposto.

PALAVRAS-CHAVE: Internet: protocolos, serviços e aplicações; Qualidade de Serviço; Sistemas multimídia.

1. INTRODUÇÃO

A Internet e, mais especificamente, a WWW têm sido os alvos preferidos para a pesquisa e desenvolvimento de aplicações hipermídia. Apesar da introdução de serviços com garantias de QoS na interconexão de redes, dentre os quais incluem-se os modelos *IntServ* e *DiffServ* [1,2], ainda é necessário outros esforços para uma verdadeira provisão de QoS *fim-a-fim*.

A garantia do serviço fim-a-fim significa que *todos* os subsistemas participantes, incluindo as próprias aplicações, são responsáveis pela provisão da QoS. Nesse sentido, as tecnologias de nível de aplicação que implementam a WWW não oferecem suporte adequado às aplicações hipermídia. Dessa forma, torna-se necessário, entre outras coisas, a utilização de protocolos de comunicação hipermídia que ofereçam serviços com garantias de QoS, permitindo: a negociação de parâmetros de qualidade, a alocação dos recursos necessários para a transmissão de dados multimídia com a qualidade solicitada e a utilização de protocolos de transporte adequados a cada tipo de transmissão.

Este trabalho propõe um protocolo de comunicação hipermídia com as características acima. As funções de provisão de QoS incluídas nesse protocolo são definidas com base nos Frameworks para Provisão de QoS em Ambientes

Genéricos de Processamento e Comunicação, desenvolvidos no Laboratório Telemídia da PUC-Rio [3]. Esses frameworks foram concebidos de forma a apenas documentar esboços das várias funções de provisão de QoS que são encontradas em qualquer parte de um sistema de comunicação. Para poderem ser aplicados em uma parte específica do sistema, alguns dos seus pontos de flexibilização devem ser preenchidos com informações (algoritmos, parâmetros, etc) relativas ao contexto desejado. Este trabalho mostra como esses pontos de flexibilização são preenchidos no contexto do protocolo de comunicação hipermídia proposto.

O restante do artigo encontra-se estruturado da seguinte forma. A Seção 2 apresenta alguns trabalhos relacionados. A Seção 3 apresenta o protocolo de comunicação hipermídia proposto, com ênfase nas funções de provisão de QoS. A Seção 4 apresenta um cenário de uso do protocolo (uma solicitação de um hiper-documento com garantias de QoS), enquanto a Seção 5 é reservada às conclusões.

2. TRABALHOS RELACIONADOS

Existe uma grande variedade de tecnologias relacionadas à transferência de documentos multimídia. Analisamos neste artigo somente alguns dos mais importantes e mais conhecidos: os protocolos desenvolvidos pela IETF (HTTP, RTSP), WAPForum (WSP) e ITU (H.323). Em [4] pode ser encontrado um estudo mais aprofundado de outros trabalhos.

O HTTP [5] é o protocolo padrão para a transferência de hiper-documentos na WWW. Apesar de estar em constante evolução, o HTTP apresenta várias características que o tornam inadequado às aplicações hipermídia. Pode-se citar como a principal dessas características a ausência de distinção entre mídias contínuas e mídias discretas. Em particular, o seu uso disseminado sobre um mesmo serviço de transporte confiável, oferecido pelo protocolo TCP, constitui-se em um problema para mídias contínuas, pois os mecanismos de controle de erro por retransmissões e de controle de fluxo são em geral inadequados para transmissões cuja característica principal é a manutenção da relação temporal da informação nos destinatários. Em resposta às limitações supracitadas, o IETF desenvolveu o RTSP [6]. Este protocolo pode utilizar os protocolos de transporte TCP, UDP ou RTP, dependendo do tipo de informação a ser transportada. Apesar disso, o RTSP não oferece quaisquer garantias de QoS.

O WSP [7] é o protocolo de comunicação responsável por organizar as trocas de informações entre aplicações cliente e servidor WAP. Ele define o conceito de sessão de trabalho e pode utilizar dois tipos de serviço: orientado à conexão, utilizando um serviço de transporte confiável (WTP), e não-orientado à conexão, utilizando um serviço de transporte não-confiável (WDP). Como principal limitação do WSP podemos destacar a ausência de primitivas de QoS, não sendo possível portanto requisitar serviços com os parâmetros de QoS desejados.

O H.323 [8] é um padrão para a transmissão de informações multimídia em tempo real em redes comutadas por pacotes em geral. Foi desenvolvido pelo grupo de estudos SG16 do ITU-T. Atualmente encontra-se na sua quarta versão (H.323 v4). Vários estudos têm sido desenvolvidos para permitir que terminais H.323 estabeleçam uma conexão especificando os parâmetros de QoS desejados. Na recomendação H.323 v4, anexo C *(H.323 on ATM)*, está detalhado o uso do protocolo de transporte AAL5 para permitir o

estabelecimento de circuitos virtuais com QoS em redes ATM. No apêndice II desta mesma recomendação (*Transport Level Resource Reservation Procedures*), está descrito como o padrão H.323 pode incorporar o protocolo RSVP [9] no seu modelo de referência, para permitir o transporte de dados multimídia na Internet com garantias de qualidade. Utilizando o RSVP, terminais H.323 podem estabelecer canais de comunicação com a QoS especificada. A principal limitação observada é que para estabelecer conexões com garantias de qualidade, os terminais H.323 devem ter conhecimento dos parâmetros de QoS especificados pelo RSVP.

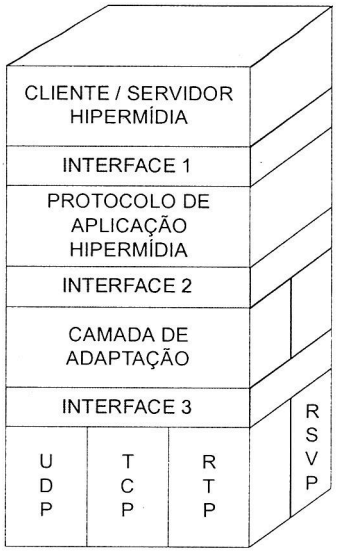


Figura 1 – Arquitetura do protocolo de comunicação hipermídia

3. PROTOCOLO DE COMUNICAÇÃO HIPERMÍDIA

O protocolo de comunicação hipermídia proposto no presente trabalho posiciona-se entre as aplicações hipermídia e os protocolos de nível de transporte, conforme ilustrado na Figura 1. A camada de adaptação permite que diferentes protocolos de nível de transporte sejam utilizados para a transmissão dos dados multimídia, oferecendo ao protocolo de comunicação hipermídia uma interface uniforme. Entre outras funções, essa camada se responsabiliza por solicitar os serviços de transporte que mais se adeqüem ao tipo das mídias a serem transportadas. A especificação detalhada de todas as interfaces ilustradas na Figura 1 pode ser encontrada em [4].

O protocolo utiliza três estruturas básicas para propiciar o oferecimento de serviços às aplicações: *pipes*, *MediaPipes* e sessões de aplicação. Um *pipe* representa todos os componentes (roteadores, comutadores, *proxies*, etc.) que fazem parte do "caminho" fim-a-fim entre os clientes e servidores que compõem uma aplicação hipermídia. *Pipes* que atendam a requisitos específicos de QoS definidos pela aplicação hipermídia são denominados *MediaPipes*. Um cliente pode solicitar ao protocolo de comunicação hipermídia várias partes de um mesmo hiper-documento em um mesmo servidor, cada uma dessas partes com requisitos de QoS diferentes, tornando necessária a criação de diversos *MediaPipes*. Para facilitar o gerenciamento de diversos *MediaPipes* criados entre os mesmos clientes e servidores, utilizou-se o conceito de sessão de aplicação. Por questões de segurança, uma aplicação do usuário só pode ter acesso ao(s) *MediaPipe*(s) criado(s) na sua sessão de aplicação. O protocolo desenvolvido oferece a possibilidade de criação de diversas sessões de aplicação agrupando vários *MediaPipes*. A Figura 2 ilustra a existência de uma sessão de aplicação entre um cliente e um servidor, sessão esta contendo um *MediaPipe* para áudio e outro para texto.

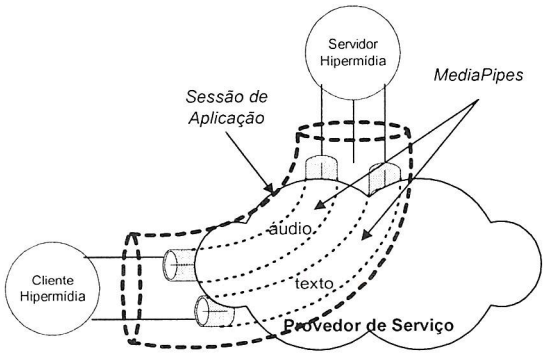


Figura 2 – Exemplo dos conceitos de MediaPipe e de sessão de aplicação

■ Tratamento de QoS

Para oferecer tratamento de QoS no protocolo de comunicação hipermídia, foram instanciados os Frameworks para Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação apresentados em [3]. A notação UML (*Unified Modeling Language*) [11] é utilizada na representação dos diagramas de classes que descrevem esta arquitetura. A Figura 3 apresenta a instância do Framework para Parametrização de Serviços.

O nível de serviço desejado é representado pela super-classe *ServiceCategory* e os parâmetros de QoS são especificados pela super-classe *Parameters*. Estas classes estão relacionadas através do atributo *parameterList* que define o conjunto de parâmetros de QoS associados a uma determinada categoria de serviço. As categorias de serviço efetivamente oferecidas pelo protocolo se baseiam nas classes de serviço definidas pelo modelo *Intserv*, estando definidas em duas sub-classes[1]: *Guaranteed* e *Controlled-load*. Como os *MediaPipes* oferecidos pelo protocolo são específicos para um determinado tipo de mídia, foram criadas sub-classes de *Guaranteed* e *Controlled-load* específicas para cada tipo. É importante enfatizar, entretanto, que a partir das super-classes (os pontos de flexibilização) desse framework é possível criar novas categorias de serviço (com novos parâmetros), o que torna o protocolo adaptável à introdução de novos serviços.

Todas as sub-classes de *ServiceCategory* possuem métodos que definem quais parâmetros de QoS devem ser utilizados. Como exemplo, a sub-classe *AudioGuaranteed*, utilizada quando a aplicação do usuário solicita a transferência de áudio com garantias estritas de qualidade possui cinco parâmetros de QoS: *max_delay, sample_size, sample_rate, loudness* e *audioCodification*.

É importante ressaltar que o protocolo irá manipular parâmetros de QoS das aplicações e da camada de transporte (via camada de adaptação), sendo necessário o mapeamento dos requisitos de QoS expressos por esses parâmetros entre essas camadas. Dessa forma, existem sub-classes de *Parameters* que representam parâmetros de QoS dessas duas camadas. Os parâmetros de QoS da camada de transporte estão agrupados no retângulo pontilhado na Figura 3.

A Figura 4 apresenta as instâncias dos Frameworks para Alocação de Recursos, Negociação de QoS e Sintonização de QoS. O Framework para Alocação de Recursos, cuja classe principal é *InfrastructureQoSNegotiator*, tem como responsabilidade gerenciar a alocação de recursos. Esta classe implementa os métodos definidos na interface *PrimitivesResourceReservation* para realizar a reserva dos recursos necessários para o estabelecimento de

[1] A categoria de serviço *best-effort* é o nível de serviço padrão oferecido pelo protocolo, não sendo necessária a sua modelagem.

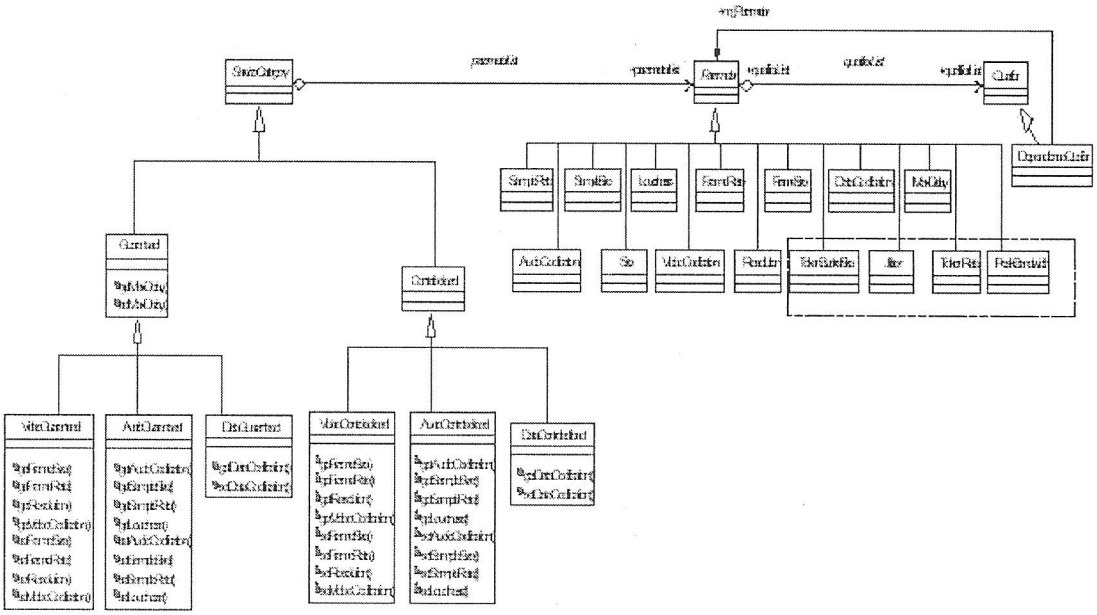


Figura 3 – Instância do Framework para Parametrização de Serviços

*MediaPipe*s. Esta classe possui o atributo *HypInf* que define o negociador de QoS utilizado pelo protocolo de comunicação hipermídia (representado pela classe *ProtocolQoSNegotiato*r). O Framework para Negociação de QoS modela os mecanismos de negociação e mapeamento, assim como os de controle de admissão.

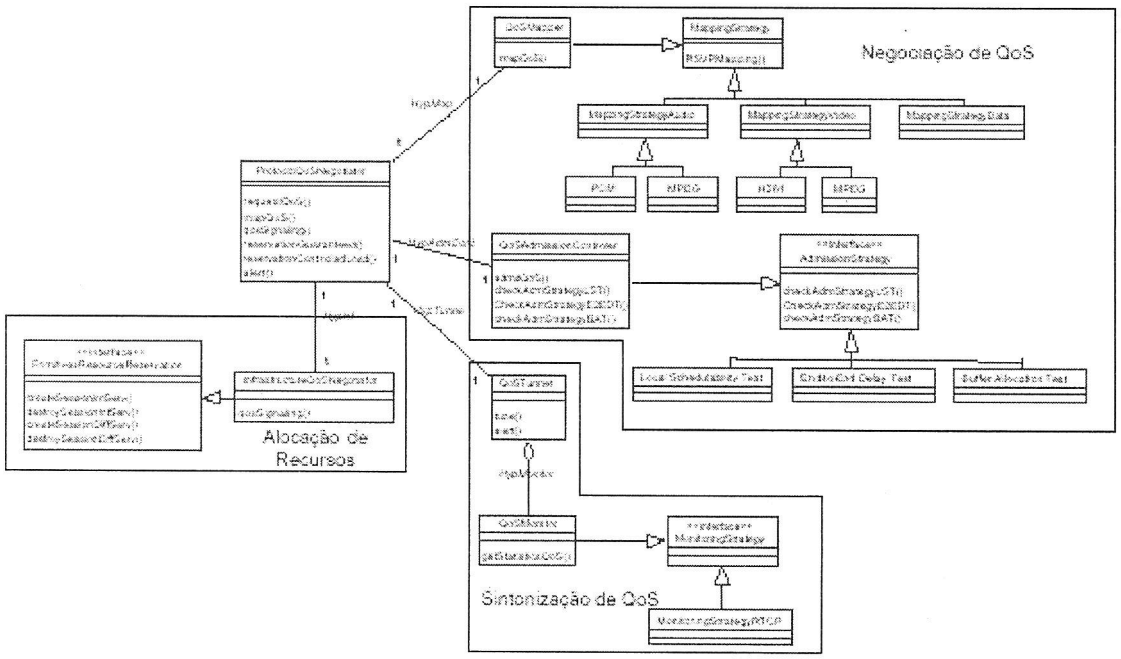


Figura 4 – Instância dos Frameworks para Alocação de Recursos, Negociação de QoS e Sintonização de QoS

O Framework para Sintonização de QoS modela os mecanismos de monitoração e sintonização. As classes *QoSAdmission Controller* e *QoSMapper* são responsáveis, respectivamente, por realizar o controle de admissão de novos *MediaPipes* e por mapear os parâmetros de qualidade das aplicações para parâmetros da camada de transporte. A classe *QoSMonitor* é utilizada para monitorar a QoS estabelecida por um *MediaPipe*, verificando se está de acordo com o nível de serviço solicitado pelo usuário. Caso não esteja, mecanismos de sintonização[2] podem ser acionados (classe *QoSTuner*). Todas estas classes estão relacionadas com o negociador de QoS utilizado pelo protocolo hipermídia (*Protocol QoSNegotiator*).

A classe *ProtocolQoSNegotiator* pode ser analisada como uma extensão do protocolo de comunicação hipermídia, sendo responsável por negociar o oferecimento de serviços com a qualidade especificada pelas aplicações. A classe *QoSMapper* utiliza as regras de mapeamento definidas na interface *MappingStrategy*. As sub-classes definidas em *MappingStrategy* são utilizadas para definir regras de mapeamento específicas para cada tipo de mídia que pode ser utilizada. Neste trabalho, foram definidas duas regras de mapeamento: uma para áudio PCM e a outra para vídeo H.261. Maiores detalhes sobre a regra de mapeamento PCM será mostrada na Seção 4.

A classe *QoSAdmissionController* utiliza as regras de controle de admissão definidas na interface *AdmissionStrategy*. As sub-classes definidas são específicas para cada tipo de controle de admissão. A classe *QoSMonitor* utiliza as regras de monitoração definidas na interface *MonitoringStrategy*. Como exemplo, a sub-classe *Monitoring Strategy TCP* é responsável por monitorar os pacotes de sinalização RTCP[3].

4. Cenário de Uso do Protocolo de Comunicação Hipermídia

Com o objetivo de validar a especificação do protocolo, foi desenvolvido no Laboratório TeleMídia um protótipo[4] em JAVA que implementa, além do protocolo, um cliente e um servidor hipermídia. Para testar o funcionamento do protótipo, foram usadas duas estações Pentium II/333MHz interligadas por uma rede *Ethernet 100 Base*-T. A Figura 5 ilustra as estações utilizadas na rede e a base de dados (BD) gerenciada pelo servidor hipermídia. A BD possui um hiper-documento contendo áudio (A), vídeo (V), texto (T) e imagem (I).

[2] Mecanismos de sintonização são responsáveis pela manutenção da orquestração dos recursos com a QoS negociada, sem que haja necessidade de interrupção, isto é, renegociação do fornecimento do serviço.
[3] É importante ressaltar que o RTCP faz parte do RTP, por isso não foi especificado no Modelo de Referência do protocolo de comunicação hipermídia.
[4] Maiores detalhes sobre o desenvolvimento do protótipo e o código fonte das classes e interfaces desenvolvidas podem ser encontrados em [14].

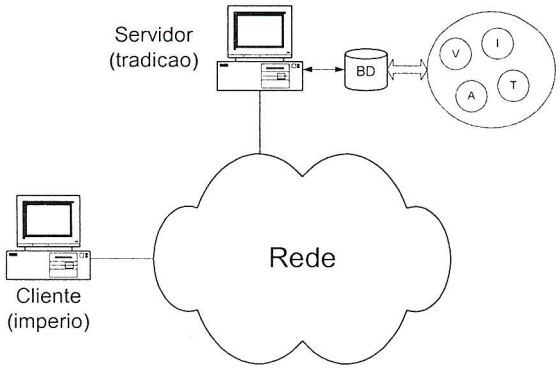


Figura5 – Estações Cliente e Servidor utilizadas no experimento

O cliente e o servidor desenvolvidos no protótipo executam na estação império (porta 1054) e tradicao (porta 1234), respectivamente. Dois arquivos de configuração, transport.ini e mediaPipeTransport.ini, são utilizados para descrever quais os protocolos de transporte disponíveis que mais se adeqüam ao tipo de informação a ser transportada.. Os protocolos de transporte UDP, TCP e RTP, e de reserva de recursos RSVP, podem ser usados nestas duas estações.

Após a iniciação do cliente e servidor, serviços de comunicação podem ser requisitados ao protocolo. Para solicitar qualquer serviço com QoS, as seguintes etapas devem ser realizadas: (a) a aplicação (usuário) define a categoria de serviço *(best-effort,guaranteed* ou *controlled-loa*d) desejada; (b) os parâmetros de QoS devem ser negociados; (c) os parâmetros de qualidade devem ser traduzidos para cada uma das diferentes camadas do sistema de comunicação; e (d) os recursos necessários devem ser reservados no caminho "fim-a-fim" entre o transmissor e o receptor.

Exemplo: o cliente (império) deseja receber um áudio em tempo-real com um desempenho aceitável do servidor (tradicao). Desta forma, as etapas descritas nos itens (a), (b), (c) e (d) devem ser realizadas no protocolo de comunicação hipermídia. Neste exemplo, considera-se que já existe uma Sessão de Aplicação criada entre o cliente (imperio) e o servidor (tradicao).

As subseções a seguir descrevem os passos necessários para a solicitação de um documento áudio em tempo-real com garantias de qualidade.

4.1. Solicitação do serviço

Passos Realizados no Cliente:

Passo1: Solicitando o Transporte de um Documento

Utilizando a primitiva *HYP_ GetQoS.request (session_key, entity_id)*, o cliente especifica qual a Sessão de Aplicação a ser utilizada para a criação do *MediaPipe (session_key)* e qual o documento a ser recebido *(entity_*id).

Continuando o Exemplo, a aplicação cliente na estação império solicita a transferência de um documento, que possui identificador 10, na Sessão de Aplicação (abc@tradicao1054) criada com o servidor tradicao.

Passo2: Enviando Mensagem XML ao Servidor

O protocolo hipermídia cliente, ao receber a primitiva *HYP_GetQoS.request (session_key, entity_id)*, codifica os parâmetros da primitiva, gerando uma mensagem *GetQoS*. No protocolo especificado, optou-se por codificar todas as mensagens definidas em XML *(Extensible Markup Language)*.

Seguindo o exemplo anterior, a primitiva HYP_GetQoS.request (abc@ tradicao1054, 10) será utilizada para gerar

a mensagem GetQoS.XML (ver Figura 6). Utilizando qualquer protocolo de transporte, a mensagem GetQoS.xml é enviada ao servidor.

Passos Realizados no Servidor:
Passo3: Recebendo mensagem XML do Cliente

Fazendo uso de qualquer protocolo de transporte, o protocolo hipermídia servidor recebe a mensagem XML, decodifica-a e gera uma primitiva de serviço.

Seguindo o Exemplo, o protocolo de transporte recebe a mensagem GetQoS.xml,decodifica-a e gera a primitiva Hyp_GetQoS.indication (abc@tradicao1054, 10).

Passo4: Enviando Primitiva de Serviço ao Servidor

O servidor recebe a primitiva de serviço HYP_GetQoS.indication (session_key,entity_id), verifica se o documento solicitado (entity_id) existe e quais os parâmetros de descrição de tráfego associados ao mesmo. Se o documento solicitado existir, a primitiva HYP_GetQoS.response (session_key, entity_id, trafficSpec[5], resourceSpec[6]) é retornada ao protocolo hipermídia servidor; caso contrário, a primitiva HYP_Error.response(message) é retornada.

Seguindo o Exemplo, o servidor recebe a primitiva HYP_GetQoS. indication (abc@ tradicao1054, 10), verifica que o documento solicitado (identificador 10) existe, identifica o seu conteúdo (áudio – 'sound.wav') e analisa os parâmetros de descrição de tráfego associados a este documento (sampleRate – 8Khz, sampleSize – 8bits, Loudness – Stereo e audioCodification - PCM). Para a sua transmissão com estes parâmetros de tráfego, o servidor exige um retardo de 4 ms. A primitiva HYP_GetQoS.response (abc@tradicao1054, sound.wav, (8Khz, 8bits, Stereo, PCM), 4 ms) é então retornada.

Passo5: Recebendo Primitiva de Serviço do Servidor

O protocolo hipermídia servidor recebe a primitiva HYP_GetQoS.response (session_key,entity_id, trafficSpec, resourceSpec), calcula o retardo que irá oferecer para processar os pacotes de dados a serem transmitidos pelo servidor e inicia o processo de anúncio de tráfego.

Continuando o Exemplo, o protocolo hipermídia recebe a primitiva HYP_GetQoS.response (abc@tradicao1054, sound.wav, (8Khz, 8bits, Stereo, PCM), 4 ms), verifica que exigirá um retardo de 1ms, atualiza o valor do parâmetro resourceSpec, de 4ms para 5 ms (retardo_servidor + retardo_protocolo_hipermídia_servidor), e inicia o processo de anúncio de tráfego. Para que isto seja feito, o protocolo hipermídia servidor deverá realizar o **mapeamento dos parâmetros de descrição de tráfego,** para permitir que todos os subsistemas envolvidos na comunicação fim-a-fim compreendam os parâmetros especificados.

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Chamando uma DTD externa -->
<!DOCTYPE GetQoS SYSTEM "HypProtocol.dtd">
<GetQoS>
    <session_key>abc@tradicao1054</session_key>
     <entity_id>10</entity_id>
    <TrafficSpec></TrafficSpec>
    <ResourceSpec></ResourceSpec>
</GetQoS>
```

Figura 6 – Mensagem GetQoS.XML gerada pelo protocolo hipermídia

[5] Identifica os parâmetros de descrição de tráfego associados a um documento.
[6] Identifica os parâmetros de desempenho de cada entidade (servidor, protocolo hipermídia servidor, protocolo hipermídia cliente e cliente) e de cada provedor de serviços.

Passo6: Realizando Mapeamento dos Parâmetros de Descrição de Tráfego

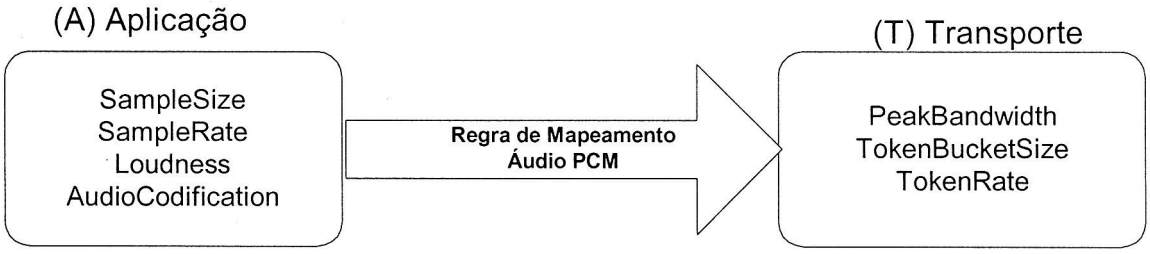
Os parâmetros de descrição de tráfego especificados no nível de aplicação deverão ser mapeados em parâmetros do nível de transporte, para ser possível iniciar o processo de anúncio de tráfego e verificar qual a quantidade de recursos que serão alocados pela infra-estrutura de comunicação para transportar o documento solicitado com garantias de qualidade.

Seguindo o Exemplo, os parâmetros de tráfego do nível de aplicação, especificados na primitiva HYP_GetQoS. response (abc@tradicao1054, sound.wav, (8Khz, 8bits, Stereo,PCM), 5 ms), deverão ser mapeados em parâmetros do nível de transporte, para ser possível iniciar o processo de anúncio de tráfego. Neste trabalho, foi definida uma regra de mapeamento específica para áudio PCM, conforme ilustrada na Figura 7. Os parâmetros nível de transporte obtidos foram: peakBandwidth = 128 Kb/s, tokenBucketSize = 16 bits/pixel e tokenRate = 128 Kb/s. Tendo sido realizado o mapeamento dos parâmetros com sucesso, utilizam-se as primitivas definidas na API IPQoS[7] para solicitar à infra-estrutura de comunicação um serviço de transmissão compatível com os parâmetros de tráfego especificados.

Passo7: Iniciando o Anúncio do Tráfego

Uma vez realizado o mapeamento dos parâmetros de qualidade, do nível de aplicação para os de nível de transporte, o protocolo hipermídia servidor descobre, junto à camada de adaptação, qual protocolo de reserva de recursos deve ser utilizado para iniciar o processo de anúncio de tráfego.

Continuando o Exemplo, o protocolo hipermídia servidor, através de sua camada de adaptação, utiliza o protocolo RSVP para iniciar o processo de anúncio do tráfego para a transmissão do documento ('sound.wav') com os seguintes parâmetros: 128 Kb/s, 16 bits/pixel e 128 Kb/s. Como mencionado, o protótipo desenvolvido usa a API IPQoS, que



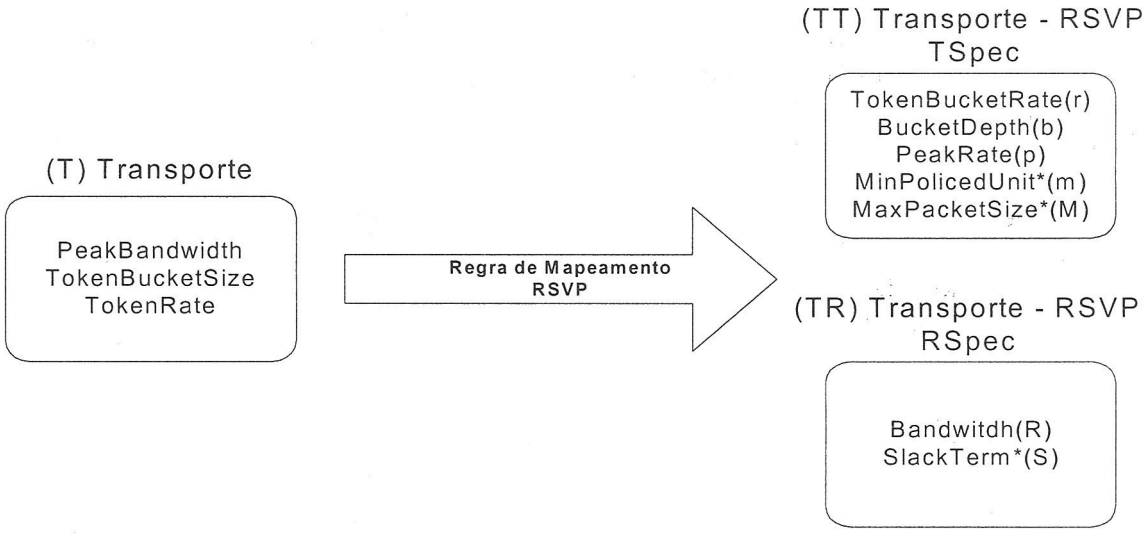
1. T(PeakBandwidth) = A(SampleRate) x A(SampleSize) x A(Loudness)
   1.1. Loudness = 1 (Mono) ou 2 (Stereo)
2. T(TokenRate) = T(PeakBandwitdh), pois a codificação PCM é CBR
3. T(TokenBucketSize) = A(Samplesize) x Loudness

Figura 7 – Regra de Mapeamento para Áudio codificação PCM

[7] IPQoS é a implementação em Java de uma interface de programação para suporte à QoS em uma infra-estrutura Internet. A versão utilizada, 1.02, inclui somente negociadores de QoS do tipo "intserv-RSVP", isto é, utiliza somente o protocolo de reserva de recursos RSVP.

implementa o RSVP. Assim sendo, para solicitar serviços a esta interface, o protótipo deve mapear os parâmetros de transporte em parâmetros específicos RSVP, utilizando para isto a regra definida na Figura 8.

Os parâmetros RSVP obtidos foram: TokenBucketRate = 16000 B/s, BucketDepth = 2 B, Bandwidth = 16000 B/s, MinPolicedSize[8] = 128 B e MaxPacketSize = 1500 B. Utilizando-se a primitiva IPQoS.sender(RSVPsession, end_server, TSpec) da API IPQoS, inicia-se o processo de anúncio do tráfego que será gerado pelo servidor hipermídia para transmitir o documento solicitado com garantias de qualidade e que deverá ser suportado pela infra-estrutura de comunicação. Este anúncio de tráfego (16000 B/s, 2 B, 16000 B/s, 128 B, 1500 B) é realizado através do envio de mensagens PATH para o cliente **(imperio)**. Após ter realizado com sucesso as etapas de mapeamento de parâmetros e anúncio de tráfego, o protocolo hipermídia servidor irá codificar os parâmetros da primitiva HYP_GetQoS. response (session_key, entity_id, trafficSpec, resourceSpec) em uma mensagem GetQoS.xml[9], e enviá-la ao cliente **(imperio)**. Caso alguma das etapas não tenham sido realizadas com sucesso, a primitiva HYP_Error(message) é retornada, sendo gerado a mensagem Error.xml.



1. TT(TokenBucketRate) = T(TokenRate)*(1000)/8
2. TT(BucketDepth) = T(TokenBucketSize)/8
3. TT(PeakRate) = T(PeakBandwidth)*(1000)/8
4. TR(Bandwitdh) = T(TokenRate)*(1000)/8

* - deduzidos através de informações fornecidas pelos provedores de comunicação

Figura 8 – Regra de Mapeamento dos Parâmetros de Transporte para Parâmetros RSVP

[8] Os parâmetros *MinPolicedSize* e *MaxPacketSize* são definidos pelos provedores de infra-estrutura. Neste exemplo, considerou-se o valor normalmente utilizado em redes Internet.

[9] Esta mensagem possui os parâmetros de descrição de tráfego (trafficSpec) e de desempenho dos fornecedores (resourceSpec – servidor e protocolo hipermídia servidor) que serão utilizados para transmitir o documento solicitado.

Passos Realizados no Cliente:
Passo8: Recebendo mensagem XML do Servidor

Fazendo uso de qualquer protocolo de transporte e de reserva de recursos, o cliente recebe a mensagem XML e a mensagem de sinalização RSVP, decodifica-as e gera duas primitivas de serviço.

Seguindo o Exemplo, o protocolo de transporte utilizado pelo cliente recebe as mensagens GetQoS.xml e PATH, decodifica-as e gera as primitivas dataTransp. confirm(getQoS.xml), indicando o recebimento da mensagem GetQoS.xml, e IPQoS.dispatch (PathMessageArrived), indicando o recebimento da mensagem PATH. A primeira mensagem, GetQoS.xml, identifica qual o documento solicitado (áudio 'sound.wav') e quais os parâmetros de descrição de tráfego (8Khz, 8bits, Stereo, PCM) e de desempenho dos fornecedores (retardo de 5 ms) necessários para transmitir este documento com garantias de QoS. A segunda mensagem, PATH, identifica qual o tráfego que será gerado na infra-estrutura de comunicação para transmitir este documento.

Opcionalmente, pode possuir um objeto Adspec que identifica quais são as características[10] da infra-estrutura de comunicação no caminho fim-a-fim. A Figura 9 ilustra o recebimento da mensagem PATH pelo cliente (imperio).

A mensagem PATH recebida possui o objeto Adspec que contém o parâmetro 'minimum_path_latency', responsável por determinar o retardo fim-a-fim na infra-estrutura de comunicação utilizada. Neste exemplo, o valor obtido foi de 30 ms.

Após ter recebido estas duas mensagens, o protocolo hipermídia cliente realiza o processo de decodificação da mensagem XML, calcula o retardo total oferecido, retardo_fornecedor_serviço (5ms) + retardo_infra_estrutura (30 ms) + retardo_protocolo_hipermídia_cliente (2 ms), e envia a primitiva HYP_GetQoS. confirm(abc@tradicao1054, sound.wav, (8Khz, 8bits, Stereo, PCM), 37 ms), ao cliente hipermídia.

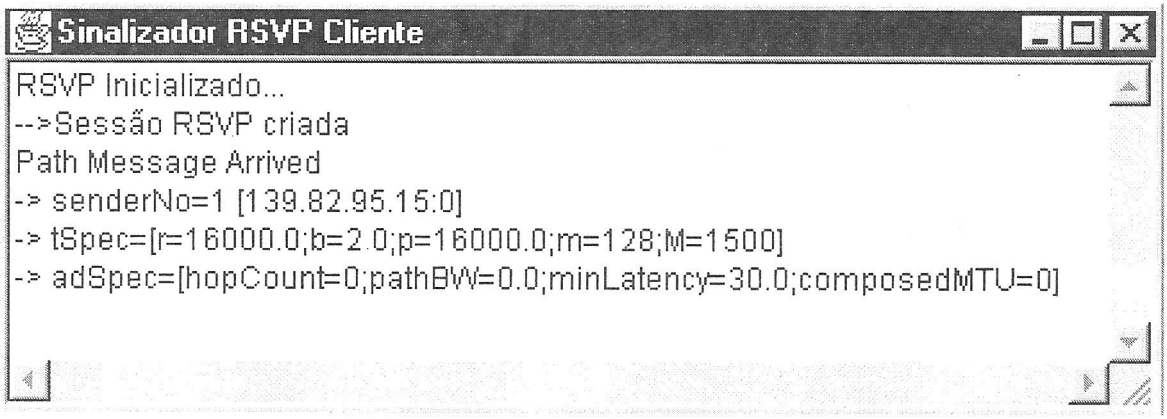


Figura 9 – Sessão RSVP Cliente recebendo mensagem PATH

[10] Retardo fim-a-fim, total de banda passante, número de roteadores, etc.

Passo9: Enviando Primitiva de Serviço ao Cliente

O cliente recebe a primitiva de serviço HYP_GetQoS.confirm (session_key, entity_id,trafficSpec, resourceSpec), verifica que o documento solicitado (entity_id) existe e analisa quais são os parâmetros de descrição de tráfego (trafficSpec) e de alocação de recursos (resourceSpec) associados à transmissão do documento com garantias de qualidade. Com base nestas informações, o cliente inicia o processo de criação do MediaPipe para transportar às informações solicitadas.

Continuando o Exemplo, a aplicação cliente recebe a primitiva HYP_GetQoS.confirm (abc@ tradicao 1054, sound.wav, (8Khz, 8bits, Stereo, PCM), 37 ms), verifica que o documento solicitado existe ('sound.wav') e analisa quais são os parâmetros de descrição de tráfego (8Khz, 8bits, Stereo, PCM) e de alocação de recursos (37 ms) associados à transmissão do mesmo.

4.2. Criação do MediaPipe
Passos Realizados no Cliente:
Passo1: Realizando o Controle de Admissão no Cliente

Dando continuidade ao Exemplo, a aplicação hipermídia cliente é responsável por iniciar o processo de **controle de admissã**o, para verificar se os parâmetros de descrição de tráfego e de alocação de recursos especificados (8Khz, 8bits, Stereo, PCM), 37 ms) podem ser oferecidos por todos os elementos (cliente, protocolo hipermídia cliente, protocolo hipermídia servidor e servidor) e pelo provedor de serviços envolvidos na comunicação com QoS fim-a-fim.

A aplicação hipermídia realiza os testes de controle de admissão, verificando que é possível receber o áudio solicitado com os parâmetros de descrição de tráfego e de alocação de recursos especificados. Calcula o seu retardo (3 ms) para receber e apresentar o áudio solicitado em tempo-real ao usuário final, reserva recursos correspondentes a este valor e solicita aos outros elementos a reserva de todos os recursos necessários para a criação do MediaPipe com a categoria de serviço ('AudioGuaranteed') e os parâmetros de QoS (maxDelay – 60 ms[11] requisitado – 3 msreservado = 57 ms, sampleRate – 8Khz, sampleSize – 8 bits, Loudness – Stereo e audioCodification – PCM) na Sessão de Aplicação (abc@tradicao1054) criada com o servidor (tradicao. telemidia. puc-rio.br). A primitiva HYP_Reserve.request (abc@tradicao1054, sound.wav, AudioGuaranteed, 57 ms, 8Khz, 8bits, Stereo, PCM) é enviada ao protocolo hipermídia cliente.

Passo2: Realizando Controle de Admissão no Protocolo

Dando continuidade ao Exemplo, ao receber a primitiva HYP_Reserve.request (abc@tradicao1054, sound.wav, Audio Guaranteed, 57 ms, 8Khz,8bits, Stereo, PCM), o protocolo hipermídia cliente verifica se já existe algum MediaPipe áudio criado com a QoS especificada. Se houver, verifica se o MediaPipe está sendo utilizado. Não havendo uso do MediaPipe, este será utilizado para o transporte do áudio solicitado em tempo-real com garantias de qualidade. Caso contrário, inicia-se o processo de criação de um novo MediaPipe.

[11] Note que o parâmetro maxDelay deve ser maior ou igual ao retardo fim-a-fim anteriormente calculado, uma vez que o sistema pode, na melhor hipótese, garantir esse último valor. Este valor, entretanto, pode ser menor do que o necessário para aplicação executar a contento.

$$(1)\ MaxDelay = \frac{(b-M)(p-R)}{R(p-r)} + \frac{(M+Ctot)}{R} + Dtot, caso(p > R \geq r)$$

$$(2)\ MaxDelay = \frac{(M+Ctot)}{R} + Dtot, caso(p > R \geq r)\text{, onde:}$$

No Exemplo acima apresentado, não existe nenhum MediaPipe áudio criado. Assim sendo, o protocolo de comunicação hipermídia cliente realiza o controle de admissão,verificando a necessidade de um retardo de (2 ms) para processar os pacotes de áudio em tempo-real. Reserva então esse valor e continua o processo de criação do MediaPipe, com a categoria de serviço ('AudioGuaranteed') e com os parâmetros de QoS (maxDelay – 55ms (60 msrequisitado – 5 msreservado), sampleRate – 8Khz, sampleSize – 8bits, Loudness – Stereo e audioCodification - PCM), entre o cliente **(imperi**o) e o servidor **(tradica**o). Para que isto seja feito, o protocolo hipermídia cliente deverá realizar o mapeamento de parâmetros de QoS, para permitir que todos os elementos envolvidos na comunicação fim-a-fim compreendam os parâmetros de qualidade solicitados, e, posteriormente, iniciar o processo de **reserva de recurso**s.

Passo3: Realizando o Mapeamento dos Parâmetros de Qualidade

Seguindo o Exemplo, o protocolo de comunicação hipermídia cliente irá mapear os parâmetros de aplicação (AudioGuaranteed, 55 ms, 8 KHz, 8 bits, Stereo) nos parâmetros de transporte (Guaranteed, 128 Kb/s, 55 ms, 16 bits/ pixel, 128 Kb/s). Posteriormente, os parâmetros de transporte são mapeados em parâmetros específicos do RSVP (Guaranteed, 55 ms, 16000 B/s, 2 B, 16000 B/s, 128 B, 1500 B).

Passo4: Realizando a Reserva de Recursos

Dando seqüência ao Exemplo, para solicitar a transferência do áudio em tempo-real com a categoria de serviço 'Guaranteed', o protocolo cliente hipermíd ia deverá calcular os parâmetros R e S do objeto RSpec, utilizando o algoritmo CalculaR() que se baseia em duas equações matemáticas, a seguir descritas.

onde: p (peak rate of flow), b (bucket depth), r (token bucket rate), m (minimum policed unit), M (maximum datagram size) e são parâmetros Tspec; e R (bandwidth), S (slack term) são parâmetros RSpec. Maiores detalhes podem ser encontrados em [9].

(a) MaxDelay(req) = MaxDelay (user) – MaxDelay (infra_estrutura) = 60 ms – 30 ms = 30ms;

(b) Utilizando a equação (2), temos: R = p = 16000 B/s;

(c) $MaxDelay = \frac{(M+Ctot)}{R} + Dtot \rightarrow MaxDelay = \frac{(1500+0)}{16000} + 0 \rightarrow 93ms$

(d) MaxDelay (93 ms) > MaxDelay(req)(30 ms) ® Utilizar a equação (2)

(e) $MaxDelay = \frac{(M+Ctot)}{R} + Dtot \rightarrow R = \frac{(M+Ctot)}{MaxDelay} + Dtot \rightarrow R = \frac{(1500+0)}{93} = 16129B/s$

```
1. Inicio Algoritmo CalculaR
        2. Calcular MaxDelay(req) = MaxDelay(user) - minimun_path_latency
        3. Utilizando a equação (2), fazer com que R = p e calcular MaxDelay.
        4. Se (MaxDelay > MaxDelay(req))
                5. usar equação (2)
        6. Senão
                7. usar equação (1)
8. Fim
```

Figura 10 – Algoritmo para calcular o parâmetro R do protocolo RSVP

Por aproximação, o valor de R utilizado foi 16000 B/s. Segundo [9], o valor de S deve ser iniciado com zero. Utilizando-se as primitivas definidas na API IPQoS, o protocolo hipermídia cliente envia a mensagem RESV com os parâmetros de qualidade acima especificados.

É importante ressaltar que o retardo total solicitado pelo cliente hipermídia para a transmissão do áudio foi de 60 ms. Deste valor, (3 ms) foram reservados pelo próprio cliente, (2 ms) reservados pelo protocolo hipermídia cliente e (30 ms) solicitados à infra-estrutura de comunicação. A diferença restante, 25 ms, deverá ser reservada pelos fornecedores de serviço (protocolo hipermídia servidor e o servidor).

O protocolo hipermídia cliente irá codificar os parâmetros da primitiva HYP_Reserve.request (session_key, entity_id, serviceQoS) em uma mensagem Reserve.xml.

Passos Realizados no Servidor:
Passo5: Recebimento da mensagem RESV e da mensagem XML enviadas pelo Cliente

Seguindo o Exemplo, o servidor (tradicao) recebe as mensagens Reserve.xml e RESV, decodifica-as, gerando as primitivas HYP_ Reserve.indication (abc@tradicao1054, sound.wav, (AudioGuaranteed, 25 ms (60 msrequisitado – 35 msreservado), 8Khz, 8bits, Stereo, PCM) e IPQoS.dispatch (ResvMessageArrived).

O servidor tendo recebido a mensagem RESV, garante-se que o retardo de 30 ms solicitado foi reservado pela infra-estrutura de comunicação para a transmissão do áudio em tempo-real com a categoria de serviço e os parâmetros de QoS especificados.

Passo6: Realizando Controle de Admissão no Protocolo

Dando continuidade ao Exemplo, ao receber a primitiva HYP_Reserve.indication(abc@tradicao1054, sound.wav, (AudioGuaranteed, 25 ms, 8Khz, 8bits, Stereo, PCM), o protocolo hipermídia servidor realiza com sucesso os testes de controle de admissão, levando um retardo de (1 ms) para processar e transmitir o áudio em tempo-real, reserva então os recursos, e envia ao servidor a primitiva HYP_Reserve.indication(abc@tradicao1054, sound.wav, (AudioGuaranteed, 24 ms (60 msrequisitado – 36 msreservado), 8Khz, 8bits, Stereo, PCM).

Passo7: Realizando Controle de Admissão no Servidor

Continuando o Exemplo, a aplicação servidor (tradicao) recebe a primitiva HYP_Reserve.indication (abc@tradicao1054, sound.wav, (AudioGuaranteed, 24 ms, 8Khz, 8bits, Stereo, PCM), realiza os testes de controle de admissão com sucesso, reserva o retardo de 4 ms para a transmissão do áudio solicitado e inicia a transmissão deste documento com os parâmetros de QoS especificados.

Passo8: Iniciando a Transmissão do Áudio Solicitado em tempo-real

O servidor utiliza o protocolo de transporte RTP para enviar os pacotes de áudio no MediaPipe criado. A escolha do RTP deve-se ao fato deste ser o protocolo de transporte padrão da Internet para o envio e recebimento de dados de mídia contínua, como áudio e vídeo.

Passos Realizados no Cliente:
Passo9: Recebimento da Stream de Áudio em tempo-real

O cliente hipermídia cria um player RTP para receber os pacotes de áudio enviados pelo servidor e apresentar a informação ao usuário final. A Figura 11 ilustra o player RTP [15] criado pelo cliente para receber o áudio (sound.wav).

6. CONCLUSÕES

O presente trabalho apresentou um protocolo de comunicação hipermídia que oferece serviços com garantias de qualidade. Para garantir os requisitos de qualidade solicitados, todos os elementos participantes da comunicação fim-a-fim deverão prover QoS. Desta forma, além de utilizar o protocolo de comunicação hipermídia com QoS especificado neste trabalho, seria necessário fazer uso de: 1- aplicações do usuário (um browser, por exemplo) que possuíssem funcionalidades de QoS, permitindo neste caso, a solicitação de ajustes nos parâmetros de qualidade recebidos durante a comunicação; 2- sistemas operacionais que oferecessem QoS, para que fosse possível a alocação, de forma eficiente, de todos os recursos necessários na estação do usuário (memória, tempo de processamento e recursos de comunicação) para a comunicação com QoS; e 3- provedores de comunicação que implementassem algum dos modelos de serviços com QoS propostos para a Internet (serviços integrados e diferenciados).

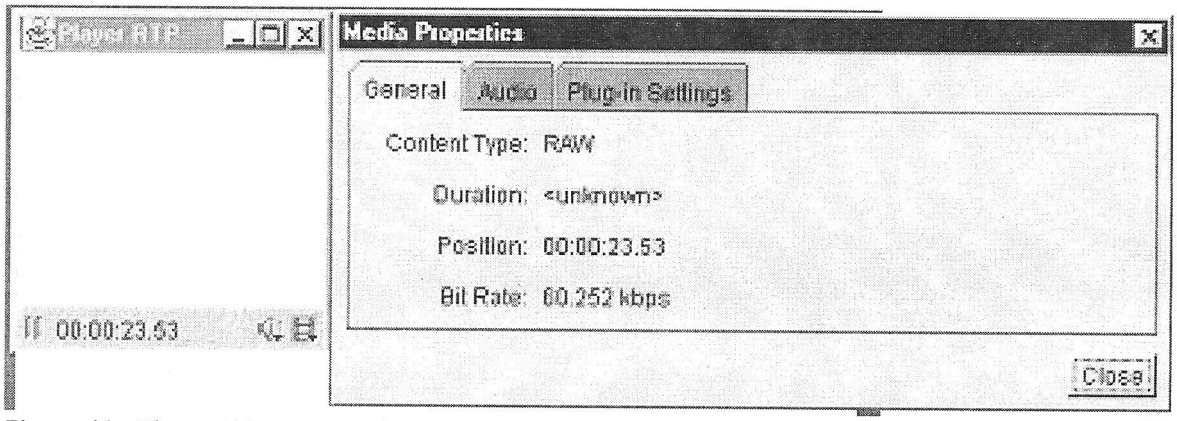


Figura 11 – Player RTP para receber a transmissão do áudio (sound.wav) em tempo-real

REFERÊNCIAS BIBLIOGRÁFICAS

BRADEN, R. et al. Integrated Services in the Internet Architecture: an Overview, **RFC**, n. 1633, jun. 1994.

BLAKE, S. et al. An architecture for differentiated services, **RFC**, n. 2475, dez. 1998.

MOTA, O. T. J. D. D. L., et al. Uma arquitetura adaptavel para provisão de QOS na internet. In: SIMPOSIO BRASILEIRO DE REDES DE COMPUTADORES, 19. , 2001, Florianópolis. **Anais ...** Florianópolis, 2001

SAMPAIO JR., A B. C. **Especificação de um protocolo de comunicação hipermídia com qualidade de serviço e modularidade funcional**. 2001. Dissertação (Mestrado)- PUC, Rio de Janeiro, 2001.

BERNERS-LEE, T., et al. Hypertext transfer protocol- HTTP/1.1. **RFC**, n. 2616, jun. 1999.

SCHULZRINNE, H.; et al. Real time streaming protocol (RTSP). **RFC**, n. 2326, abr. 1998.

WAP WSP. **WAPForum Recommendation**. Disponivel em: < http:// www.wapforum.org/docs/ copyrigtht.htm>.

JONES, P. **H.323v4**, fev. 2001. Disponivel em: <http:// www.packetizer.com/iptel/ h323/papers/>.

WHITE, P. P. RSVP and integrated services in the internet: a tutorial. **IEEE** Communications Magazine, maio 1997.

MOTA, O. T. J. D. D. L .et al.**Serviços com qualidade na internet**: relatório técnico do laboratório telemidia. Rio de Janeiro: PUC, 1999.

BOOCH, G.; et al. **The unified modeling language user guide**. [S.l.]: Addison-Wesley, 1999.

COLCHER, S. **Um meta modelo para aplicações e serviços de comunicação adaptáveis com qualidade de serviço**. 1999. Tese (Doutorado)- PUC-Rio, Rio de Janeiro, 1999.

GOMES, A T., A . **Um framework para provisão de QoS em ambientes genéricos de processamento e comunicação**. 1999. Dissertação (Mestrado)- PUC- Rio, 1999.

SAMPAIO JR., A B., C. **Especificação de um protocolo de aplicação hipermídia com qualidade de serviço**: projeto final de programação. Rio de Janeiro: PUC, 2000.

DE CARMO, L. **Core Java media framework**. Prince- Hall, 1999.