Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia Elétrica

Implementação de um Mecanismo de Adaptação da Qualidade de Serviço para uma Aplicação de Vídeo-Conferência

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de **Mestre em Engenharia Elétrica**

Hallthmann Lima dos Reis

Florianópolis, novembro de 2000

à meus pais e irmãos pelo apoio total.

Agradecimentos

Agradeço primeiramente à Deus pela oportunidade de vencer mais esta fase na minha vida.

Também agradeço à meus pais e irmãos que estando distantes tanto contribuiram com apoio para realização deste sonho.

Ao professor Jean-Marie pela orientação no desenvolvimento deste trabalho.

Ao amigo Cristian Koliver de fundamental importância para o desenvolvimento e conclusão deste trabalho.

Agradeço à CAPES pelo apoio financeiro.

Finalmente, gostaria também de agradecer a todos os amigos do LCMI, que contribuiram com suas amizades e momentos de descontração, para que este trabalho se torna-se menos penoso.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

Implementação de um Mecanismo de Adaptação da Qualidade de Serviço para uma Aplicação de Vídeo-Conferência

Hallthmann Lima dos Reis

novembro/2000

Orientador: Prof. Dr. Jean-Marie Farines
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: multimídia, QoS, adaptação, grau de qualidade.
Número de Páginas: 72

Este trabalho descreve a filosofia e implementação de mecanismos de adaptação da qualidade de serviço para ambientes melhor-esforço sobre a aplicação de vídeo-conferência VIC. Os mecanismos aqui implementados atuam sobre o Sistema Multimídia Distribuído como um todo, considerando duas situações: a sobrecarga de processador da máquina local que gera o fluxo de dados, e a sobrecarga de rede no qual os dados gerados trafegam. A atuação do mecanismo local (sobre o processador) é norteada por uma função de grau de qualidade, que considera a qualidade como um todo, abrangendo todos os parâmetros da aplicação percebidos pelo usuário (freqüência de quadros e fidelidade da imagem). A atuação do mecanismo global (sobre a rede) usa duas filosofias diferentes para adaptação: uma usando a função grau de qualidade e "sockets", e a outra usando os pacotes de Relatório de Receptor do Protocolo de Controle Tempo-Real (RTCP). Para esses diversas situações são apresentados resultados como também são feitas análises de desempenho.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Implementation of a Mecanims of Quality of Service for a Video Conference Application

Hallthmann Lima dos Reis

November/2000

Advisor: Prof. Dr. Jean-Marie Farines
Area of Concentration: Control, Automation and Industrial Informatic
Keywords: multimedia, QoS, adaptation, quality degree function.
Number of Pages: 72

This work describes the philosophy and implementation of a quality of service adaptation mechanism for best-effort environments using the VIC video-conferencing tool. The mechanism implemented acts over distributed multimedia systems in two scenarios: a local one, when there is an overcharge of the processor generating the video stream, and a global one, when the network where the stream is flowing becomes congested. The local mechanism works based on a quality degree function, where quality encompasses all application parameters preceived by end users, such as frame rate and image fidelity. The global mechanism, on the other hand, uses two methods to perform the adaptation: one using a quality degree function and a socket, and the other using report packets of Real-Time Control Protocol (RTCP). Performance tests were carried out in varied situations.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A melhoria do desempenho das redes de comunicação, associada à queda de preços dos dispositivos multimídia e ao desenvolvimento de novas tecnologias para programação distribuída, tem motivado o crescimento do número de aplicações multimídia distribuídas, este crescimento por sua vez tem sido acompanhado de um aumento do grau de exigência dos usuários em relação à qualidade dessas aplicações, e que vem se expressando através de vários atributos referenciados como *parâmetros de qualidade de serviço* (ou QoS). A noção de QoS surgiu originalmente em comunicações, para descrever certas características técnicas da transmissão de dados, e segundo Vogel et al. [33] o termo QoS pode ser definido como:

> *Qualidade de Serviço* representa um conjunto das características quantitativas e qualitativas de um sistema multimídia distribuído, necessário para se **garantir** as funcionalidades requeridas pela aplicação.

Os parâmetros de QoS se apresentam na forma de atributos das diversas entidades que compõem o sistema multimídia distribuído (SMD), como rede, sistema operacional, aplicativos, dispositivos de "hardware", etc., e podem ser instanciados com diferentes valores que influenciarão na qualidade final percebida pelos usuários dessas aplicações.

Os diversos parâmetros de QoS podem ser organizados segundo uma visão arquitetural hierárquica do SMD (figura 1.1), onde cada camada oferece serviços com QoS para as camadas superiores. Neste trabalho adota-se uma arquitetura para SMD's semelhante àquelas propostas em [15] e [24], onde, no topo da arquitetura situa-se o

usuário. A seguir, vem a *camada de aplicação*, que possui parâmetros de QoS relacionados diretamente à qualidade da apresentação, representados por aspectos quantitativos como freqüência de amostras de áudio, resolução cromática, freqüência e resolução de quadros, relação entre os componentes YUV (que representa a profundidade dos pixels e pode ser medido em três campos: cor, brilho e contraste), tamanho dos pacotes de amostra de áudio, atraso e variação do atraso fim-a-fim. Entre a camada de aplicação e as camadas mais baixas da arquitetura está a *camada de sistema*, composta pelos "middlewares" com destaque para o sistema operacional. Os principais parâmetros de QoS desta camada são atributos de tarefas: tempos de processamento, períodos, "deadlines", prioridades, etc. Na base da hierarquia estão as *camadas de comunicação* e *dispositivos*. A primeira é composta por todas as entidades físicas e/ou abstratas relacionadas ao processo de comunicação (conexão, roteadores, protocolos, unidades de transporte, etc.) e os parâmetros de QoS desta camada correspondem aos atributos dos níveis de transporte, rede, enlace e físico (do modelo OSI [2, 14], por exemplo) como taxa de transmissão necessária, taxa de erro de bits, atraso de rede e prioridade dos sistemas finais. A camada de dispositivos contém os parâmetros de QoS correspondente aos atributos desses dispositivos: resolução dos monitores, capacidade de memória RAM, capacidade de disco e memória "cache", velocidade do processador, latência de disco, etc.

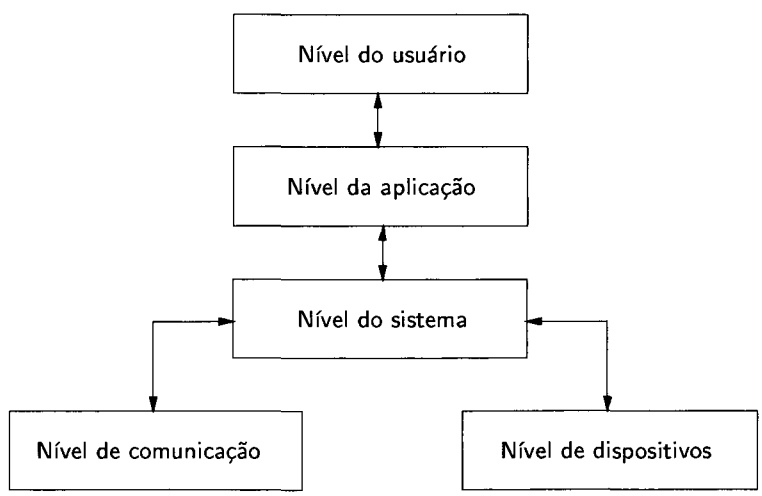

Figura 1.1: Arquitetura do SMD.

Muitos dos parâmetros que definem a qualidade de serviço (QoS) acima mencionados, podem sofrer variações bruscas e descontroladas em seus valores durante a transferência de dados na rede, ou durante a execução das tarefas relacionadas ao processamento destes dados nos sistemas finais, em virtude da variação de carga da rede e dos processadores, especialmente. O usuário percebe essa variação na forma de lapso de som, distorção da imagem, falta de sincronismo entre imagem e som e redução da freqüência de exibição. Essa degradação brusca e descontrolada na QoS face as mudanças de contexto do SMD, diminui bastante o grau de satisfação do usuário.

Uma forma de realização de adaptação de QoS, consiste na moldagem das necessidades da aplicação em termos de recursos (particularmente, largura de banda da rede e de ciclos de processador) ao contexto corrente do SMD, através da modificação dos valores de um ou mais de seus parâmetros de QoS visando diminuir a taxa de transmissão e/ou tempo de processamento da tarefa de descompressão. Uma limitação desse tipo de abordagem é que, é necessário um conhecimento prévio da arquitetura de rede sobre a qual a aplicação trabalha.

Este trabalho consiste na implementação de um mecanismo de adaptação de QoS para aplicações multimídias distribuídas orientado ao usuário final, realizado na aplicação de vídeo-conferência VIC (Video Conference Tool) [23], visando de uma maneira mais geral testar mecanismos de adaptação em uma aplicação real de vídeo-conferência. Em condições de flutuações de carga da rede e dos processadores, o mecanismo atua sobre os parâmetros de QoS na interface do VIC de forma a aumentar ou degradar a QoS suavemente, dentro de limites pré-fixados de acordo com as expectativas do usuário. Todo o processo de adaptação é conduzido pela definição de uma função *grau de qualidade* proposto em [20], que associa a combinações dos valores dos parâmetros de QoS do VIC uma medida arbitrária de desempenho, permitindo maximizar a qualidade oferecida *como um todo*, ou seja, como o usuário final percebe a qualidade, e não através de parâmetros de QoS individuais.

Neste trabalho no capítulo 2, é definido um modelo genérico de QoS para aplicações em SMD's com e sem garantias de qualidade de serviço, na qual o mecanismo aqui implementado se baseia.

No capítulo 3, é feita uma descrição das características do ambiente de teste com a aplicação de vídeo-conferência VIC, de sua arquitetura assim como da arquitetura de

rede, dando uma descrição objetiva de alguns objetos que implementam as funcionalidades de sua estrutura.

No capítulo 4, é apresentada a implementação do mecanismo de adaptação de QoS de escopo local, que atua em condições de sobrecarga de processadores, sendo ainda apresentados alguns resultados e discussões.

No capítulo 5, é apresentada a implementação do mecanismo de adaptação de escopo global, que atua em condições de sobrecarga da rede. Aqui duas variantes do mecanismo são implementadas: uma usando os "sockets"; e outra parte usando os pacotes de relatório de receptor do protocolo RTCP (Protocolo de Controle Tempo-Real). Em seguida são apresentados os resultados e discussões.

No capítulo 6, é apresentada uma síntese do trabalho, algumas conclusões obtidas e perspectivas para trabalhos futuros.

No apêndice A é dada uma visão básica da estrutura do código fonte do VIC.

Capítulo 2

Modelo Genérico de QoS

2.1 Introdução

O crescimento explosivo da Internet tem levado cada vez mais o compartilhamento de recursos entre usuários num SMD próximos ao seus limites. Devido à dinamicidade dos SMD's, nos quais as exigências dos usuários em relação a QoS e a disponibilidade de recursos variam no espaço e no tempo, surge a necessidade de modelos compostos de seqüências de etapas e de atividades para tratamento da QoS. Neste capítulo, é descrito o modelo genérico de QoS, que norteou a construção do mecanismo de QoS proposto neste trabalho. Tal modelo é baseado em outros pré-existentes encontrados na literatura e foi proposto em [18].

2.2 Um Modelo Genérico de QoS

A adaptação de QoS em uma aplicação multimídia distribuída, e de forma mais geral, a gerência de QoS, pressupõe um conjunto de etapas e atividades, algumas anteriores à própria sessão, e que são organizadas na forma de um *modelo genérico de QoS*, de tal forma que ele formalize a QoS na rede e nos sistemas finais.

O modelo genérico de QoS aqui proposto é baseado naquele definido por Campbell et al.[2]. Tal modelo é composto de três fases (figura 2.1): configuração da sessão, sessão e término da sessão de usuário.

Na literatura existem vários outros modelos de arquiteturas de QoS propostas:

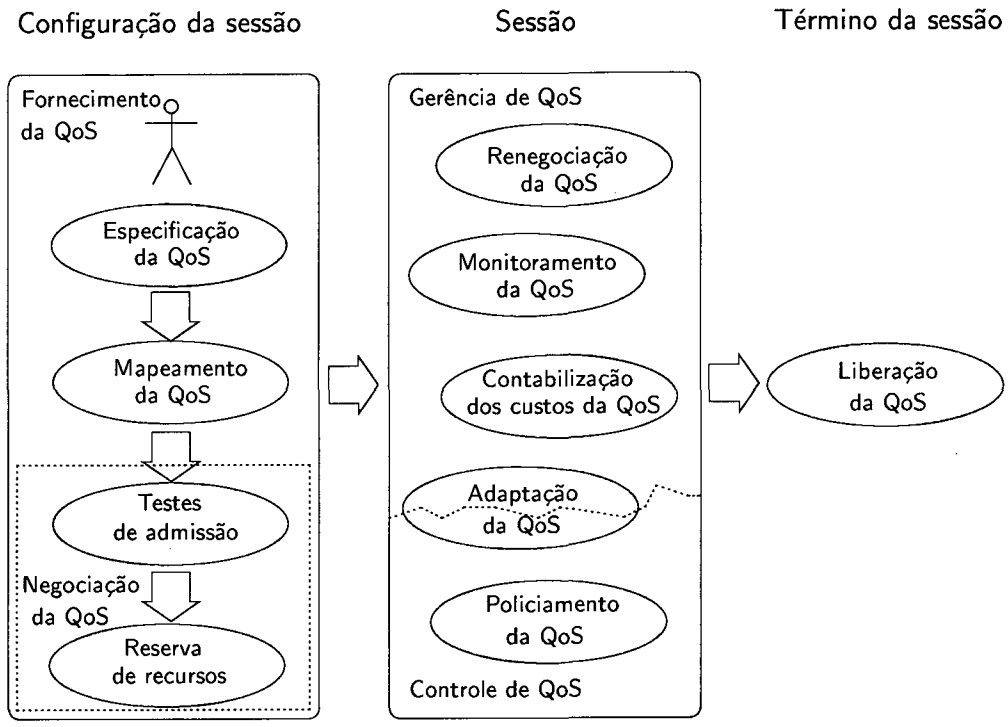

Figura 2.1: Modelo genérico de QoS.

"Tenet Architeture" [2, 14], "OSI QoS Framework" [2, 14], "Quality of Service Architeture - QoS-A" [2, 14], "OMEGA" [2, 14], "Heidelberg QoS Model" [2, 14], entre outros.

2.2.1 Configuração da Sessão

Na *configuração da sessão* ocorre o ajuste de QoS, que abrange as seguintes etapas [14]:

Especificação de Requisitos de QoS

A etapa de especificação de QoS é a responsável pela definição do nível de QoS requerido pela aplicação e pela política de gerenciamento. A especificação tem natureza declarativa, ou seja, o usuário especifica o que é necessário, ao invés de definir como isto deve ser feito pelos mecanismos de QoS.

Geralmente, o usuário especifica os valores desejáveis a partir de uma lista de parâmetros de QoS oferecidas pela interface da aplicação. As interfaces também pos-

suem diferentes formas para que o usuário escolha determinados valores, como através de limites máximo e mínimo, valores-alvo, lista discreta de valores aceitáveis, etc. A forma como os valores são expressados também difere de aplicação para aplicação: em algumas, os valores são expressos quantitativamente; em outras, os valores são expressados de forma qualitativa, através do uso de termos como "EXCELENTE", "BOM", "MEDIO" e "BAIXO".

Mapeamento de QoS

O papel da atividade de mapeamento de QoS em sistemas com reserva de recursos, é da tradução dos valores de QoS entre os diferentes níveis do sistema. O mapeamento dos requisitos do usuário nos parâmetros de QoS relevantes para o SMD, se faz necessário, de forma que a qualidade possa se adequar às necessidades dos diversos usuários.

Os valores dos parâmetros significativos das camadas inferiores do SMD, são obtidos a partir dos valores dos parâmetros especificados pelo usuário, através de funções de mapeamento.

Negociação da QoS

A função da etapa de negociação é estabelecer um *contrato* [1] entre os usuários e provedores do serviço, a partir da instanciação dos diversos parâmetros de QoS da camada de aplicação. Para isso, dois passos são necessários:

1. *testes de admissão*: usados na comparação do recurso requisitado (requisitos de QoS obtidos através de mapeamento) com os recursos disponíveis no sistema, em todas as camadas da arquitetura; e

2. *reserva de recursos*: aloca os recursos necessários para o fluxo entre os sistemas finais emissor e receptor, com valores de QoS suficientes para que a QoS solicitada pelo usuário seja fornecida.

2.2.2 Sessão

Durante a *realização da sessão*, ocorrem as seguintes etapas:

Policiamento de QoS

A etapa de policiamento de QoS é definida como um conjunto de ações tomadas pelo sistema, para verificar se a QoS contratada está sendo seguida pelo usuário, através do monitoramento do sistema e do controle do tráfego sobre o usuário, de modo a proteger os recursos do sistema de uma má conduta do usuário, o que pode afetar as garantias de QoS existentes. Em outros termos, o policiamento de QoS permite reagir quando o usuário viola os valores de QoS inicialmente especificados.

Monitoramento de QoS

A etapa de monitoramento visa verificar se a QoS contratada pelo usuário está sendo mantida pelo provedor, através da observação do valor de certos parâmetros como, por exemplo, a taxa de perdas de unidades de transporte, a taxa de erros de bits e a taxa de perdas de "deadlines".

O monitoramento possui duas principais tarefas:

1. *detectar e notificar qualquer violação de QoS*: quando o valor de um determinado parâmetro de QoS não respeita o valor contratado, uma notificação é gerada, indicando uma violação e a possível causa

2. *armazenar informação*: uma descrição da informação a ser selecionada, e uma descrição do trabalho computacional a ser executado sobre a informação armazenada quando requerida.

O monitoramento de QoS é uma atividade importante também para outras etapas do controle QoS da sessão, tal como o policiamento, a adaptação e a renegociação de QoS iniciada pelo sistema. Os dados monitorados podem ser usados para produzir estatísticas úteis como, o número de violações de um determinado tipo, o tempo da última violação, a média de tempo entre violações, o tempo gasto para se recuperar das violações e as causas mais freqüentes de violações. Tais estatísticas podem ser usadas para o planejamento de capacidade, analise de congestionamento, cálculo de componentes disponíveis e desenvolvimento de sistema.

Renegociação da QoS

A renegociação pode ser inicializada pelo usuário ou pelo sistema, visa adequar a QoS da sessão da aplicação ao contexto do ambiente quando o nível mínimo de QoS contratado não pode mais ser mantido, ou se o usuário está insatisfeito com os níveis contratados, desejando alterar os limites máximo e mínimo. Esta etapa é crucial se as aplicações/usuários não são capazes de especificar a QoS necessária para toda a duração do serviço. Por outro lado, a renegociação iniciada pelo sistema ocorre devido à falta de recursos e objetiva reduzir a QoS fornecida, para não interromper os serviços fornecidos. Tipicamente a renegociação deve ser inicializada somente quando uma adaptação automática não é possível.

A etapa de renegociação usa os mesmos mecanismos da etapa de negociação, com a vantagem do conhecimento "a priori" do contrato da etapa de negociação.

Adaptação de QoS

Visa adequar a QoS da sessão da aplicação ao ambiente quando o nível corrente de QoS não pode mais ser mantido, ou quando o usuário deseja alterar o nível corrente dentro dos limites máximo e mínimo contratados.

O mecanismo de adaptação é executado para reagir a mudanças no contexto do ambiente, toda vez que uma violação de QoS é detectada, mantendo o serviço e eventualmente baixando a QoS em casos de escassez de recursos.

Geralmente, a sobrecarga de recursos pode ser resolvida por políticas definidas pelo usuário/aplicações. Assim, ao detetar uma violação de QoS, o sistema pode usar uma política adequada para reagir a situações de sobrecarga. Em suma, o ideal é que a etapa de adaptação seja executada automaticamente pelo próprio sistema, só requerendo intervenção do usuário/aplicação quando todas as demais estratégias falharem.

Contabilização de Custos da QoS

Realiza as operações de cálculo dos custos do serviço para o usuário, o que pode ser uma atividade bastante complexa, tendo em vista que o SMD pode fornecer uma vasta gama de valores de QoS.

2.2.3 Término da Sessão

No *término da sessão da aplicação* ocorre a *liberação de QoS*, quando os recursos alocados são liberados e as tarefas relacionadas são finalizadas. Durante o término da sessão uma mensagem de notificação deve ser enviada aos componentes de sistema envolvidos no fornecimento do serviço, para que os recursos sejam liberados e atualizados.

2.3 QoS em Aplicações Multimídias Distribuídas

Para suportar os requerimentos das aplicações multimídias distribuídas, todos os componentes do sistema envolvidos devem suportar um determinado valor de QoS de maneira que durante toda a comunicação fim-a-fim (entre sistemas finais), esses requerimentos sejam satisfeitos. Existem basicamente duas abordagens para o gerenciamento de QoS em aplicações multimídia [11]: com e sem reserva de recursos. Na primeira, um gerenciador de QoS verifica se existem recursos suficientes disponíveis para garantir o mínimo de QoS em uma comunicação, caso exista, o gerenciador reserva estes recursos e dispara a aplicação. Nesta abordagem o gerenciador precisa monitorar o uso e a disponibilidade de recursos, mapeando os requerimentos de QoS da aplicação para a necessidade de recursos do sistema. Durante a comunicação existe a possibilidade do ajuste da QoS do sistema dentro de um limite máximo e mínimo, de acordo com a vontade do usuário. Na segunda abordagem, o gerenciamento de QoS [1] é proposto sem garantias de QoS em um sistema melhor-esforço, sendo fornecido aos usuários o melhor serviço possível. Esta abordagem assume que a aplicação deve ter um conhecimento prévio do sistema, reagindo dinamicamente para fornecer a melhor qualidade de acordo com os recursos disponíveis. Esta é a solução adotada por *aplicações adaptativas*, que fornecem a melhor QoS possível com o mínimo de degradação e o máximo de robustez. No contexto de SMD's, a adaptação é um processo complexo que envolve uma grande quantidade de componentes do sistema, e que tem por objetivo aumentar a faixa de condições sobre a qual uma aplicação pode ser executada satisfatoriamente.

[1] Alguns autores não consideram este caso como gerenciamento de QoS.

2.3.1 Gerenciamento no Caso de Reserva de Recursos

Sistemas com reserva de recursos garantem que as aplicações só serão executadas se um mínimo de qualidade pode ser oferecido. Os valores de QoS são especificados pelo usuário e transmitidos aos gerenciadores de QoS. A fase de negociação entre aplicações e o gerenciador de QoS determina se o valor da qualidade requerida pelo usuário pode ser garantida. Esta etapa tem início com o mapeamento das especificações de QoS da aplicação, o qual define o modelo de comunicação[2] para as aplicações. Esses modelos definem a quantidade de recursos do sistema que devem ser alocados, de forma a garantir a QoS especificada. Esta etapa é finalizada quando existirem recursos suficientes disponíveis para a execução da aplicação, através de sua reserva. Diz-se então que a aplicação tem garantias de QoS.

2.3.2 Gerenciamento no Caso Sem Reserva de Recursos

Em ambientes melhor-esforço onde não existe a facilidade de reservas de recursos, a fase de negociação é simplificada. A especificação, o mapeamento de QoS e a determinação do modelo de comunicação neste caso, não são necessários. O gerenciamento de QoS é executado pelas próprias aplicações. Nesta abordagem, o usuário seleciona as regras para a adaptação, isto é, os serviços e as modificações necessárias para o fornecimento da melhor qualidade para o usuário, diante de flutuações e sobrecarga no SMD. Segundo Gecsei [13], a adaptação pode ser definida em dois espaços (figura 2.2): um relacionado a performance "P", e outro relacionado aos recursos "R".

O domínio de P inclui medidas de QoS orientado ao usuário, tais como qualidade da reprodução, tempo de resposta, funcionalidade, custo e a possibilidade de outros fatores como o uso dos recursos e justiça com outros usuários. A região de aceitação, AR, é uma região dentro de P na qual o SMD é capaz de fornecer os recursos desejados. O domínio de R corresponde as características dos recursos do ambiente de operação (UCP, memória, largura de banda, "jitter", taxa de perda, etc...).

Para simplificar, suponha que para uma determinada classe de aplicação exista um mapeamento $M : R \leftrightarrow P$, que mapeia AR em uma região B em R. Acontecendo mudanças de adaptação M tal que AR agora é mapeado em uma grande região A.

[2] O modelo de comunicação engloba todos os dispositivos entre os sistemas finais inclusive.

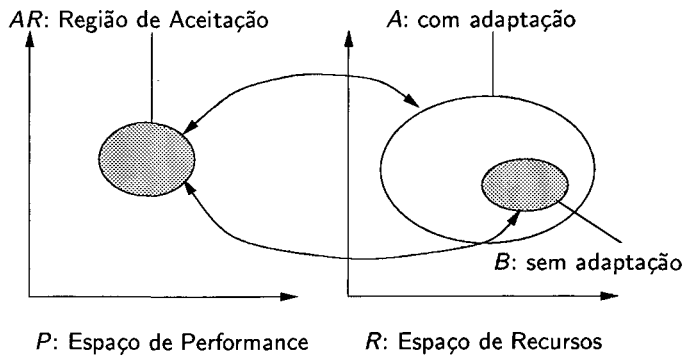

Figura 2.2: Efeito da adaptação em variações aceitáveis no espaço de recursos.

Normalmente, é de se esperar que A inclua B; o tamanho de $(A - B)$ representa a eficácia da adaptação.

Dinamicamente, a adaptação em SMD's trabalha em cima de um conjunto de mecanismos, cujo objetivo é manter o ponto de operação (OP) dentro de AR. Quando o ponto de operação se move para fora desta região, um controlador inicia alguma ação corretiva chamada de *adaptação*, a qual é responsável por tentar trazer o ponto de operação OP de volta para a região de aceitação AR. O controlador faz isso através do monitoramento dos valores dos parâmetros observados (coordenadas em R e P), e executa algum algoritmo de adaptação. Esta ação resulta na mudança dos parâmetros de controle:

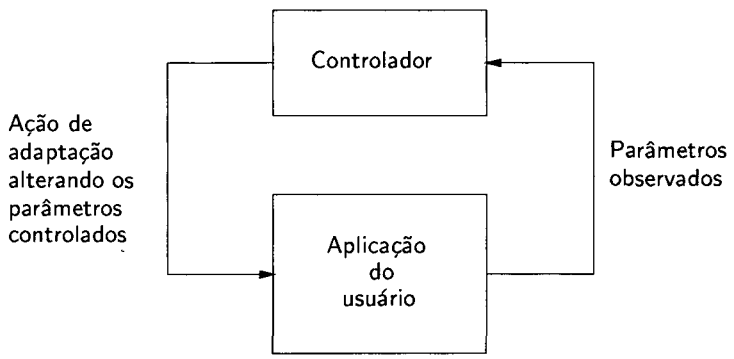

Figura 2.3: Mecanismo básico de adaptação em um SMD.

A figura 2.3 possui os seguintes elementos, cuja natureza e localização do controlador no sistema definem diferentes tipos de adaptação:

1. *localização do controlador*: podendo ele residir com o usuário (por exemplo, quando se prioriza a alta resolução de imagem em certo ponto da aplicação), na aplicação (quando se prioriza o aumento da taxa de perda de quadros) ou em diferentes pontos do sistema.

2. *o componente do SMD que se adapta* (pois a ação afeta alguns de seus parâmetros e conseqüentemente seu comportamento): neste caso adaptação pode ocorrer em S (por exemplo, quando chaveando entre protocolos diferentes, diminuindo a taxa de transmissão, ou alocando mais ciclos de UCP para um processo), em A (por exemplo, mudando as cores de colorido para preto-e-branco) ou em U (por exemplo, aceitando o áudio com qualidade de telefone).

3. *parâmetros observados*: tais como satisfação do usuário frente a qualidade do contexto corrente, ao "jitter" de transmissão e a taxa de perdas, que podem estar relacionados com qualquer componente do SMD. Deve também ser considerado o método de observação: "polling", monitoramento, requisição de exceção, demanda ou intervenção do usuário.

Na literatura existem diversos trabalhos implementados ou propostos em diferentes contextos, que podem ser classificados em três grupos [13] [3]: centrado no usuário, "Mobile Open System Technologies" - **MOST**[9], "The Krakatoa Chronicle"[17], centrado no sistema, "The INRIA Videoconferencing System" - **IVS**[4] e misto, "Fast Web"[12], "Multimedia News on Demand"[34].

2.4 Conclusões

O controle de QoS em SMD's tem um papel importante quando o objetivo é o de oferecer bons serviços para os usuários. A definição de um modelo de QoS é a base para se criar uma estrutura para controle de QoS entre sistemas finais em um SMD, tal que ele possa dar conta das necessidades do usuário. O modelo aqui definido é composto de várias etapas que são realizadas de acordo com o andamento da sessão, sendo válido tanto para sistemas que oferecem serviços com garantia de recursos quanto

[3] Em Gecsei[13] existe uma pequena descrição e um estudo comparativo de alguns trabalhos propostos e implementados aqui citados.

sem garantias, diferindo no caso dos sistemas sem garantias de recursos, nos quais algumas etapas do modelo não são necessárias.

O modelo aqui definido serve de base para a implementação do mecanismo de controle de QoS sobre a aplicação de vídeo-conferênciav VIC, tal mecanismo é norteado pelo modelo de QoS mais restrito, voltado para ambientes melhor-esforço na qual a aplicação alvo está inserida.

Capítulo 3

A Aplicação de Vídeo-Conferência – VIC

3.1 Introdução

O desenvolvimento do "IP Multicast" [3, 27] tem alimentado o desenvolvimento de várias ferramentas (aplicações) coletivamente conhecidas como ferramentas "MBone", para conferências multimídias em tempo-real. A tecnologia "IP Multicast" extende o modelo de roteamento tradicional, a um eficiente modelo de entrega de pacotes multi-ponto, através da criação de uma rede virtual no topo da Internet existente conhecido como o **Multicast Backbone** – "MBone".

As primeiras aplicações a transmitirem vídeo via "Mbone" foram o **Xerox PARC Network Video Tool**, *nv* e o **INRIA Video Conferencing System**, *ivs*. Ambos os sistemas suportam uma baixa taxa de transmissão de bits de dados de vídeo na Internet. Como se tratam dos primeiros trabalhos em transmissão de vídeo, algumas restrições foram impostas para facilitar o processo de desenvolvimento das mesmas. Para contornar algumas dessas limitações, a aplicação "Video Conference Tool – VIC" [23] realizada na UCB/LBL, foi desenvolvida enfocando principalmente a flexibilidade. Ela é uma aplicação extensível, orientada a objetos, cuja estrutura suporta: múltiplas abstrações de rede, CODECS baseados em "hardware", um modelo de coordenação de conferência, uma interface de usuário extensível e diversos algoritmos de compressão de vídeo, além de ser implementado segundo as especificações do RTP (Protocol Tempo-

Real) [31].

Neste capítulo será apresentada uma descrição da estrutura do VIC, tanto do ponto de vista da arquitetura de rede quanto da arquitetura da própria aplicação. As partes do código fonte responsáveis pelo controle de fluxo e estruturação dos dados são descritas no Apêndice A, e destacam principalmente a parte do código necessária para a implementação do mecanismo de controle de QoS, que é o objetivo do trabalho.

3.2 Arquitetura de Rede

Desde que o "Mbone" foi criado para estudar questões de escalonabilidade multiponto, a interoperabilidade se tornou um fator importante. Como o princípio de funcionamento do roteamento "multicast" se baseia na motivação de que um grupo de pessoas espalhadas geograficamente por uma vasta região pode estar interessado em trocar informações entre si, tem-se aí um excelente ponto de partida para o desenvolvimento de aplicações que explorem tal interoperabilidade.

3.2.1 Protocolo de Transporte Tempo-Real – RTP

Para promover tal interoperabilidade, o grupo de trabalho de **Transporte de Áudio/Vídeo** da **Internet Engineering Task Force** - IETF desenvolveu o RTP um protocolo que trabalha ao nível de aplicação para o transporte de dados multimídia. O RTP tem por objetivo fornecer uma *fina* camada de transporte integrada à aplicação, sem comprometer seu funcionamento. Ele é um protocolo fim-a-fim que não tem a noção de conexão, podendo operar sobre protocolos de camadas inferiores com ou sem conexão (por exemplo TCP/UDP, figura 3.1). O RTP também não possui dependências de formatos de endereços e somente necessita que as camadas inferiores trabalhem com o transporte de informações em pacotes.

O RTP consiste de duas partes, uma para a transmissão e outra para controle (RTCP) dos dados.

Pacote de Dados RTP

O pacote de dado RTP consiste de um cabeçalho de 12 bytes seguido pelo "payload", que indica o formato dos dados que estão sendo transmitidos, por exemplo, se é um

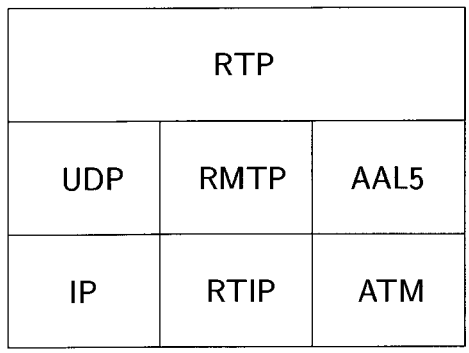

Figura 3.1: RTP e a pilha de protocolos.

quadro de vídeo JPEG ou uma amostra de áudio GSM. Algumas das informações contidas no cabeçalho são por exemplo: o tipo de "payload", o "timestamp", que descreve o instante em que o dado contido no pacote foi gerado, o "sequence number" que permite fazer a detecção da perda e o seqüenciamento de pacotes dentro de uma série de pacotes com o mesmo "timestamp", o "synchronization source identifier – SSRC" um campo de 32 bits que identifica um participante único dentro de uma sessão, etc.

3.2.2 Protocolo de Controle Tempo-Real – RTCP

O RTP oferece um protocolo de controle chamado RTCP que suporta as suas funcionalidades. Uma mensagem RTCP consiste de um certo número de pacotes empilháveis, cada um de um certo tipo e tamanho. O seu formato é muito similar ao dos pacotes de dados. Os pacotes RTCP são enviados periodicamente para o mesmo grupo "multicast" dos pacotes de dados, servindo também para indicar quais os usuários estão conectados à sessão, mesmo que não estejam transmitindo dados. Algumas das funcionalidades do RTCP são descritas a seguir:

1. *monitoramento de QoS e controle de congestão*: os pacotes RTCP contém a informação necessária para monitoramento de QoS, já que os pacotes são enviados a todos os membros da sessão, assim todos podem examinar como está se apresentando o desempenho dos participantes. Aplicações que recentemente enviaram dados geram um pacote com um *relatório de emissor*, que contém informações

úteis para sincronização entre mídias (áudio e vídeo), assim como um contador para os pacotes e a quantidade de bytes enviados permitindo estimar a taxa de dados atual. Os membros que estão recebendo dados em uma sessão geram pacotes com um *relatório de receptor*, informando sobre o maior número de seqüência recebido, o número de pacotes perdidos, o "jitter" entre pacotes e o "timestamp" de acordo com os dados de cada membro ativo na sessão.

2. *identificação*: os pacotes de dados RTP identificam a sua origem única, somente através de um número de 32 bits gerado randomicamente. Um dos pacotes RTCP é o SDES (Descrição de usuário) composto de vários campos com informações sobre o usuário, como por exemplo, o CNAME (Nome canônico) que é um identificador global e único de cada participante da sessão.

3. *escala e estimação do tamanho da sessão*: determina como os pacotes RTCP são enviados periodicamente a todos os membros da sessão. O desejo de atualizar as informações de controle, deve ser balanceado frente o desejo de limitar o tráfego de controle a uma pequena percentagem do tráfego de dados, mesmo com a sessão consistindo de um grande número de membros. A carga do tráfego de controle é definida em torno de 5% do tráfego de dados.

3.3 Arquitetura do VIC

A arquitetura do VIC é construída sobre um modelo dirigido a eventos, com a otimização do caminho do fluxo de dados e controlado por uma estrutura flexível implementada em Tcl/Tk [37, 36]. Um conjunto de objetos básicos é implementado em C++ [8] e coordenado via Tcl/Tk. Partes do código em C++ implementam comandos Tcl permitindo manipular objetos e construir o caminho do fluxo de dados usando uma interface uniforme, enquanto classes derivadas suportam instâncias específicas de dispositivos de captura, tipos de monitores, algoritmos de codificação, módulos de decodificação, etc.

O uso de "scripts" em Tcl/Tk é o responsável pela construção da interface, pelo caminho do fluxo dos dados e pelo controle de eventos resultantes de ações da rede ou da intervenção do usuário. Como toda interface é implementada em Tcl/Tk, se

torna bem mais prático controlar as funcionalidades do VIC, através de um simples elemento da interface, que invoca uma série de primitivas de controle que implementam tal funcionalidade.

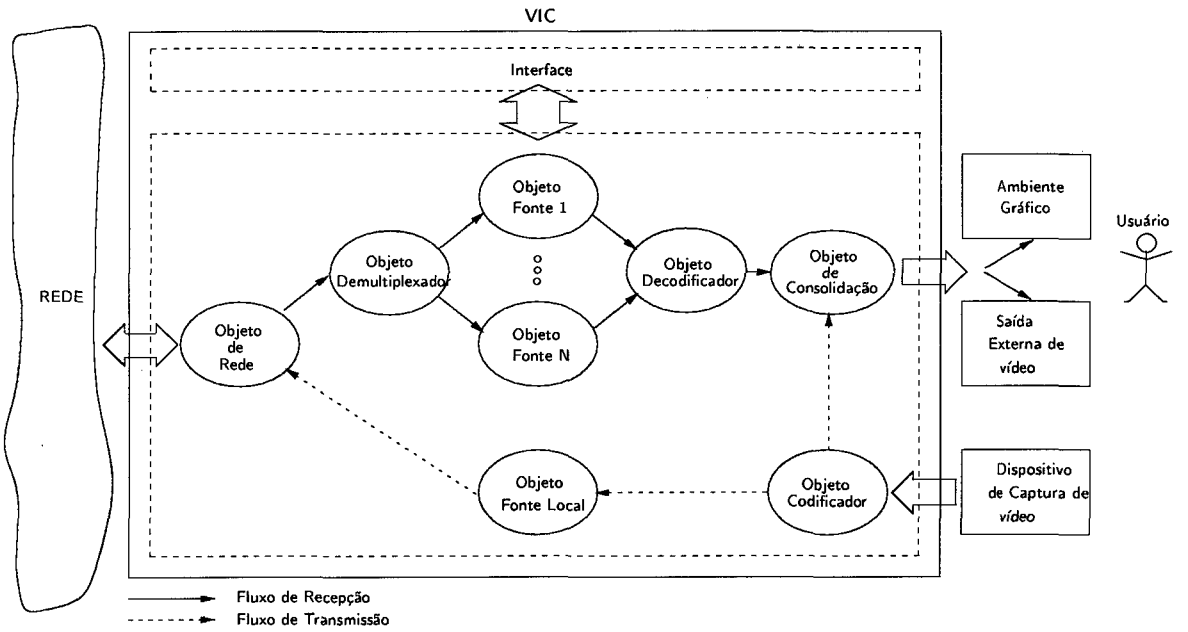

Figura 3.2: Arquitetura do VIC.

A figura 3.2 mostra a arquitetura geral do VIC onde pode ser visto o caminho de transmissão/recepção do fluxo de dados. As elipses representam os objetos implementados em C++ tudo controlado via "scripts" Tcl/Tk, enquanto os retângulos representam os dispositivos de saída (monitor, saída externa de vídeo, etc.) e captura de vídeo (câmera).

O *Objeto de Rede* é responsável pelo tratamento de todos os eventos relacionados à rede: transmissão, recepção e de/criptografia dos dados da sessão. O *Objeto Fonte* corresponde a instância de cada participante dentro da aplicação, inclusive a instância local do VIC. No VIC exitirão tantas instâncias quantos forem o participantes transmitindo dados na sessão. Este objeto faz o tratamento dos dados, separando os fluxos de vídeo de forma que eles não se interfiram. O *Objeto de Demultiplexação* que atua na recepção dos dados, e é o responsável por endereçar os pacotes que chegam da rede para a instância correta do participante dentro da aplicação, representada pelo objeto

fonte. Os dados relacionados a cada objeto são usados para gerar estatísticas de desempenho e se pode ter acesso a eles a partir da interface. O *Objeto Decodificador* é o responsável pela decodificação do fluxo de vídeo de cada participante, fluxo esse que pode ser gerado de acordo com vários tipos de algoritmos de compressão suportados pelo VIC, como por exemplo, JPEG, H.261, CellB, BVC, H.263, H.263+, NV, NVDCT e RAW. O objeto de decodificação tanto pode atuar via "software" ou "hardware", sendo que neste último caso é necessário um placa de captura de vídeo que suporte esta funcionalidade. O *Objeto Codificador* é o responsável por gerar o fluxo de dados no algoritmo de compressão desejado, aqui também a compressão pode ser feita tanto via "software" ou "hardware", com um dispositivo de captura de vídeo. O *Objeto de Consolidação*[1] é o responsável pela montagem da imagem que é exibida para o usuário local.

Do ponto de vista do código fonte, que é mais bem detalhado do Apêndica A, a estruturação do VIC é feita a partir da alocação de memória compartilhada feita de maneira específica para cada plataforma (SUN Solaris, IBM AIX, FreeBSD, Microsoft Windows, etc.) onde o VIC é executado, controle de aceitação dos parâmetros da linha de comando, inicialização do Tcl e do Tk, e dos recursos relacionados com a inicialização da instância local do VIC. A continuação da estruturação global de funcionamento é feita posteriormente por "scripts" em Tcl/Tk onde são inicializadas as funcionalidades do RTP/RTCP (SSRC, CNAME, etc.) relacionados a instância local do VIC, o *barramento de conferência* seção 3.3.5, o suporte de rede nativo (IP, RTIP ou ATM), e a construção da interface principal.

3.3.1 Fluxo de Recepção dos Dados

Quando um pacote é recebido da rede, ele é despachado pelo objeto de rede para o objeto demultiplexador. O demultiplexador é o responsável por todo o processamento RTP do pacote, enviando-o posteriormente ao objeto fonte, que representa um participante específico ativo na sessão. Caso não exista um objeto fonte relacionado a um determinado pacote, uma chamada dentro do Tcl é feita para instanciar um novo caminho para o fluxo de dados (novo usuário). Uma vez que o caminho é estabeleci-

[1] O termo consolidação é usado neste trabalho, em substituição ao termo em inglês "rendering", que representa a construção de uma imagem em três dimensões a partir de um modelo matemático.

do os pacotes fluem do objeto fonte para o objeto decodificador. Para cada instância de participante existe uma instância do objeto decodificador que faz a decodificação dos dados de acordo com o formato do algoritmo de codificação especifico. Quando o objeto de decodificação decodifica um quadro completo, objeto de consolidação é posteriormente instânciado para exibir o quadro no dispositivo de saída.

3.3.2 Fluxo de Transmissão dos Dados

O fluxo de vídeo que vem do dispositivo de captura é enviado ao objeto de codificação, que executa tanto a compressão dos dados segundo o algoritmo da compressão desejado, quanto o empacotamento dos dados via RTP. Diferentes algoritmos de compressão requerem diferentes formatos de vídeo, por exemplo, o padrão **H.261** requer o vídeo no formato 4:1:1 na geometria CIF, enquanto o padrão **NV** requer o vídeo no formato 4:2:2 em qualquer geometria. Uma solução que chegou a ser proposta para a implementação da aplicação por seus desenvolvedores, era a de fazer com que o dispositivo de vídeo capturasse e exportasse o vídeo em um formato comum, que posteriormente seria convertido no formato desejado pelo padrão de codificação. O que ocorre é que infelizmente a manipulação de dados de vídeo não comprimido resulta em um grande sobrecarga de processamento. A solução adotada então é a de suportar vários formatos, sendo que outros também pudessem ser incorporados à aplicação.

O ganho de performance é feito realizando um *reabastecimento condicional* ("conditional replenishment") sempre que possível. Muitos dos algoritmos de compressão utilizam o modelo de reabastecimento condicional, em que nele a imagem capturada (quadro) é dividida em pequenos blocos, e somente os blocos que variam acima de um certo limiar são codificados e enviados. A decisão de enviar os blocos depende de uma pequena variação dinâmica de "pixels" do respectivo bloco. Conseqüentemente, se a decisão de não enviar determinado bloco é adotada, muito da sobrecarga de leitura/escrita na memória é evitada.

3.3.3 Consolidação

Outra tarefa de performance crítica no VIC, é a conversão do vídeo da representação YUV dos "pixels" usada por muitos algoritmos de compressão em um formato

adequado a ser exibido no dispositivo de saída, ou seja, a "consolidação" da imagem. A profundidade do "pixel" pode ser dividida em três campos que representam dois elementos: a cor (primeiro campo) e a luminância (os outros dois campos) que carrega informações sobre brilho e contraste. A relação existente entre o número de bits usado para representar cada campo é chamada de relação entre os componentes YUV, sendo que as mais comuns são 4:2:0, 4:2:2 e 4:4:4.

O vídeo a ser exibido pode ser "consolidado" tanto em uma saída externa de vídeo quanto em um ambiente gráfico ("X Server"). No caso de ambientes gráficos são necessários algoritmos de suavização da imagem para sua exibição, ou pode-se simplesmente converter a representação da relação YUV para o formato RGB, de forma que em ambos os casos as imagens possam ser mapeadas em monitores ("displays") em cores. Uma hierarquia de classes construída em C++ é usada para suportar todos os modos de operação, especialmente no caso de vídeos em 4:1:1 e 4:2:2 para otimizar a performance.

Para o mapeamento de cores em ambientes gráficos o VIC suporta diversos algoritmos de suavização. O padrão é o da difusão de erro ("error-diffusion") realizado no domínio da relação YUV, o que produz uma imagem de alta qualidade que o por sua vez é computacionalmente custoso. O VIC possui um outro algoritmo de suavização, que caso a performance se torne uma tarefa crítica, pode ser utilizado pelo usuário se desejado. Este algoritmo é menos custoso computacionalmente que o uso direto da quantização de cores. Nela, uma paleta de cores é otimizada a partir do ambiente gráfico e então cada "pixel" é quantizado com a cor mais próxima dentro da paleta. Esta abordagem produz uma imagem de qualidade razoável quando o mapeamento de cores na paleta é corretamente escolhido.

A otimização da "consolidação" final é feita de maneira que somente seja suavizado as regiões onde a imagem é alterada. Assim, cada decodificador mantém um controle dos blocos que são atualizados em um quadro e "consolida" somente esses blocos. Os "pixels" são "consolidados" em um "buffer" compartilhado entre o ambiente gráfico e a aplicação, tal que somente uma cópia dos dados seja necessária para atualizar o monitor com um novo quadro de vídeo.

3.3.4 Privacidade

Para oferecer confidencialidade a uma sessão, o VIC criptografa os dados entre os sistemas finais por especificação do RTP, se desejado. Este modelo fim-a-fim assume que a rede pode ser facilmente monitorada, e assim a criptografia previne que receptores não desejados possam receber os dados da transmissão. Em uma sessão privada o VIC criptografa todos os pacotes, como sendo a última operação no caminho de transmissão do fluxo de dados, e decriptografa os pacotes como a primeira operação no caminho de recepção do fluxo de dados. A chave de criptografia é especificada para os participantes da sessão através de algum mecanismo externo de comunicação seguro.

O VIC suporta diversos algoritmos de criptografia implementados através de uma hierarquia de classes C++. O DES ("Data Encryption Standart") é o algoritmo padrão do VIC.

3.3.5 Barramento de Conferência

Desde que várias mídias em uma conferência podem ser manipuladas por diferentes aplicações, é necessário que haja um mecanismo para coordenação entre essas diversas aplicações. A abstração do *barramento de conferência* (figura 3.3), oferece esse mecanismo. Cada aplicação envia via "broadcast" sua mensagem no barramento, e todas as aplicações registradas para receber este tipo de mensagem recebem uma cópia.

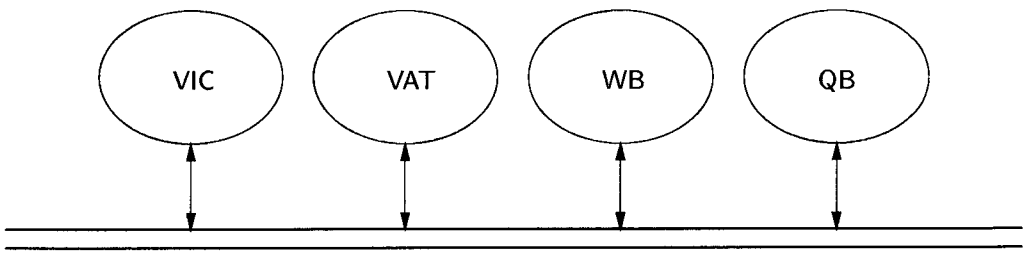

Figura 3.3: Barramento de Conferência

A figura 3.3 retrata o barramento de conferência em uma sessão formada por diversas aplicações: de áudio (VAT)[2], de vídeo (VIC) e quadro negro (WB)[3], tudo coorde-

[2]Visual Audio Tool do Lawrence Berkeley Laboratory
[3]White Board do Lawrence Berkeley Laboratory

nado através de uma aplicação de coordenação (QB)[4].

Os principais mecanismos suportados pelo VIC no modelo do barramento de conferência são:

1. *janelas chaveadas por voz*: neste caso o VIC usa algumas características do VAT, para determinar qual o membro que está correntemente falando na sessão. O que ocorre é que, o VAT envia uma mensagem para o barramento indicando qual membro está atualmente falando, indicando o seu CNAME (seção 3.2.2). O VIC monitora esta mensagem chaveando para a janela do respectivo membro. Se existem múltiplos membros transmitindo na sessão a janela do mais recente membro a falar é exibida.

2. *controle da palavra*: o VIC tem implementado um mecanismo que permite o controle da palavra a partir de uma aplicação externa, tal que todos os participantes de uma sessão possam seguir os passos definidos por um moderador, sendo ele responsável por conceder a palavra ao membro que ele achar mais adequado de acordo com sua política de controle. Localmente só são exibidos os dados enviados pelo membro que possui o controle da palavra.

3. *sincronização*: a sincronização entre mídias diferentes também pode ser realizada no barramento de conferência. Cada aplicação tempo-real possui um *ponto de reprodução* usado para adaptar a reprodução dos dados a variações de atraso na transmissão. Este ponto de reprodução pode ser ajustado para sincronização entre mídias.

4. *acesso a dispositivos*: cada sessão ativa possui um barramento de conferência para coordenar uma determinada mídia dentro da sessão. Como em algumas operações de coordenação o acesso a dispositivos de "hardware" requer uma interação entre diferentes sessões, o VIC usa uma barramento de conferência global compartilhado entre as diferentes sessões, a partir do qual sessões diferentes possam ter acesso a dispositivos em comum.

[4]Questionary Board do Lawrence Berkeley Laboratory

3.3.6 Interface do Usuário

A interface do usuário é totalmente implementada com "scripts" em Tcl/Tk. Isto permite que com um pequeno conhecimento da linguagem, seja relativamente fácil de se alterar a interface. Outra facilidade oferecida pelo VIC é que em tempo de execução o usuário pode incluir algum código adicional, para alterar a interface via um arquivo de configuração no diretório do usuário (arquivo *.vic.tcl*).

Uma deficiência da interface é que ele é completamente independente das interfaces de outras ferramentas de transmissão de mídia, o ideal seria que toda a apresentação fosse exibida de maneira uniforme em uma conferência, de forma que todas as aplicações pudessem compartilhar de uma única interface.

O VIC consiste de uma janela principal (de usuários) onde o fluxo de vídeo dos participantes de uma mesma sessão é exibido. Para destacar o vídeo de um determinado participante basta somente clicar sobra a sua janela. A janela de configuração local ("Menu") pode ser acessada a partir da janela de usuários. Nela é permitido alterar as configurações de alguns parâmetros para a própria transmissão de vídeo. Esta janela é dividida em quatro sub-janelas, cujas principais funções são:

1. *transmissão* ("Transmission"): para a configuração da banda de transmissão e do número de quadros por segundo a serem gerados;

2. *codificação* ("Encoder"): para seleção do algoritmo de compressão (NV, NVD-CT, CellB, M-JPEG, H.261, BVC, H.263+, H.263 e RAW), resolução da janela ("small", "normal" e "large"), qualidade da imagem[5] para configuração do fator de quantização, selecionar também se a compressão será executada por "hard-ware", etc.

3. "*display*": que define como será o processo de suavização da imagem, se será usado "hardware" para decodificação do vídeo, etc.

4. *sessão* ("Session"): que lista os dados locais relacionados com a identificação do usuário e da sessão, também lista todos os membros participante da sessão,

[5]Posteriormente esse parâmetro será referenciado como **fidelidade** da imagem, já que o termo qualidade tem um escopo muito mais abrangente.

estatísticas globais, permite entrar com a chave de criptografia para sessões privadas, etc.

A estrutura da interface de usuário está implementada no "script" *ui-main.tcl*. É ele o responsável pelo controle sobre os fluxos de vídeo dos participantes ativos na sessão, adição/remoção de membros, atualização da exibição dos dados. A janela de "Menu" é toda definida no "script" *ui-ctrlmenu.tcl*. Neste arquivo está implementada toda a estrutura de controle sobre os parâmetros de configuração do usuário local.

3.4 Conclusão

A implementação de ferramentas multimídia usando duas linguagens de programação, uma para controle do fluxo de dados, o Tcl/Tk e outra para a implementação das estruturas de dados, o C++, se mostra bastante adequada como é o caso do VIC, quando o objetivo é o de oferecer flexibilidade no desenvolvimento da aplicação, de modo que ela possa ser facilmente modificada no futuro sem a necessidade de se reprojetar toda a aplicação. Como por exemplo, a capacidade da aplicação de suportar diversas arquiteturas de redes, algoritmos de compressão, etc., torna-se um atrativo a mais para que o usuário possa obter um melhor desempenho da aplicação de acordo com suas necessidades. O modelo de reabastecimento condicional também se mostra adequado no ganho de performance, principalmente em conferências onde a movimentação é pouca, evitando assim, que cada quadro seja completamente processado, não comprometendo tanto dos recursos da máquina local já que a tarefa de "consolidação" da imagem é bastante crítica em termos de performance.

Nos dois capítulos seguintes é mostrado a implementação de dois mecanismos de controle de QoS: um com escopo local; e outro com escopo global, cuja aplicação alvo em ambos é o VIC.

Capítulo 4

Mecanismo de Adaptação Local da Qualidade de Serviço

4.1 Introdução

Geralmente, a sobrecarga do processador do sistema final receptor de um SMD acaba induzindo à perda de "deadlines" por parte da tarefa de descompressão. Esta perda de "deadlines" na camada da aplicação, implica em uma perda de quadros com a conseqüente redução da freqüência de exibição, ou seja, a freqüência de quadros é o parâmetro de QoS penalizado em decorrência do problema supramencionado. Parâmetros como cor, fidelidade da imagem e resolução não são afetados por sobrecargas de processador.

O fato da freqüência de quadros ser praticamente o único parâmetro afetado pela sobrecarga do processador, tem direcionado a maior parte dos mecanismos de adaptação de QoS propostos na literatura a atuarem apenas sobre esse parâmetro, de forma direta nas ações de adaptação, através de uma supressão selecionada de quadros, e de forma indireta no monitoramento pela observação da taxa de perdas de "deadlines".

Contudo, a solução mais abrangente para tentar diminuir essa queda da freqüência de quadros é a degradação de outros parâmetros, visando diminuir o tempo de processamento da tarefa de descompressão, com a conseqüente diminuição da perda de "deadlines".

Neste capítulo, é descrito a implementação de um mecanismo de adaptação de QoS

de escopo local (sistemas finais), implementado sobre a aplicação de vídeo-conferência VIC, que segue uma política de adaptação *centrada no usuário final.* Segundo essa política, a qualidade deve ser analisada como um todo e não através de parâmetros individuais. Para a implementação desta política, o mecanismo de adaptação faz uso de uma função chamada *grau de qualidade*[1] que é fruto do trabalho desenvolvido em [18, 20], cujos argumentos são todos parâmetros da camada de aplicação configuráveis pelo usuário. Em seguida são apresentados os resultados e a discussão sobre eles, assim como algumas conclusões.

4.2 A Função Grau de Qualidade

Como definido em [19, 20], em SMD's os parâmetros de QoS como freqüência de quadros, cor, fidelidade da imagem[2] e freqüência de amostras de áudio, possuem a capacidade de serem adaptados à carga da UCP ou à disponibilidade de largura de banda, mudando seus valores correntes para outros obtidos a partir do mapeamento (empírico) desses parâmetros em termos de recursos. Entretanto, esta abordagem se mostra incompleta haja visto que várias combinações de valores de parâmetros de QoS, com exigência de recursos similares, podem representar qualidades totalmente distintas do ponto de vista do usuário.

A definição de uma *função grau de qualidade* que associa um grau a cada combinação de valores dos parâmetros de QoS, permitirá disponibilizar um critério de seleção mais otimizado, da melhor combinação diante do contexto corrente do SMD.

O *domínio de um parâmetro de QoS* ρ_i, é um conjunto $\Omega_{\rho_i} = [\rho_{i_{min}}, \rho_{i_{max}}]$ ou $\Omega_{\rho_i} = \{\rho_{i_{min}}, \rho_{i_{min}} + 1, ..., \rho_{i_{max}}\}$ $(i = 1, 2, ..., n)$.

Um *nível de QoS* L, é uma n-tupla $< \rho_1, \rho_2, ..., \rho_n >$ representando uma combinação de valores dos n parâmetros de QoS de uma determinada camada da arquitetura do SMD (neste trabalho, parâmetros da camada de aplicação).

O *conjunto de todos os níveis de QoS*, $\Omega_{NiveisQoS}$, é o produto cartesiano dos n

[1] A função grau de qualidade aqui descrita tem um escopo genérico, no sentido de que ela contempla vários parâmetros de QoS da aplicação alvo. No caso específico do VIC serão abordados apenas dois parâmetros da camada de aplicação: fidelidade da imagem e freqüência de quadros.

[2] A fidelidade da imagem aqui é um parâmetro representado quantitativamente pelo fator de quantização usado pelo algoritmo de compressão e que indica quão próxima da original é a imagem digital.

domínios Ω_{ρ_i}, i.e., $\Omega_{NiveisQoS} = \Omega_{\rho_1} \times \Omega_{\rho_2} \times ... \times \Omega_{\rho_n}$.

O *grau de qualidade*, $Q\hat{o}S$, é uma métrica definida arbitrariamente como incluída no domínio $[0, 1]$ e que permite diferenciar quantitativamente os níveis de QoS. O valor de $Q\hat{o}S$ é obtido usando a função $\mathcal{QoS}^{\rho_1,\rho_2,...,\rho_n}_{\mapsto}[0, 1]$. Para o nível de QoS L, $Q\hat{o}S_L = \mathcal{QoS}(\rho_1, \rho_2, ..., \rho_n)$.

A função $\mathcal{QoS}$ é moldada de acordo com as seguintes premissas:

Premissa 1 *$\mathcal{QoS}$ é multidimensional, considerando o maior número possível de parâmetros de QoS da camada de aplicação.*

Premissa 2 *$\mathcal{QoS}$* **não é** *um mapeamento, ou seja, dois ou mais níveis de QoS podem ter o mesmo grau de qualidade, existindo, dentre outros, dois subconjuntos de níveis de QoS cujos elementos têm grau de qualidade 0 e grau de qualidade 1.*

Premissa 3 *$\mathcal{QoS}$ reflete a noção intuitiva de que a alteração da qualidade* não é linear *em relação à alteração do valor de um parâmetro ρ_i.*

Premissa 4 *$\mathcal{QoS}$ tende a fornecer um valor baixo para o grau de qualidade se um parâmetro ρ_i qualquer tende a $\rho_{i_{min}}$ (seu valor mínimo), independentemente dos valores dos outros parâmetros.*

Premissa 5 *$\mathcal{QoS}$ deve considerar pesos para cada parâmetro atribuídos (intuitivamente ou a partir de análises) de acordo com a natureza da aplicação.*

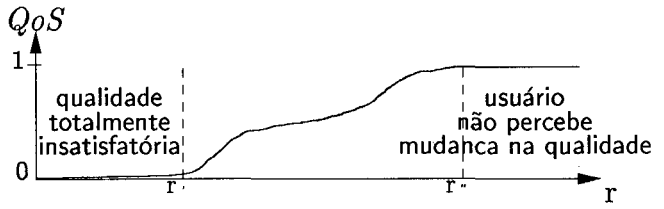

Figura 4.1: Tradução genérica de um parâmetro da aplicação para grau de qualidade [29].

A Premissa 1 limita $Q\hat{o}S$ a um fator de análise puramente qualitativa, não levando em conta a relação custo × benefício. A Premissa 2 reflete uma noção intuitiva em

relação à percepção do usuário quanto à qualidade: existem subconjuntos de níveis de QoS cujos elementos em termos de qualidade, são indistinguíveis entre si para o usuário final; um desses subconjuntos contém níveis de QoS cuja qualidade está aquém das expectativas mínimas do usuário, e outro contém níveis de QoS que representam a percepção máxima de qualidade por parte do usuário final. A figura 4.1, adaptada de [29], mostra esse comportamento para uma função grau de qualidade com um único argumento contínuo ρ. A Premissa 3 pode ser interpretada da seguinte forma: quanto mais baixo for o valor de um parâmetro de QoS, maior será a alteração do grau de qualidade quando o valor desse parâmetro é alterado, mantendo-se os outros constantes. A Premissa 4 também reflete uma noção intuitiva em relação à percepção do usuário final, para o qual a qualidade é baixa se um parâmetro de QoS tem um valor muito baixo, independentemente dos valores dos outros parâmetros. Por exemplo, se $\rho_1 \approx \rho_{1_{max}}$ mas $\rho_2 \approx \rho_{2_{min}}$ então $\mathcal{QoS}(\rho_1, \rho_2)$ tende a fornecer um valor baixo. A Premissa 5 reflete o fato de que, para diferentes aplicações, os parâmetros de QoS terão diferentes importâncias: em uma vídeo-conferência do tipo palestra, por exemplo, os pesos dos parâmetros relacionados à qualidade do som, devem ser maiores do que os pesos dos parâmetros relacionados à qualidade da imagem. Outra dificuldade em relação ao estabelecimento de pesos é que eles podem ser dinâmicos: em um determinado momento, o peso do parâmetro ρ_i na função $\mathcal{QoS}$ deve ser maior do que o do parâmetro ρ_j; em outro momento, essa situação pode ser inversa. No caso de um a vídeo-conferência novamente, se em determinado momento um dos conferencistas mostra alguma figura em que os detalhes são importantes, os pesos dos parâmetros relacionados à qualidade da imagem devem passar a ser superiores àquelas relacionadas à qualidade de som.

4.3 Metodologia para Obtenção de $\mathcal{QoS}$

A metodologia proposta neste trabalho para obtenção da função grau de qualidade é baseada na seção 4 de [22] e é composta dos seguintes passos:

1. atribuição de um grau de qualidade arbitrário para um subconjunto de instâncias de cada parâmetro ρ_i da camada de aplicação. Esse grau pode ser arbitrado de forma intuitiva ou obtido através de opiniões de um conjunto de avaliadores;

2. obtenção, através de interpolação, dos graus de qualidade para todas as instâncias de cada parâmetro ρ_i da camada de aplicação. Esse passo representa a construção de funções grau de qualidade para cada parâmetro individual, ou seja, a construção de n funções $\mathcal{QoS}_{\rho_i}(\rho_i)$ $(i = 1, 2, ..., n)$;

3. construção da função $\mathcal{QoS}(< \rho_1, \rho_2..., \rho_n >)$ a partir das n funções $\mathcal{QoS}_{\rho_i}(\rho_i)$. A definição do grau de qualidade associado ao k-ésimo nível de QoS $L_k = <\overset{k}{\rho_1}, \overset{k}{\rho_2}, ..., \overset{k}{\rho_n}>$ é dado pelo menor grau de qualidade dentre os n parâmetros, ou seja,

$$QoS_k = \mathcal{QoS}(<\overset{k}{\rho_1}, \overset{k}{\rho_2}, ..., \overset{k}{\rho_n}>) = min(\mathcal{QoS}_{\rho 1}(\overset{k}{\rho_1}), \mathcal{QoS}_{\rho 2}(\overset{k}{\rho_2}), ..., \mathcal{QoS}_{\rho n}(\overset{k}{\rho_n})) \quad (i = 1, 2, ..., n); \tag{4.1}$$

Essa escolha visa fazer com que a função obedeça à Premissa 4 do comportamento da função grau de qualidade. Alternativamente, se forem considerados os pesos (ω) para cada parâmetro, tem-se:

$$QoS_k = \mathcal{QoS}(<\overset{k}{\rho_1}, \overset{k}{\rho_2} ..., \overset{k}{\rho_n}>) = min(\mathcal{QoS}_{\rho 1}(\overset{k}{\rho_1}).\omega_{\rho_1}, \mathcal{QoS}_{\rho 2}(\overset{k}{\rho_2}).\omega_{\rho_2}, ..., \mathcal{QoS}_{\rho n}(\overset{k}{\rho_n}).\omega_{\rho_n}) \quad (i = 1, 2, ..., n); \tag{4.2}$$

4.4 Mecanismo de Adaptação

O mecanismo de adaptação implementado [22] segue a política que procura manter o maior grau de qualidade considerando o contexto corrente do processador.

Com relação à *quem adaptar*, a política escolhida visa envolver o maior número possível de parâmetros, no processo de adaptação (melhora ou degradação da QoS), de modo que cada um "colabore um pouco" para a manutenção do grau de qualidade em um determinado patamar. Essa escolha decorre do fato que o grau de qualidade é, intrinsecamente, uma função que abrange *todos parâmetros de QoS da camada de aplicação*, o que torna a própria freqüência de quadros um parâmetro candidato à degradação. Os parâmetros de QoS do VIC usados na adaptação são: a fidelidade da imagem e freqüência de quadros.

Com relação ao *quanto degradar*, a política arbitra o grau de qualidade $0,5$ como sendo limite máximo de degradação. Este valor corresponde ao mínimo aceitável para o usuário e é definido de maneira arbitrária. O limite máximo de melhora é $1,0$, o que corresponde ao valor máximo que pode ser apresentado pela aplicação.

Com relação a *como adaptar*, a política é a da realização de uma adaptação onde a mudança de um nível de QoS para outro, seja definida a partir do valor intermediário entre o grau de qualidade configurado e o exibido ou de visualização (erro), levando-se em consideração tanto o seu valor atual quanto os valores de n instantes anteriores, para que a adaptação ocorra através de *saltos* de qualidade, de maneira que a busca pelo ponto de estabilização ocorra de forma direta.

Finalmente, com relação a *quando iniciar o processo de adaptação*, a política de adaptação proposta neste trabalho considera não apenas o valor de um parâmetro, mas dos dois parâmetros de QoS (indiretamente representados pelo grau de qualidade) para decidir da necessidade ou não de adaptação.

Em suma, o objetivo da política de adaptação é o de maximizar o grau de qualidade através de uma adaptação *rápida*, *direta* e *orientada para a perspectiva do usuário.*

4.4.1 Mecanismo de Adaptação Local

O modelo básico do mecanismo de adaptação local proposto neste trabalho é mostrado na figura 4.2.

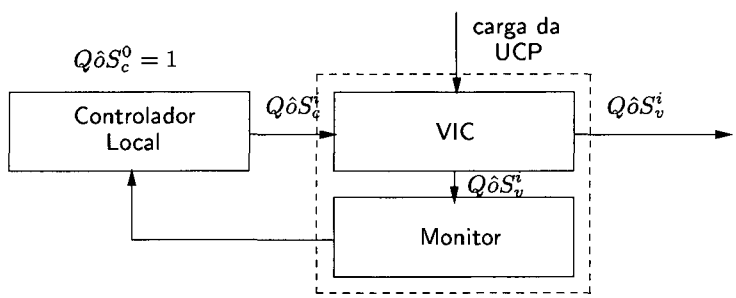

Figura 4.2: Mecanismo de adaptação local.

Pode-se perceber que esse mecanismo segue o modelo básico dos sistemas de controle realimentados. A partir da observação do grau de qualidade de exibição ou visualização $Q\hat{o}S_v^i$, o mecanismo altera o grau de qualidade de configuração $Q\hat{o}S_c^i$ (aquele associado

ao nível de QoS correntemente configurado na interface) tentando adaptá-lo ao contexto de carga corrente do processador.

O funcionamento do monitor local é o seguinte: a aplicação é configurada com um valor inicial ($Q\hat{o}S_c^0$) máximo para o o grau de qualidade, isto é, 1. Periodicamente, o controlador observa o grau de qualidade de visualização $Q\hat{o}S_v^i$. Esse grau de qualidade é então enviado para o controlador, de posse do grau de visualização, o controlador verifica se existe ou não a necessidade de adaptação através do cálculo do erro atual ε, onde

$$\varepsilon = Q\hat{o}S_c^i - Q\hat{o}S_v^i.$$

Se o erro ε é menor que um valor arbitrado de $1,0 \times 10^{-3}$ (i.e., o grau de qualidade de exibição é muito próximo ao de configuração), o mecanismo pressupõe que o processador está com folga e altera o nível de QoS de configuração para aquele com grau de qualidade superior ao corrente, somando ao grau de qualidade configurado a metade da média dos erros em n instantes anteriores; se o erro ε é positivo (i.e., o grau de qualidade de exibição é menor que o de configuração), o mecanismo pressupõe que o processador está sobrecarregado e altera o grau de qualidade para aquele com grau intermediário entre o valor configurado e o visualizado, isto é, ao valor do grau de qualidade configurado subtrai-se a metade da média do erro em n instantes anteriores. Com isso, obtem-se um valor do grau de qualidade que exija menor necessidade de processamento. A vantagem de se usar os valores do erro em um determinado número de instantes, se justifica pelo fato de se estar fazendo uma filtragem sobre as variações do grau de qualidade, fazendo que as alterações ocorram de maneira mais suave e direta.

As ações de adaptação propriamente ditas para alterar o grau de qualidade corrente, tanto no caso de melhora quanto na degradação, serão realizadas através da alteração dos valores dos parâmetros $\rho_1, \rho_2, ..., \rho_n$ que compõe o grau de qualidade corrente (por razões que serão explicadas na seção seguinte, foram considerados apenas os parâmetros freqüência de quadros e fidelidade da imagem).

4.5 Implementação

O mecanismo foi implementado para atuar sobre o sistema de vídeo-conferência VIC (versão 2.8-ucl3), executado sobre o sistema operacional Solaris 7, em uma máquina Sun

Sparc Ultra com freqüência de "clock" de 143 MHz e 64 megabytes de memória RAM. O algoritmo de compressão usado foi o M-JPEG em virtude das limitações do "frame grabber" (placa de captura) disponível que aceita apenas esse padrão. O mecanismo de adaptação é uma tarefa executada de forma concorrente (processo separado) com o VIC. A comunicação entre os dois é realizada através de "sockets". A cada intervalo de monitoramento (aqui definido como sendo o mesmo intervalo de tempo usado pelo VIC para fazer a atualização do valor da freqüência de quadros da sessão local, que é de 1 segundo), ele passa para o mecanismo de adaptação o valor da freqüência de quadros de visualização e da fidelidade da imagem, e recebe de volta os novos valores da freqüência de quadros e fidelidade da imagem com os quais ele deve reconfigurar a sessão da aplicação local.

4.5.1 Determinação da Função de Qualidade

Para a construção da função grau de qualidade e, conseqüentemente, para o processo de adaptação, foram considerados apenas dois parâmetros da camada da aplicação, a freqüência de quadros t com domínio $\Omega_t = \{0, 1, 2, ..., 29, 30\}$ e a fidelidade da imagem[3] f com um domínio $\Omega_f = \{0, 1, 2, ..., 99, 100\}$. Essa restrição deve-se a algumas limitações do VIC e outras intrínsecas ao próprio algoritmo compressão no que tange às possibilidades de configuração de QoS. Apesar da interface permitir também a definição dos valores dos parâmetros de QoS resolução e cor, a adaptação deles não é contemplada pelo mecanismo em decorrência do número restrito de valores configuráveis para esses parâmetros (três resoluções, *pequena, normal ou grande* e para a cor, *colorido ou preto-e-branco*).

Os níveis de QoS configuráveis estão contidos em uma tabela *hash*, sendo que a entrada primaria para essa tabela corresponde aos valores da fidelidade f e da freqüência de quadros t. Efetivamente o número de níveis de QoS configuráveis são representados pelo conjunto dado pelo produto cartesiano $\Omega_t \times \Omega_f$. O número total de níveis de QoS é, então, $31 \times 101 = 3131$ (o valor real é, na verdade, $31 \times 96 = 2976$, já que o VIC limita a configuração da fidelidade da imagem para o padrão M-JPEG a valores entre 0 e 95).

[3] A fidelidade da imagem aqui é um parâmetro que representa quão próximo da imagem real é a imagem digital, a faixa de valores desse parâmetro é uma variável do algoritmo de compressão M-JPEG e representa o conjunto de valores possíveis configuráveis.

Wallace [35]		Reininger [28]		
Fidelidade da Imagem (bits/pixel)	**Qualidade Subjetiva**	**Fidelidade da Imagem (bits/pixel)**	**Escala de Quantização**	**Qualidade Subjetiva**
		0.4	10	Pobre
0.50 a 0.75	Boa a muito boa, suficiente para muitas aplicações	0.5	30	Média
		0.6	50	Boa
0.75 a 1.50	Excelente, suficiente para a maior parte das aplicações	1.3	85	Muito boa
1.50 a 2.00	Usualmente indistinguível da imagem original			
		2.5	95	Excelente

Tabela 4.1: Relacionamento entre a qualidade da imagem e o número de bits/pixel (após a compressão) [35, 28].

A função $\mathcal{QoS}$ foi moldada de acordo com as premissas definidas na seção 4.2 usando a metodologia descrita na seção 4.3.

Para definição do grau de qualidade em função da freqüência de quadros t, foram escolhidos alguns valores definindo o grau máximo de 1 a uma qualidade equivalente a do sistema de TV comum, de 25 (sistema PAL) a 30 (sistema NTSC) fps e um grau de qualidade 0 para uma freqüência de quadros abaixo de 5 fps; nesse intervalo, foram definidos alguns valores representando pontos de inflexão, de maneira a moldar o comportamento da função $\mathcal{QoS}_t$ de acordo com as premissas assumidas anteriormente. De maneira análoga, foram definidos pontos representando o grau de qualidade QoS em função da fidelidade f baseados na tabela 4.1, obtida a partir dos dados levantados em [35] e [28], que mostram os níveis de qualidade típicos para intervalos de compressão usando o algoritmo JPEG. Os pontos definidos para $QoS \times t$ e $QoS \times f$ estão indicados por círculo nas figuras 4.3 e 4.4.

A partir destes pontos, as funções $\mathcal{QoS}_t(t)$ e $\mathcal{QoS}_f(f)$ são obtidas através de uma aproximação polinomial, conforme as figuras 4.3 e 4.4.

As funções são aproximadas por polinomios de 1^a a 4^a ordem. A cada aproximação

é analisado o resíduo que representa a diferença entre os valores de saída esperados e os obtidos. Um padrão de comportamento dos valores resíduais indica uma função de interpolação insatisfatória. Checando os valores apresentados nas figuras observa-se que o resíduo não apresenta um padrão de comportamento, variando bastante em termos de valores para as diversas aproximações.

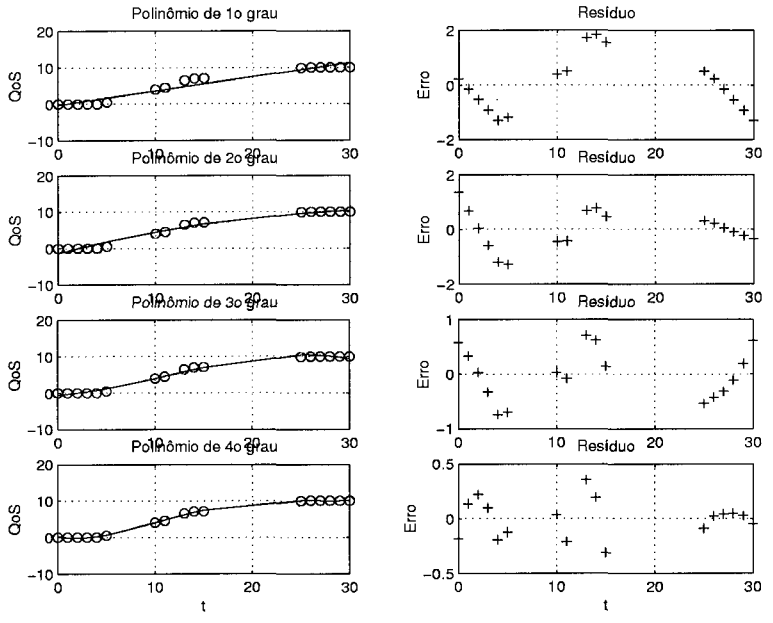

Figura 4.3: Aproximação polinomial para obtenção de $QoS \times t$ (função $\mathcal{QoS}_t(t)$).

A função final $\mathcal{QoS}(t, f)$ apresentada na figura 4.5 é obtida através da combinação das funções $\mathcal{QoS}_t(t)$ e $\mathcal{QoS}_f(f)$. Para o k-ésimo nível de QoS $L_k = <\overset{k}{t}, \overset{k}{f}>$,

$$QoS_k = min(\mathcal{QoS}_t(\overset{k}{t}), \mathcal{QoS}_f(\overset{k}{f})). \tag{4.3}$$

Como os níveis de QoS foram armazenados em uma *tabela "hash"*, a busca pelo grau de qualidade é feita de forma direta pela combinação dos valores da fidelidade f e da freqüência de quadros t. Isso nos permite otimizar a busca de forma a não se perder tempo em processamento desnecessário, o que terminaria por prejudicar o próprio desempenho do mecanismo e do VIC.

Na figura 4.5 é mostrado o comportamento da função $Q\hat{o}S = \mathcal{QoS}(freqüência\ de\ quadros,\ fidelidade\ da\ imagem)$ para uma aplicação de vídeo-conferência moldado de acordo com as premissas assumidas anteriormente ($\Omega_{freq} = \{0, 1, 2, ...30\}$ e

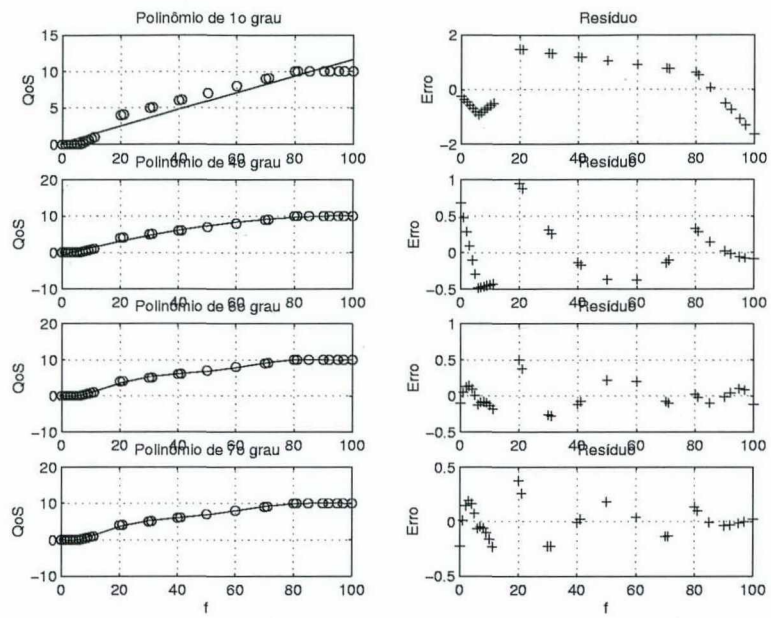

Figura 4.4: Aproximação polinomial para obtenção de $QoS \times f$ (função $\mathcal{QoS}_f(f)$).

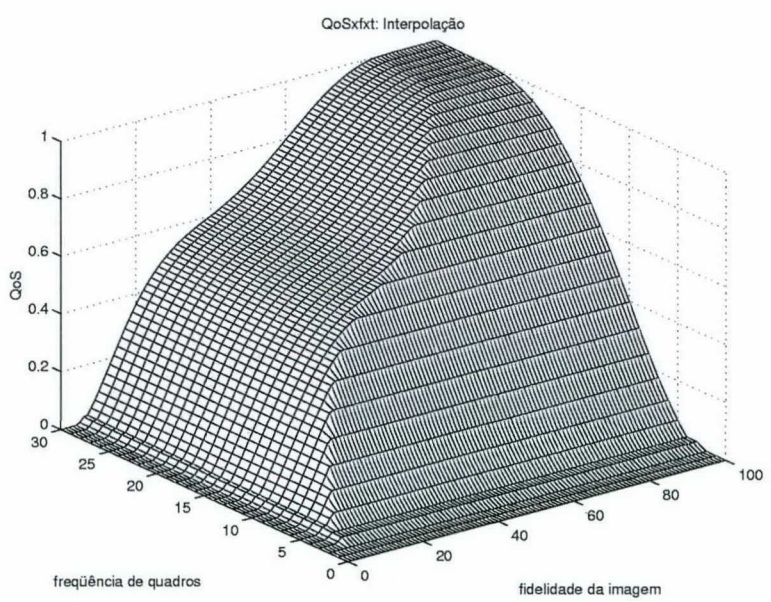

Figura 4.5: Grau de qualidade QoS em função da freqüência de quadros t e da fidelidade da imagem f ($\mathcal{QoS}(t, f)$).

$\Omega_{fid} = \{0, 1, 2, ...100\}$); foram atribuídos pesos iguais (0,5) para os dois parâmetros. A superfície foi construída utilizando-se uma metodologia baseada em técnicas de interpolação, com pontos de inflexão arbitrados a partir de analogias (qualidade de videoconferência, qualidade de TV-PAL, qualidade de TV-NTSC) e dos dados levantados em [35] e [28] que mostram os níveis de qualidade típicos para intervalos de compressão (fator de quantização), usando o algoritmo de compressão JPEG.

4.5.2 Implementação da Função de Qualidade

A cada nível de QoS L_i, foi associado o grau de qualidade QoS_{L_i} correspondente dado por $\mathcal{QoS}(\overset{i}{t}, \overset{i}{f})$. Para adequar a organização da tabela a forma de funcionamento do mecanismo, o registro contém dois campos a mais, *next* e *prev*, que fazem a ordenação do nível de QoS de maneira crescente e decrescente respectivamente. Eles funcionam como ponteiros dentro dos registros apontando do nível de QoS inferior para o imediatamente superior, no caso do campo *next*, e apontando do nível de QoS superior para o inferior no caso do campo *prev*. Para níveis de QoS com o mesmo grau de qualidade, foi estabelecida uma relação de precedência dada pelo consumo de processador e obtida de forma empírica.

O intervalo de monitoramento foi ajustado em 1 segundo. Este intervalo também reflete o tempo necessário para a realização da mudança do nível de QoS configurado (realização da adaptação).

A única pesquisa realizada na *tabela "hash"* visa determinar o grau de qualidade associado ao nível de QoS de visualização. Assim, de posse dos valores da fidelidade e freqüência de quadros visualizados, é gerada a entrada na tabela onde está armazenado o valor do nível de QoS correspondente. O armazenamento do índice do registro correspondente ao nível de QoS de configuração permite que se faça um acesso direto a ele: sendo L_{c_k} o nível de QoS de configuração corrente, onde k é seu índice na tabela, e L_{v_k} o nível de QoS de visualização corrente, a ação de adaptação é definida então de acordo com o valor do erro corrente $\varepsilon = L_{c_k} - L_{v_k}$. Caso o erro seja maior que zero o mecanismo assume que o processador está sobrecarregado, e os parâmetros de qualidade de serviço precisam ser degradados. O novo nível com que o VIC deve ser reconfigurado será L_{c_k}, definido de acordo com a seção 4.4.1, neste caso o valor de n é definido como sendo de 4 instantes. Essa solução reduziu substancialmente o "overhead" introduzido

pelo mecanismo de adaptação. Assim, o novo valor para o nível de QoS de configuração determinado pelo mecanismo de adaptação, exigi menos processamento que o nível de QoS de configuração corrente.

4.6 Resultados

Para a avaliação de desempenho do mecanismo, foi realizado um teste de execução do VIC sem e com a presença do mecanismo de controle de QoS local, com o intuito de estabelecer um comparativo de desempenho nos dois casos.

Durante a aplicação do teste foi usado o mesmo contexto em ambos os casos: a carga foi simulada pelo uso de um programa de CAD **Scilab** que atua como processo concorrente no acesso ao processador, e é disparado após um certo intervalo de tempo. Considerou-se para poder analisar o mecanismo de adaptação, situações similares nos dois casos: a câmera filmando sempre um ponto fixo sem movimentação em volta e a fidelidade e a freqüência de quadros configurada inicialmente com seus valores máximos (95 e 30 respectivamente). O uso do **Scilab** como processo concorrente que por si só já sobrecarrega a UCP da máquina, permitiu fazer os testes na situação de limite mínimo do grau de qualidade sobre o qual o mecanismo atua.

Um dos parâmetros que compõe o grau de qualidade, a freqüência de quadros visualizada, tem seu comportamento mostrado na figura 4.6 no caso sem o mecanismo de adaptação e na figura 4.7 com o mecanismo de adaptação. A partir do disparo da ferramenta **SciLab** o valor da freqüência é reduzido drasticamente estabilizando em uma média de 10 fps, no caso sem o mecanismo. No caso com o mecanismo de adaptação a freqüência de quadros estabiliza em um valor de 11 fps, isto é, houve um ganho de 1 fps a favor do mecanismo de adaptação.

O outro parâmetro, a fidelidade da imagem, sofre degradação no caso da sobrecarga quando a ferramenta **SciLab** é disparada, no caso com o mecanismo de adaptação, enquanto fica constante no caso sem o mecanismo de adaptação.

O desempenho do grau de qualidade sem o mecanismo de adaptação é representado na figura 4.8, observa-se que a partir do momento em que a UCP é sobrecarregada o grau de qualidade é reduzido estabilizando em um valor de $0,518$. Na figura 4.9 é mostrado o comportamento do grau de qualidade com o mecanismo de adaptação,

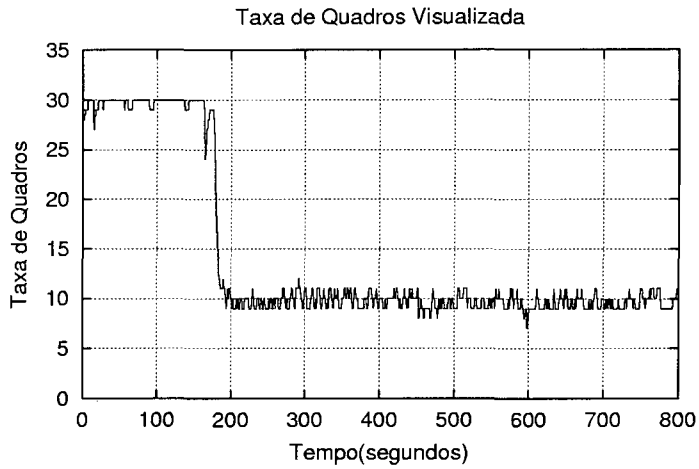

Figura 4.6: Comportamento da freqüência de quadros visualizada sem a presença do mecanismo de adaptação.

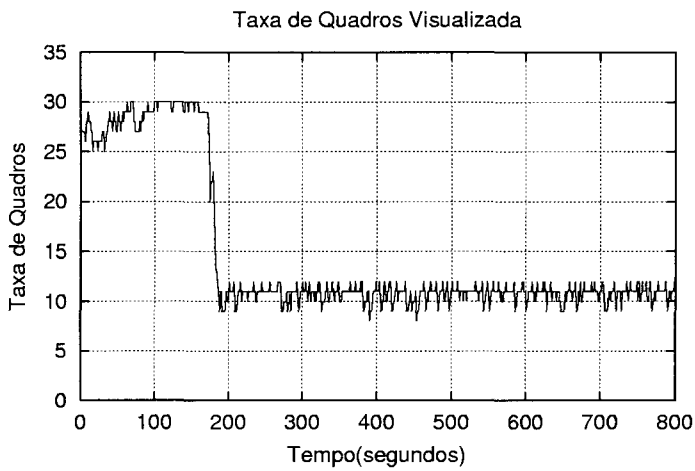

Figura 4.7: Comportamento da freqüência de quadros visualizada com a presença do mecanismo de adaptação.

sendo que seu valor durante todo o teste estabiliza em valor em torno de 0,568 o que representa um ganho de 0,05 a favor do mecanismo. Este valor de ganho se justifica devido a melhora de 1 fps na freqüência de quadros com o mecanismo, apesar da fidelidade da imagem ter sido reduzida com o mecanismo, daí conclui-se que a função grau de qualidade acaba por garantir que o usuário possa obter uma melhor satisfação de acordo com o contexto local corrente.

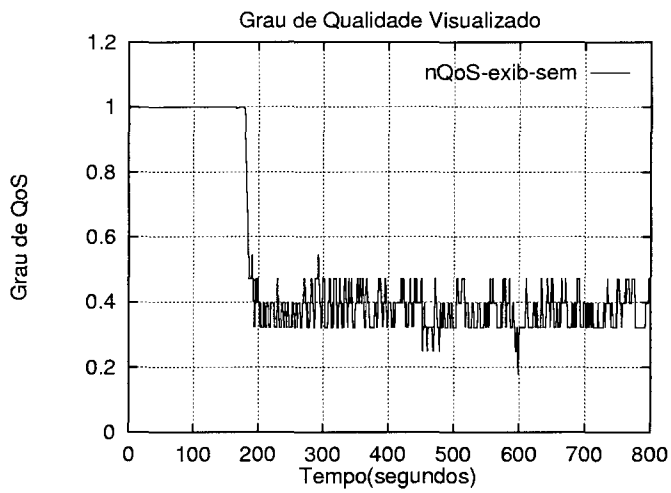

Figura 4.8: Comportamento do grau de qualidade visualizado sem a presença do mecanismo de adaptação.

4.7 Discussão

A implementação do mecanismo de adaptação demonstrou sua eficiência ao melhorar grau de qualidade fornecido pela ferramenta de vídeo-conferência alvo (VIC). Apesar da fidelidade da imagem ter sido reduzida bastante, o ganho na freqüência de quadros compensa esta perda, fazendo com que o grau de qualidade fornecido ao usuário seja o maior no contexto corrente. Isto se deve ao fato de que a adaptação é feita baseada na função grau de qualidade, que associa um valor para cada combinação de valores dos parâmetros sendo configurados.

Apesar da melhora demonstrada, só em alguns momentos o VIC conseguiu alcançar o nível de QoS configurado pelo mecanismo de adaptação, não conseguindo manter esse valor durante o período de sobrecarga do processador. Esse problema deve-se em grande

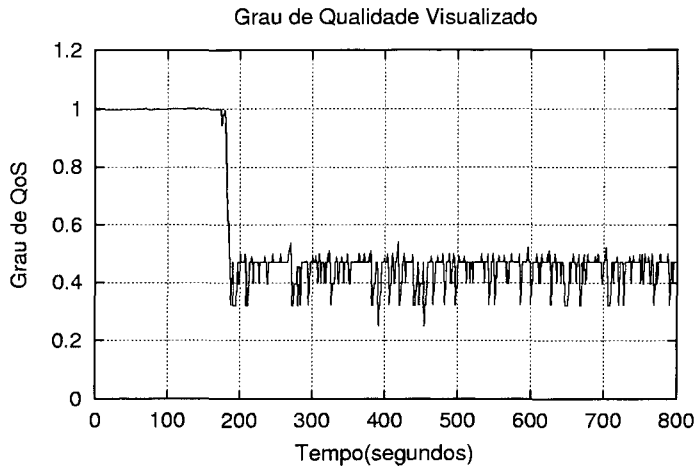

Figura 4.9: Comportamento do grau de qualidade visualizado com a presença do mecanismo de adaptação.

parte, à própria forma de implementação da aplicação, na qual toda comunicação entre os módulos é intermediada pela interface (interpretada) em Tcl/Tk, o que torna as ações do mecanismo mais lentas. Desde que as ações de adaptação alteram os valores da fidelidade da imagem e da freqüência de quadros na interface, existe um certo atraso até que as alterações causem efeitos. Com isso o peso da interface compromete a eficiência do mecanismo, sendo inadequado quando as oscilações são bruscas e de curta duração, já que existe este atraso da aplicação em responder de acordo com o novos valores configurados. Se o mecanismo de adaptação mantivesse acesso direto às variáveis usadas pelo algoritmo de compressão (M-JPEG) para definir os valores dos parâmetros de QoS, a diferença entre os valores configurados e visualizados tenderiam a diminuir e se anular.

Adotando-se como procedimento de suavização, o mecanismo leva em consideração os valores do erro nos quatro instantes anteriores. Isso nos permite realizar uma ação de adaptação mais suave, fazendo com que a alteração dos parâmetros ocorra de maneira mais direta, reduzindo o intervalo de tempo até que o mecanismo estabilize o grau de qualidade, no maior valor possível de acordo com o contexto do processador.

Na literatura existem trabalhos descrevendo modelos para adaptação de QoS em sistemas finais, que são baseados de forma explícita ou implícita em sistemas de controle. Neles a ação de adaptação atua sobre um único parâmetro de QoS o que está

bastante distante da perspectiva do usuário, adotada neste trabalho.

Em [25] são descritos experimentos envolvendo a transmissão de vídeos no formato M-JPEG onde a freqüência de transmissão do emissor é superior à capacidade de processamento do receptor, o que conduz a uma degradação da imagem. Nos testes, o emissor possui uma placa dedicada a compressão de vídeo no formato M-JPEG que permite transmitir em uma freqüência de até 30 fps. O receptor, por outro lado, armazena os quadros em um "buffer" e realiza a descompressão via "software" não conseguindo assim, acompanhar a freqüência da emissão. A adaptação é feita no emissor através da redução da freqüência de quadros a partir de mensagens enviados pelo receptor informando a taxa de ocupação do "buffer". Nesta estratégia somente um parâmetro relacionado à qualidade da imagem é alvo da aplicação.

4.8 Conclusões

O mecanismo de adaptação local apresentou uma pequena melhora na qualidade apresentada, na situação em que o processador se encontra bastante sobrecarregado, isto porque, as ações de adaptação ocorrem no limite máximo de degradação do grau de qualidade. Melhora semelhante se encontrou também em termos de freqüência de quadros. Uma melhora na determinação da função grau de qualidade de forma que ela possa contemplar outros parâmetros de QoS como cor, tamanho da janela, etc., nos permitiria obter resultados ainda mais condizentes com a realidade do usuário.

O uso do Tcl/Tk sobretudo, tornou a implementação do mecanismo de adaptação bastante crítica, no sentido de que qualquer processamento adicional é prejudicial ao desempenho da aplicação de vídeo-conferência como um todo. Assim as funções implementadas no mecanismo, de busca de valores, definição dos novos parâmetros a serem configurados, etc., devem executar serviços de maneira mais otimizada possível sem que haja prejuízo para o desempenho da aplicação. O ideal seria que a aplicação fosse totalmente integrada com o mecanismo de adaptação, acessando de maneira direta os diversos parâmetros de QoS da aplicação, com isso, seria possível se obter um melhor desempenho nas diversas condições de sobrecarga de processador.

Destaca-se ainda como dificuldade a definição de uma função grau de qualidade, que comtemple diversos parâmetros de QoS, de forma que satisfaça o desejo dos usuários.

Capítulo 5

Mecanismo de Adaptação Global da Qualidade de Serviço

5.1 Introdução

O congestionamento de rede é outro fator a ser considerado quando se trata de fornecer os melhores serviços aos usuários. Alguns dos parâmetros de QoS, particularmente a freqüência de quadros de vídeo, podem sofrer variações bruscas e descontroladas em seus valores durante a transferência de dados em virtude de congestionamentos de rede, o que causa perdas de unidades de transporte. Essa degradação brusca e descontrolada na QoS em face de mudanças na carga da rede diminui bastante o grau de satisfação do usuário. Em ambientes totalmente melhor-esforço, como a Internet, onde não há a possibilidade de reserva de recursos, tais oscilações podem degradar essa qualidade até um nível abaixo do mínimo tolerável pelo usuário. Assim, surge a necessidade de mecanismos que realizem uma adaptação de QoS, de forma suave e controlada, degradando ou melhorando essa qualidade frente ao contexto corrente da rede.

Neste capítulo é definida a implementação de um mecanismo de adaptação de QoS de escopo global, que atua sobre o VIC em caso de congestionamento de rede, em ambientes melhor-esforço (i.e. sem reserva de recursos). São implementadas aqui duas variantes do mecanismo, uma usando as informações de controle sendo transmitidas através de "sockets" entre os sistemas finais receptores e o sistema final emissor, e outro

usando as informações contidas nos pacotes RTCP, especificamente aquelas contidas nos pacotes de relatório de receptor. Ambos os mecanismos são baseados na função grau de qualidade definida na seção 4.2, o que permite ao mecanismo maximizar a qualidade oferecida como um todo, e não através de parâmetros individuais com em outros modelos. O modelo de comunicação aqui definido é do tipo 1:N, ou seja, 1 emissor e N receptores.

5.2 Mecanismo de Adaptação

O mecanismo de adaptação implementado, adota uma política que procura manter o maior grau de qualidade considerando o contexto corrente da rede. A figura 5.1 mostra a arquitetura geral do mecanismo global.

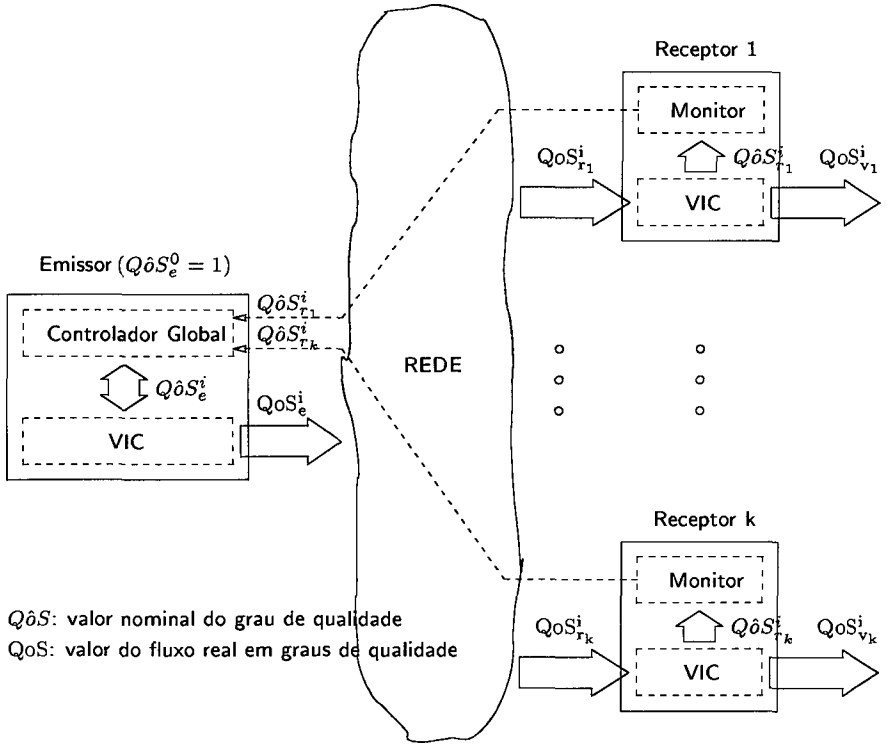

Figura 5.1: Mecanismo de adaptação global geral.

O mecanismo global geral é um sistema realimentado. Ele é formado no sistema final emissor pela aplicação (VIC) e pelo *Controlador Global*, que é o responsável pelas ações da adaptação propriamente ditas, atuando sobre a aplicação configurando-a com

os novos valores ($Q\hat{o}S_e^i$). No instante inicial da sessão, a aplicação é configurada com o valor inicial ($Q\hat{o}S_e^0$) máximo, que corresponde ao maior valor possível para o grau de qualidade, isto é, 1. No sistema final receptor ele é formado pela aplicação e o *Monitor* que observa o comportamento dos parâmetros de recepção ($Q\hat{o}S_{r_k}^i$), e realimenta com essas informações o *Controlador Global* de forma que as ações de adaptação possam ser tomadas.

Duas situações serão estudadas: "sockets" e RTCP:

- O mecanismo de adaptação global no caso de uso de "sockets", utiliza os valores dos parâmetros de QoS dos sistemas finais receptores, que expressam o grau de qualidade de recepção como informação de realimentação, isto é, $Q\hat{o}S_{r_k}^i$. O controle é feito a parte, sendo que a mudança de nível de QoS é definida a partir do valor intermediário entre o grau de qualidade de emissão e o de recepção (erro - ε). Para tal, leva-se em consideração os valores do erro nos 4 instantes anteriores.

- No caso do uso do RTCP, no receptor o mecanismo faz o monitoramento do desempenho dos parâmetros de QoS $Q\hat{o}S_{r_k}^i$, isto é, $Loss_k^i$. No emissor as informações contidas no campo de *perda percentual* dos pacotes de relatório de receptor (RR), são usadas pelo *Controlador Global* para as decisões de adaptação. Aqui para o mecanismo de adaptação global são definidas três faixas que representam a carga de rede, sendo que cada faixa corresponde a uma ação de adaptação (degrada, mantém ou melhora), o intervalo de tempo entre as ações de adaptação aqui depende da freqüência de chegada dos pacotes RR.

Em ambos os casos a ação de adaptação visa maximizar o grau de qualidade através de adaptação *orientada para a perspectiva do usuário final.*

5.2.1 Mecanismo de Adaptação Global

O modelo de adaptação proposto realiza uma adaptação fim-a-fim centrada na perspectiva dos usuários finais, para o qual a qualidade é vista como um todo e não através de parâmetros individuais da camada de aplicação em um sistema sem reservas de recursos (melhor-esforço). O modelo de adaptação é dirigido para aplicações do tipo 1:N, como por exemplo, vídeo-conferência para ensino à distância. É assumido que a UCP é dedicada à execução do VIC e das tarefas de adaptação.

As duas variações do mecanismo de adaptação implementadas objetivam maximizar o grau de qualidade de visualização das N sessões de usuário dentro do permitido pelo contexto corrente da rede.

Usando "Sockets"

O fluxo de vídeo enviado pelo emissor sai com *grau de qualidade de emissão* $\mathrm{QoS}_{\mathrm{e}}^{\mathrm{i}}$ e chega ao k-ésimo receptor com *grau de qualidade de recepção* $\mathrm{QoS}_{\mathrm{r_k}}^{\mathrm{i}}$, tal que $Q\hat{o}S_{r_k}^i \leq Q\hat{o}S_e^i$ $(k = 1, 2, ..., N)$ em termos nominais do valor do fluxo, já que quadros podem ter sido perdidos em virtude da perda de unidades de transporte na rede (nos receptores, os quadros são processados para gerar um fluxo com *grau de qualidade de visualização* $\mathrm{QoS}_{\mathrm{v_k}}^{\mathrm{i}}$, tal que $Q\hat{o}S_{v_k}^i \leq Q\hat{o}S_{r_k}^i$ em termos nominais, em virtude de mais perdas de quadros decorrentes de perdas de "deadlines").

A sessão em seu início possui uma condição inicial onde o *grau de qualidade de emissão* $(Q\hat{o}S_e^i)$ é configurado aqui para seu valor máximo 1. O controlador tenta maximizar o grau de recepção dos receptores, de maneira que ele fique o mais próximo possível do valor do grau de emissão configurado no instante anterior durante toda a sessão.

Durante cada intervalo de monitoramento i, cada receptor envia para o mecanismo de adaptação o grau de qualidade de recepção $Q\hat{o}S_{r_k}^i$. De posse desses valores o mecanismo calcula por exemplo, o valor médio (média aritmética) do grau de recepção pra este intervalo, $\overline{Q\hat{o}S_r^i}$, com isso o controlador verifica então se existe ou não a necessidade de adaptação através do cálculo do erro atual, onde $\varepsilon = Q\hat{o}S_e^i - \overline{Q\hat{o}S_r^i}$.

Se o erro ε for menor que um valor pequeno fixado neste trabalho a $1,0 \times 10^-3$, o mecanismo pressupõe que a rede não está congestionada e altera o grau de QoS de emissão para aquele com grau de qualidade superior ao corrente, somando ao grau de qualidade de emissão, a metade da média dos erros nos 4 instantes anteriores; se o erro ε é positivo, o mecanismo pressupõe que a rede está congestionada e altera o grau de qualidade de emissão, para aquele com valor intermediário entre o valor grau de emissão e o de recepção. Com isso obtem-se o novo valor do grau de qualidade de emissão $Q\hat{o}S_e^{i+1}$ a ser enviado no próximo instante pelo VIC.

A figura 5.2 mostra a representação do mecanismo de adaptação usando "sockets".

As ações de adaptação para alterar o grau de qualidade corrente tanto no caso

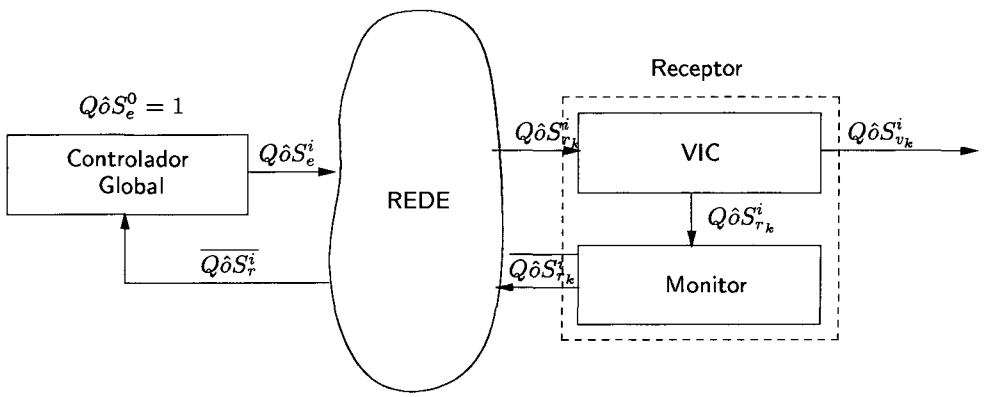

Figura 5.2: Mecanismo de adaptação global usando "sockets".

de melhora quanto na degradação, são realizadas através da alteração dos valores dos parâmetros fidelidade da imagem e freqüência de quadros que compõe o grau de qualidade corrente na interface do VIC.

Usando pacotes RR

O mecanismo de adaptação faz o monitoramento dos pacotes RR enviados pelos sistemas finais receptores a partir da própria aplicação. Aqui por uma facilidade maior para monitoração dos pacotes, o mecanismo é totalmente integrado sendo implementado dentro da aplicação. Para cada sistema, os pacotes são então desmontados e o valor da perda percentual $Loss_k^i$ (sendo $k = 1, 2, ..., N$) de pacotes é lido. Com esses valores, o mecanismo calcula o valor médio da perda de pacotes $\overline{Loss^i}$ entre todos os participantes. De acordo com o valor médio da perda são definidas três faixas nas quais as ações de adaptação são realizadas: na primeira faixa definida entre 0 e 5%, o mecanismo assume que a rede não está congestionada e atua melhorando o valor do grau de qualidade de emissão, acrescentado ao grau de qualidade emissão atual 10% do seu valor; na segunda faixa, definida entre 5 e 10%, a rede está em seu estado normal e o valor do grau de qualidade de emissão é mantido; e na terceira faixa definida entre 10 e 100%, é assumido que a rede está congestionada e o mecanismo atua degradando o valor do grau de qualidade emissão subtraindo 10% do seu valor atual. Os valores percentuais das faixas são definidas tendo por base o trabalho desenvolvido em [5], onde são definidas também faixas que representam a carga da rede. Os valores aqui

apresentados foram os que nos permitiram obter os melhores resultados práticos.

A figura 5.3 mostra a representação do mecanismo de adaptação usando os pacotes RR do RTCP.

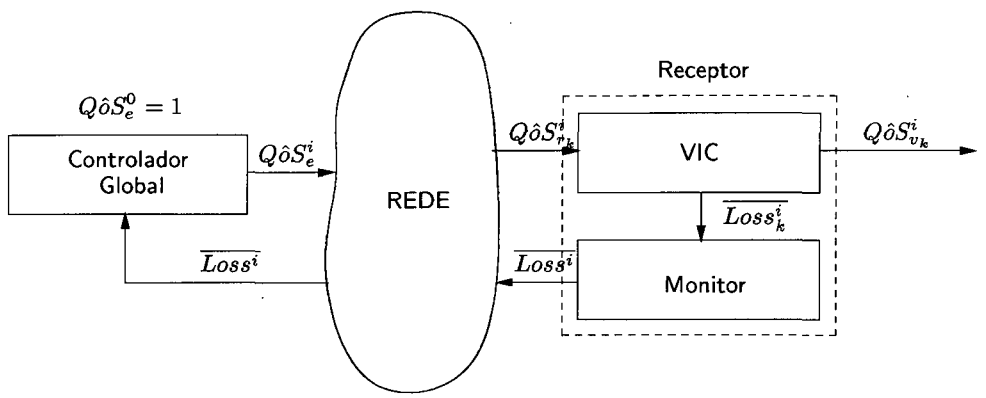

Figura 5.3: Mecanismo de adaptação global usando os pacotes RR do RTCP.

Esse modelo de controle foi escolhido por ser de fácil implementação, e porque ele usa um esquema muito conhecido usado em [16], que também foi aplicado com sucesso no contexto da aplicação de controle desenvolvida por Bolot et al. [4].

5.3 Implementação

Ambos os mecanismos foram implementados para atuar sobre o sistema de vídeo-conferência VIC, executado sobre o sistema operacional Solaris 7, em duas máquinas Sun Sparc Ultra 1 (emissor e receptor) com freqüência de "clock" de 143 MHz e 64 megabytes de memória RAM. O algoritmo de compressão usado foi o M-JPEG por motivos já descritos anteriormente (seção 4.5) e a interface de rede utilizada foi "Ethernet" com capacidade de 10 Mbps.

Para geração do fluxo de vídeo no VIC foi usada uma placa de codificação M-JPEG; o processo de descompressão, por outro lado, foi inteiramente executado por um CODEC implementado via "software". Para avaliação do desempenho da implementação, foram realizados testes de execução do VIC sem e com a presença do controlador. Em ambas situações foi usado um contexto similar: durante os 150 segundos iniciais, a rede estava com carga mínima (apenas o VIC transmitindo vídeo); nos

250 segundos finais, o programa gerador de tráfego Netperf [26] aumentou o tráfego na rede para cerca de 30% de sua largura de banda. A câmera filmava sempre um ponto fixo sem movimentação em volta e a fidelidade da imagem e a freqüência de quadros, os parâmetros de QoS configuráveis pela interface do VIC foram inicializados com seus valores máximos, 95 e 30 respectivamente (grau de qualidade de referência).

Usando "Sockets"

Aqui se optou pela transmissão das informações via "sockets" usando o protocolo TCP, devido unicamente à facilidade de implementação desse mecanismo de comunicação. Entretanto, nesta opção deve-se levar em conta suas limitações, devido a um "overhead" que pode aumentar as perdas de "deadline" da aplicação. O monitoramento dos dados é feito a partir da leitura de a cada intervalo de monitoramento (1 segundo), dos valores dos parâmetros de QoS. Esses valores são trasmitidos para o mecanismo de adaptação através de um "socket", que é estabelecido entre cada sistema final receptor e o mecanismo de adaptação. O mecanismo recebendo esses valores de todos os participantes, calcula a média do grau de qualidade de recepção e compara com o valor do grau de qualidade enviado, para que as devidas ações de adaptação (degradação ou melhora da qualidade) sejam tomadas.

Usando os Pacotes RR

Aqui se optou pela transmissão das informações de realimentação via pacotes RR do RTCP, por eles serem enviados periodicamente durante toda a duração da sessão, contendo informações de desempenho de todos os participantes. A informação que nos permite fazer um controle sobre a qualidade de apresentação é o percentual de perda dos pacotes de dados.

O monitoramento que é feito no emissor analisa a taxa de perda percentual de pacotes de todos os receptores. Esta análise é usada para estimar o estado da rede, ou seja, se está congestionada ou não. As ações de adaptação são tomadas então de acordo com o estato atual da rede, fazendo com que o grau de qualidade enviado seja o maior possível dentro desse estado. A mudança da configuração dos parâmetros de QoS faz com que a taxa de transmissão realizada pela aplicação seja ajustada de acordo com novos valores. O ajuste do grau de qualidade é feito de tal maneira que, o novo valor

a ser enviado seja obtido subtraindo ou adicionando 10% ao valor atual. Este valor foi definido a partir de teste práticos, conciliando tanto a degradação quanto a melhora do grau de qualidade.

5.4 Resultados

A seguir são a presentados os resultados obtidos para testes feitos com o mecanismo de adaptacão usando "sockets" e os pacotes RTCP.

Usando "Sockets"

Nas figuras 5.4 e figura 5.5 são mostrados os comportamentos da taxa de transmissão e percentual de perdas de pacotes, durante a sessão da aplicação sem adaptação de QoS e com adaptação de QoS. A perda de pacotes é um parâmetro monitorado para que se possa estabelecer um comparativo de desempenho da aplicação com e sem o mecanismo. Os valores são monitorados a partir de cada aplicação e transmitidos para o mecanismo. Esses valores são usados para conhecer o percentual de redução da largura de banda que a aplicação está injetando na rede.

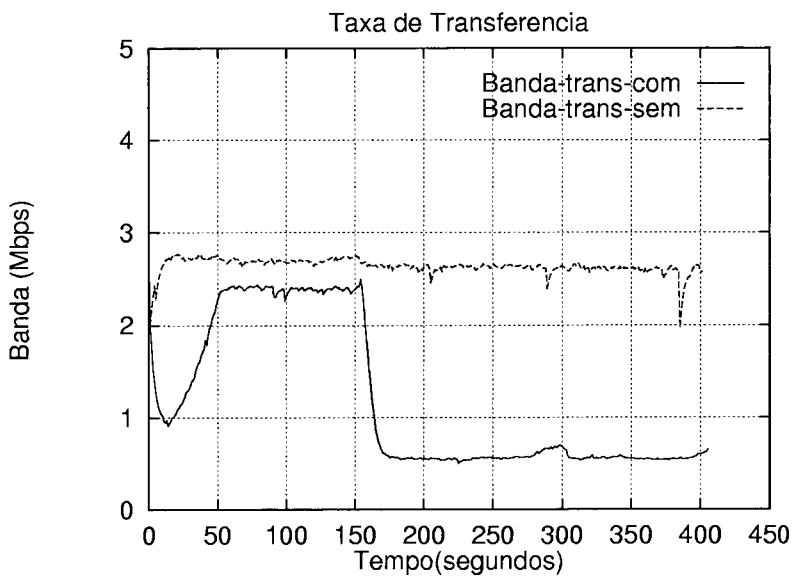

Figura 5.4: Taxa de transmissão sem e com controle de QoS.

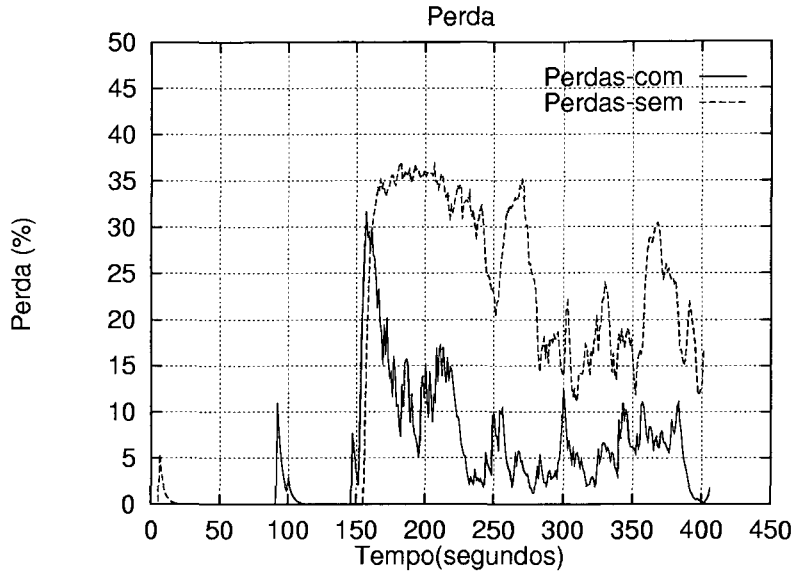

Figura 5.5: Taxa de perdas de pacotes sem e com controle de QoS.

Com a aplicação sem o controlador, a taxa de transmissão não varia muito, ficando sempre em torno de 2,61 Mbps e a taxa de perdas de pacotes média durante o período em que a rede está carregada é de 24,7%. Na execução com o controlador, a taxa de transmissão foi reduzida a partir do início da carga ficando com um valor médio de 0,65 Mbps e a taxa de perdas de pacotes média ficou em apenas 7,8%. Dos resultados anteriores, percebe-se a redução da taxa de perdas quando o mecanismo de adaptação atua. Esta redução mostra a vantagem do uso do mecanismo, pois gera-se menos tráfego na rede, ao mesmo tempo que tenta garantir o maior grau de qualidade para o usuário frente a carga da rede.

Na figura 5.6 é mostrado o comportamento do grau de qualidade de visualização sem o mecanismo de adaptação. A média do grau de qualidade de visualização obtida durante o período de carga da rede foi de 0,57, apresentando, contudo, grandes oscilações. O grau de qualidade configurado (i.e. de referência) foi sempre 1, já que os parâmetros de QoS são configurados para seus valores máximos. A diferença entre o grau de qualidade de referência e de visualização *decorre basicamente das perdas ocorridas na rede*, indicando que boa parte da informação enviada perdeu-se nos nós intermediários ou receptores.

Na figura 5.7 é mostrado o comportamento do grau de qualidade de emissão e

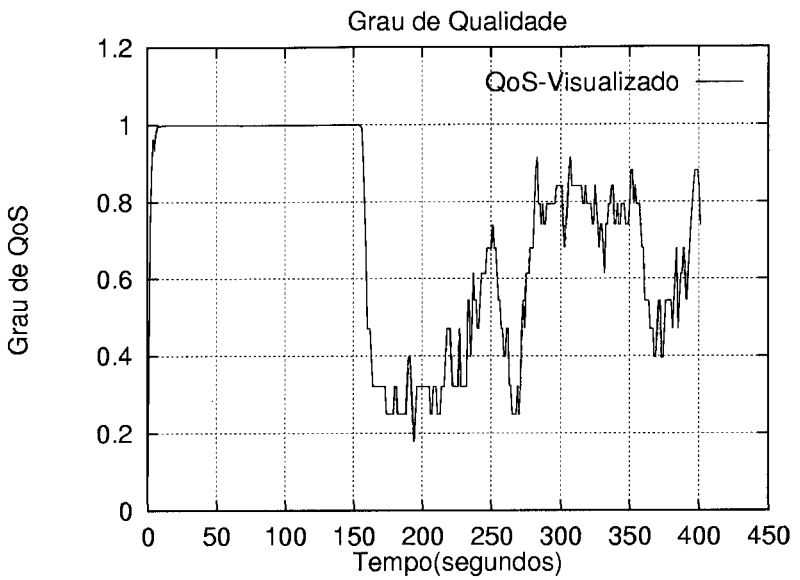

Figura 5.6: Grau de qualidade de recepção sem controle de QoS.

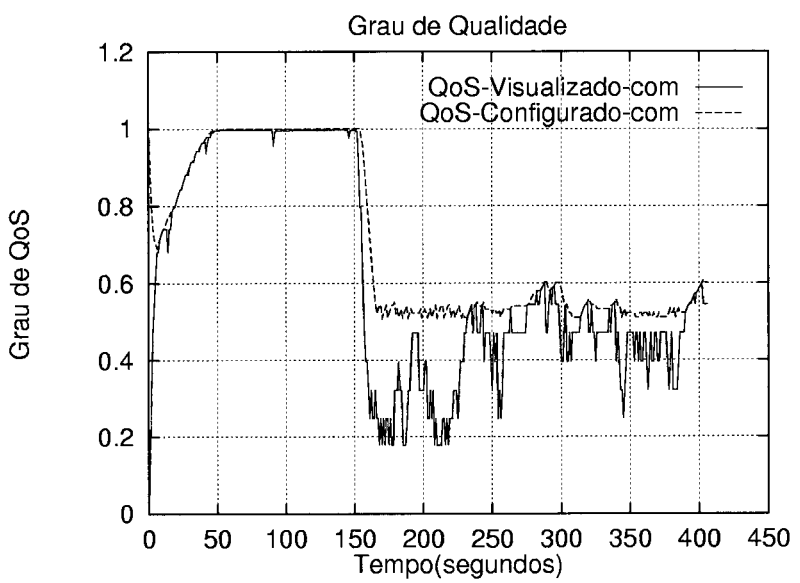

Figura 5.7: Grau de qualidade de emissão e de recepção com controle de QoS.

recepção com o mecanismo de adaptação. A média do grau de qualidade de recepção durante o período de carga da rede foi de 0,43, enquanto o grau de qualidade de emissão ficou em 0,55. Nesse caso, a diferença entre os dois foi bem menor do que no caso anterior, indicando que o mecanismo atua no sentido de evitar o desperdício de banda passante. Observa-se também que o mecanismo tenta manter o grau de qualidade de emissão em um patamar mais estável possível, o que é vantajoso para o usuário, já que oscilações bruscas na QoS acabam sendo um fator desagradável.

A diferença entre os graus de qualidades de emissão e de recepção decorreu mais de algumas dificuldades intrínsecas ao VIC do que de perdas ocorridas na rede: no VIC, a atualização dos parâmetros é feita somente a cada segundo, o que obrigou o mecanismo a trabalhar com um período de monitoramento relativamente longo e, conseqüentemente, com informações defasadas. Por outro lado, esse período de monitoramento longo foi insuficiente para que a aplicação passasse a gerar o fluxo com os valores dos parâmetros de QoS configurados pelo mecanismo, já que no VIC o controle do fluxo de dados assim como a interface é implementada usando Tcl/Tk. Esta situação poderia ser melhorada caso fosse possível o acesso direto às variáveis do CODEC usadas para definir os valores dos parâmetros de QoS. As dificuldades supramencionadas, impediram que o mecanismo permitisse que o grau de qualidade de recepção alcançasse o grau de qualidade de emissão (a estabilidade deste último, mostrada no gráfico da figura 5.7, comprova, entretanto, que o mecanismo alcançou um grau de qualidade compatível com o estado da rede).

Usando os pacotes RR

Na figura 5.8 é mostrado o comportamento da taxa de transmissão sem e com o mecanismo de adaptação de QoS. Durante a execução sem o mecanismo de QoS a taxa de transmissão varia em torno do ponto médio de 2,9 Mbps, e a taxa de recepção fica em torno de 2,4 Mbps devido às perdas de pacotes durante a sessão. Com o controle de QoS, a taxa de transmissão fica em torno de 2,1 Mbps e é reduzida bastante quando o gerador de tráfego (Netperf) é disparado (150 segundos após o início do teste), sendo que a taxa de recepção fica em torno de 1,9 Mbps. Perdas essas que são melhores representadas no gráfico da figura 5.9 que expressa o comportamento das perdas de pacote durante a sessão sem e com o mecanismo de adaptação. Os valores aqui apresentados são os

que a própria aplicação estima e que são enviados nos pacotes RR do RTCP. A média de perda sem o mecanismo de adaptação fica em torno de 13% e com o mecanismo fica em torno de 4,3%. Enquanto a taxa de transmissão foi reduzida em 72% a perda foi reduzida em 33%, o que mostra que apesar da redução da quantidade de pacotes injetados na rede ter sido reduzida, a perda de pacotes foi reduzida proporcionalmente mais ainda.

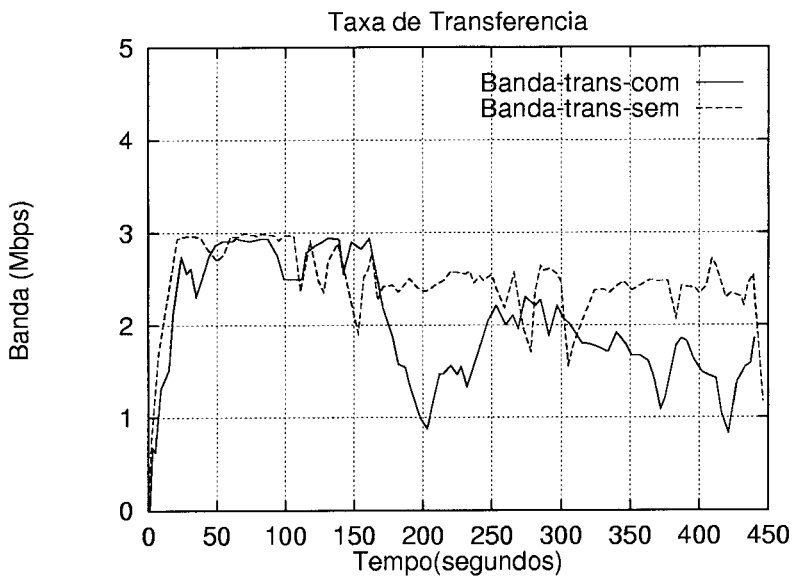

Figura 5.8: Taxa de transmissão sem e com controle de QoS.

Na figura 5.10 é mostrado o comportamento do grau de qualidade de recepção sem e com o mecanismo de controle de QoS. O valor médio do grau de qualidade sem o mecanismo de controle fica em torno de 0,88 e em 0,80 com o mecanismo. No entanto, a média da taxa de quadros recebidos no primeiro caso é de 20 fps e para o segundo de 23 fps (figura 5.11). Esses resultados ainda podem ser melhorados aumentando o peso da freqüência de quadros na função $\mathcal{QoS}$ (como já dito, o peso para a freqüência de quadros é o mesmo para a fidelidade da imagem, 0,5). Somando-se a isso, pode-se notar que a diferença entre a taxa de perda de pacotes sem e com controle de QoS, é muito maior do que a diferença entre os graus de qualidade de recepção. Isto significa que a economia na largura de banda excede a perda do grau de qualidade.

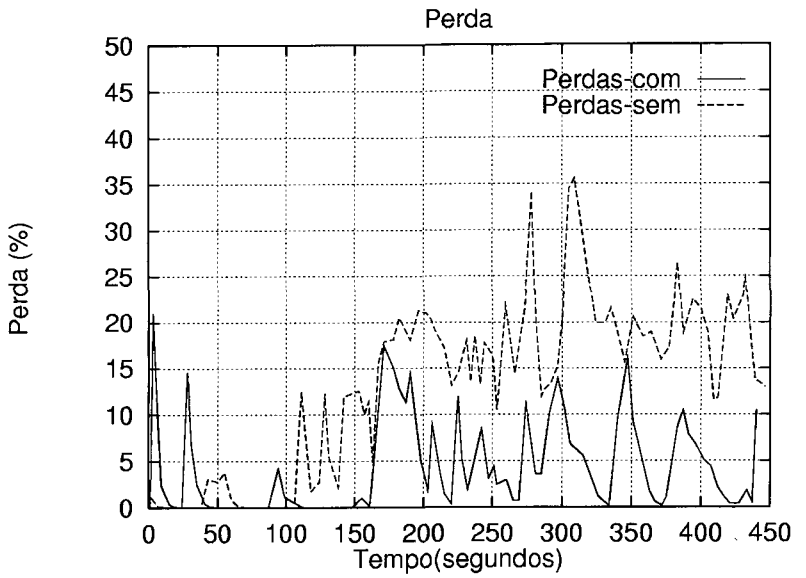

Figura 5.9: Taxa de perdas de pacotes sem e com controle de QoS.

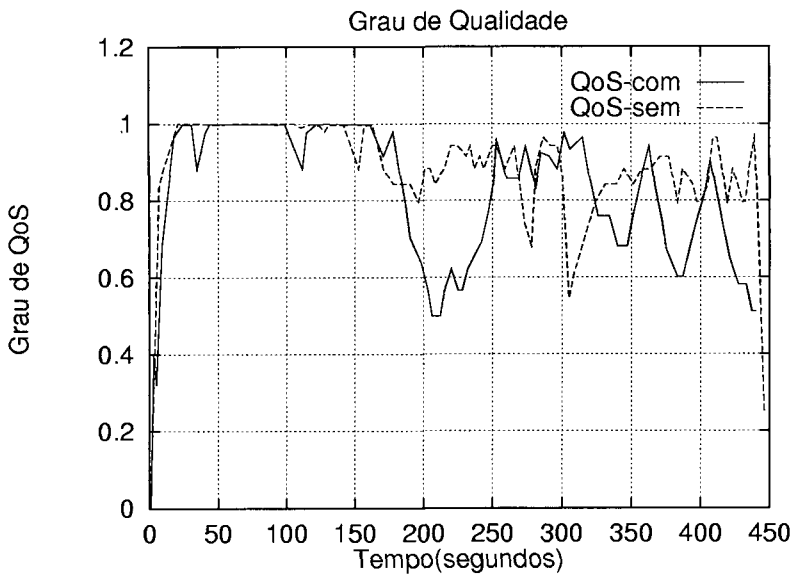

Figura 5.10: Grau de qualidade de recepção sem e com controle de QoS.

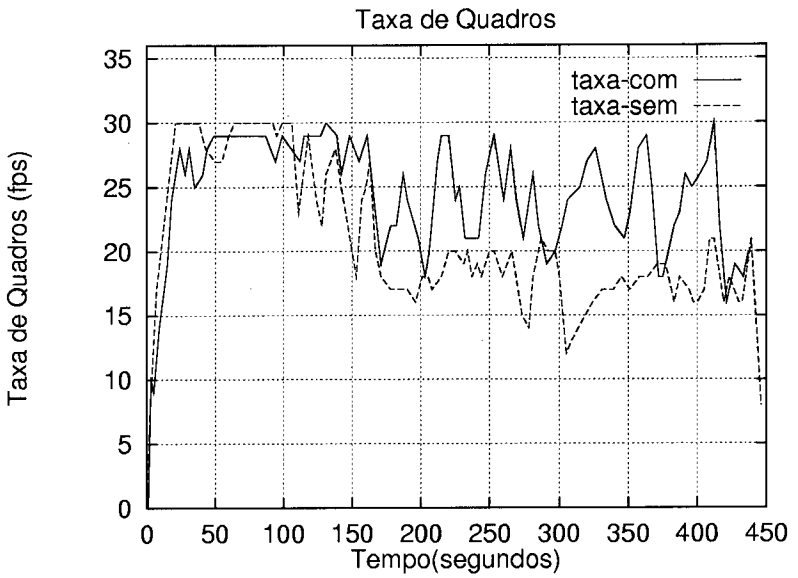

Figura 5.11: Taxa de quadros sem e com controle de QoS.

5.5 Discussão

O mecanismo de adaptação da QoS de escopo global aqui apresentado, segue uma política de adaptação centrada no usuário final, porque utiliza a função grau de qualidade que abrange parâmetros de camada de aplicação, mais especificamente a freqüência de quadros e a fidelidade da imagem, únicos parâmetros considerados no caso do VIC. O percentual de perda de pacotes e a taxa de transmissão são variáveis importantes de serem observadas, pois elas nos fornecem informações sobre o tráfego de dados na rede para que se possa estabelecer comparações de desempenho entre os mecanismos apresentados e a aplicação somente. De forma geral os resultados se revelaram bastante promissores, principalmente quando se trabalha com os dados de monitoramento fornecidos pelo protocolo RTCP. As faixas de atuação para a carga da rede definidas basearam-se em testes práticos, sendo usadas as que durante os testes obtiveram melhor desempenho, o que não exclui a possibilidade de que se possa definir outras faixas que venham ainda gerar melhores resultados.

O mecanismo de adaptação usando "sockets" apresentou poucas vantagens em levantar o grau de qualidade de recepção para os usuários. Em parte, isto se deve ao fato de que o mecanismo de comunicação - "sockets" usando o protocolo TCP exige o esta-

belecimento de uma comunicação orientada a conexão, o que implica em uma série de sinais de sinalização entre os pontos finais, (emissor e receptor) de modo que se possa estabelecer um canal de comunicação entre esses pontos até que os dados possam ser transmitidos. Isto, acaba por causar um certo "overhead" que prejudica o desempenho do mecanismo e também do VIC. O fato da interface e do controle do fluxo de dados ser implementada no VIC usando uma linguagem interpretada Tcl/Tk, acaba também influenciando no desempenho já que a ações de adaptação atuam na interface; neste caso, se ocorrem oscilações bruscas nos valores a serem configurados, existe uma certa demora da aplicação a responder de acordo com os novos valores. Levar em consideração os valores de erro em um certo número de instantes, nos permite uma ação de adaptação mais suave, conseguindo um menor tempo até que o mecanismo se estabilize.

O mecanismo usando pacotes RR do RTCP permitiu obter valores muito melhores de desempenho se comparado com o VIC sem o mecanismo, ou mesmo com mecanismo usando "sockets", tanto em termos de perdas, taxa de transmissão e freqüência de quadros. O único parâmetro que não apresentou uma melhora visível foi o grau de qualidade. Isto se deve ao fato já mencionado de que o peso da fidelidade da imagem é o mesmo da freqüência de quadros. Se fosse considerado um peso maior para a freqüência de quadros (que no caso do VIC é o parâmetro que mais influencia na qualidade final percebida pelo usuário) poderíamos obter valores bem melhores. Um fato a ser considerado aqui é que as bruscas oscilações que ocorrem no comportamento dos parâmetros monitorados, se devem ao longo período de chegada dos pacotes RR do RTCP, que é da ordem de segundos.

A seguir, retoma-se uma avaliação com a literatura já existente, conforme resultados já apresentados em [19]. Além deste trabalho, existem na literatura vários trabalhos descrevendo modelos para adaptação de QoS na rede. Neles a ação de adaptação atua também sobre um único parâmetro de QoS o que continua bastante distante da perspectiva do usuário, que é a base deste trabalho.

Em [5] e [32] são apresentados mecanismos nos quais a adaptação representa um controle de qualidade de dados injetados na rede, visando adequar o fluxo ao seu estado a partir do monitoramento da taxa de perdas de unidades de transporte fornecida pelo protocolo RTCP. Contudo, em ambas as abordagens o problema da adaptação é endereçado do ponto de vista da rede, através do uso de parâmetros da camada de

comunicação, como taxa de perdas de unidades de transporte, taxa de transmissão por conexão, atraso e taxa de ocupação de "buffers" como variáveis de controle. Tais parâmetros, todavia, não refletem de forma clara a qualidade que realmente está sendo percebida pelo usuário. Além disso, a taxa de transmissão é alterada de maneira proporcional à variável monitorada, a despeito do fato do grau de qualidade não ser linear em relação à taxa de transmissão, isto é, a variação de um parâmetro não corresponde numa variação de outro na mesma proporção.

O uso de funções pra nortear o processo de adaptação, semelhantes a função $\mathcal{QoS}$ proposta neste trabalho, tem sido endereçado em alguns modelos de adaptação de QoS. Em [34] e [12], a descrição dos limites e prioridades de degradação é feita através de um *caminho de degradação* criado a partir das preferências do usuário em relação aos parâmetros resolução da imagem, freqüência dos quadros, algoritmo de compressão e exigências em termos de taxa de transmissão. O caminho nada mais é que uma tabela na qual cada entrada representa um nível de QoS. Quando a taxa de transmissão não permite a manutenção do nível de QoS corrente (taxa de perdas de unidades de transporte ultrapassa 10%), o mecanismo de adaptação de QoS é disparado, alterando o nível de QoS corrente para aquele imediatamente abaixo no caminho de degradação. Isso é feito através da troca de mensagens entre o cliente e o servidor, usando RTCP. Essa abordagem possui dois problemas: o primeiro é (novamente) o uso de parâmetros da camada de comunicação como variável de controle e o segundo é que, devido ao fato do caminho de degradação ser criado, direta ou indiretamente, pelo usuário (detalhes de sua construção não são endereçados no trabalho), provavelmente o mecanismo trabalha com poucos níveis de QoS, o que implica em alterações bruscas de qualidade.

5.6 Conclusões

Neste capítulo foi apresentado um modelo para adaptação de QoS de escopo global, que é o responsável pela adaptação da taxa de transmissão da aplicação à largura de banda da rede. O modelo é baseado em arquiteturas similares as dos sistemas realimentados de controle. A política de adaptação é centrada no usuário final. Para tal é usada uma função grau de qualidade que é composta de dois parâmetros encontrados na camada de aplicação do SMD, aqui no caso, a aplicação VIC. O que difere de

outras propostas de controle de QoS, nas quais os parâmetros monitorados pertencem à camada de comunicação ou de sistema da arquitetura do SMD e o mecanismo de adaptação realiza ações sobre somente um parâmetro.

O modelo do mecanismo da adaptação aqui implementado atua em ambientes melhor-esforço, isto é, sem reserva de recursos, tentando maximizar o grau de qualidade percebido pelo usuário, a partir de um valor inicial de referência, valor este que é degradado conforme a rede se encontre congestionada. Os resultados apresentados nos permitem concluir que o mecanismo obteve seu objetivo em fornecer uma melhor qualidade para o usuário final, o que poderia se melhorado ainda mais com a incrementação do mescanismo e a melhor definição de valores, como as faixas de carga da rede, o quanto degradar, etc.

Foram implementadas duas variantes do mecanismo de controle de QoS, uma usando um mecanismo de comunicação baseado em "sockets", e outra usando os pacotes RR do protocolo RTCP, visando aproveitar as melhores características do modelo de comunicação fornecido por cada uma delas. Isso nos permitiu obter resultados bastante diferenciados de um em relação ao outro, com o mecanismo de controle usando os pacotes RR obtendo um resultado bem melhor em termos gerais. Um fator limitante da implementação em ambos os modelos, deve-se a existência de um compromisso para definir o intervalo de tempo entre as ações de adaptação é maior. A medida que este foi maior, melhor são resultados; em contrapartida se perde muita informação do estado da rede durante este intervalo, o que não é desejável, principalmente se o estado da rede varia constantemente. Este também é um valor que se bem ajustado, pode melhorar ainda mais o desempenho do mecanismo de adaptação.

Capítulo 6

Conclusões e Perspectivas

Em ambientes onde não existe a possibilidade de reserva de recursos, as garantias de QoS são realizadas dotando as aplicações da possibilidade de alterarem a sua própria qualidade, de forma que elas possam fazer um melhor uso dos recursos do sistema. O uso de funções de QoS para nortear o processo de adaptação, tem sido apresentado em diversos trabalhos. O que se observa é que a grande maioria deles, realiza garantias de QoS atuando sobre parâmetros que afetam o usuário final de maneira indireta.

Neste trabalho, foi implementado um mecanismo de adaptação de QoS a partir de arquiteturas similares a de sistemas realimentados de controle, para vídeo-conferência usando a aplicação VIC desenvolvida para ambientes melhor-esforço. Esse mecanismo é implementado tendo por base a definição de um modelo genérico de QoS que fornece um modelo para integração do controle de QoS e dos mecanismos de gerenciamento, para ambientes com e sem reserva de recursos. Tal mecanismo é baseado em uma função grau de qualidade que é construída neste caso em cima de dois parâmetros de QoS da camada de aplicação, isto é, fidelidade da imagem e freqüência de quadros disponíveis na aplicação, apesar da interface permitir também a definição dos valores dos parâmetros de QoS: resolução e cor. A adaptação deles não é contemplada pelo mecanismo em decorrência do número restrito de valores configuráveis para esses parâmetros (três valores para resolução e dois para a cor), o que nos permitiu simplificar as implementações. A definição da função grau de qualidade permitiu que fossem estabelecidas políticas de adaptação orientadas ao usuário final, justamente por atuarem sobre parâmetros da camada de aplicação, que são os parâmetros perceptíveis pelo usuário final. A construção da função de QoS é feita por aproximação linear,

o que permite que as ações de adaptação de QoS sejam realizadas de forma suave e controlada, degradando ou melhorando essa qualidade frente ao contexto corrente do SMD no qual a aplicação está inserida.

O mecanismo implementado aqui é dividido em dois escopos: um local que atua sobre os sistemas finais em caso de sobrecarga de processador; e outro de escopo global que atua sobre o sistema final emissor em caso de congestionamento de rede. O mecanismo de escopo global apresentado neste trabalho foi implementado de duas formas diferentes usando modelos diferentes de monitoração, um baseado no grau de qualidade e uso de "sockets" e outro nos pacotes de relatório de receptor do protocolo RTCP. Em ambos os casos, a implementação tornou-se uma tarefa bastante crítica, no sentido de que qualquer processamento desnecessário é prejudicial ao funcionamento da aplicação como um todo, principalmente, porque as ações de adaptação atuam sobre os parâmetros de QoS na interface do VIC que é implementada através da linguagem interpretada Tcl/Tk, onde por sua vez toda a estrutura de dados é implementada em C++. O que ocorre é que alterações na interface sofrem um atraso até que elas se tornem perceptíveis ao usuário, o que é bastante crítico quando as alterações ocorrem de forma brusca sobre os valores dos parâmetros de QoS. O ideal seria que o mecanismo atuasse diretamente sobre as variáveis usadas pelo CODEC para definir os valores dos parâmetros de QoS; isto, com certeza, melhoraria o desempenho da aplicação em caso de sobre carga do SMD.

Daí, conclui-se que é aconselhável que o mecanismo de adaptação de QoS seja completamente integrado à aplicação, podendo assim manipular todas as suas estruturas internas. O que não ocorre no caso do VIC que é uma aplicação que não foi desenvolvida para suportar tal mecanismo. Outro fator a ser considerado é que, quanto mais parâmetros de QoS envolvidos no processo de adaptação, melhores serão os resultados, pois cada parâmetro terá de contribuir menos com esse processo.

Como perspectivas para trabalhos futuros sugere-se:

- a implementação do mecanismo de adaptação atuando sobre as variáveis do CODEC de forma direta, o que permite melhorar em muito o desempenho da aplicação como um todo, no lugar de usar a interface implementada em Tcl/Tk;
- o próprio desenvolvimento de uma aplicação de vídeo-conferência, que seja to-

talmente integrada com a filosofia do mecanismo de controle de QoS e inclua o mecanismo de adaptação;

- a implementação do mecanismo de QoS não só para a mídia de vídeo mas também para o áudio, como uma forma de melhorar o desempenho de uma vídeo-conferência como um todo.

Apêndice A

Descrição dos Componentes da Aplicação de Vídeo-Conferência – VIC

Este apêndice tem por objetivo dar uma descrição dos componentes da aplicação de vídeo-conferência – VIC.

main.cc : arquivo onde está definida a função *main* que é o ponto de partida para a construção da toda a estrutura do VIC. Algumas das funções relacionadas a este arquivo são: alocação de memória compartilhada, inicialização do Tcl e do Tk (para efeito de integração C++ e Tcl/Tk), controle dos parâmetros da linha de comando, incialização de recursos da máquina para instancia local, cores, valores padrões de variáveis, etc.

cf-main.tcl : este "script" é o responsável pela estruturação do VIC. A estrutura de controle básica é montada a partir dele através de procedimentos que chamam outros procedimentos, que por sua vez chamam primitivas em C++ que implementam suas funcionalidades:

init_local : este procedimento definido no "script" *cf-util.tcl*, inicializa parâmetros relacionados a informações sobre o usuário local: nome, "email", a partir de um arquivo de inicialização criado pelo próprio usuário na sua conta, ou por um criado pela própria aplicação caso não exista nenhum.

Esses parâmetros são úteis, pois definem algumas informações de valores de campos usados pelo protocolo RTP;

init_confbus : procedimento definido no "script" *cf-confbus.tcl.* Inicializa o barramento de conferência para a sessão do usuário, de forma que ele possa se comunicar com outras sessões de usuários que participam de uma mesma conferência.

init_network : procedimento definido no "script" *cf-network.* Que identifica o suporte nativo de rede[1] ao qual o usuário está conectado de forma que a transmissão dos dados possa respeitar as características de cada suporte.

init_gui : procedimento definido no "script" *ui-main.tcl.* Inicia a interface de usuários, é a partir desta interface que o fluxo de vídeo dos usuários é exibido.

init_late : este procedimento está definido no "script" *cf-util.tcl* e é o responsável pela criação do objeto fonte local, o que nada mais é do que a instância local do VIC. Variáveis RTP que representam informações sobre a instância (e.g. CNAME) são inicializados aqui, e também o uso ou não de "hardware" a partir da inicialização do VIC e do suporte de rede.

cf-network.tcl : define o suporte de rede nativo ao qual o usuário está conectado de modo que se possa estabelecer as características de transmissão dos dados sobre o suporte. De acordo com o suporte, é alocado um largura de banda para transmissão dos dados, caso nenhum valor seja passado na linha de comando, a partir deste valor é alocado um percentual do valor do canal de transmissão para o protocolo RTCP. Isto é feito antes da criação do gerenciador de sessão de forma que ele possa usar o resultado para computar o instante de tempo para o primeiro relatório. Este "script" também é o responsável pelo controle de criptografia e descriptografia caso os dados estejam sendo transmitidos em uma sessão privada. O arquivo que implementa as funcionalidades para o diversos suportes de rede é o *net.cc* e para criptografia é o *crypt.cc.*

net.cc : neste arquivo a classe "Network" implementa algumas funções básicas de configuração para que a transmissão/recepção dos dados ocorra de acordo com o

[1] As arquiteturas de redes suportados pelo VIC são o IP, IPv6, ATM e RTIP

suporte de rede específico. Do suporte de rede identificado, algumas das funções desta classe são herdadas por classes derivadas que implementam as funcionalidades do suporte rede. A cada suporte de rede está relacionado um arquivo com uma classe específica. Os arquivos correspondentes a cada suporte são: para o IP o *net-ip.cc*, para o ATM o *net-atm.cc*, para o RTIP o *net-rtip.cc* e para o IPv6 o *net-ipv6.cc*. Se o suporte é uma rede baseada no protocolo IP por exemplo, a classe "IPNetwork" herda características da classe "Network".

ui-main.tcl : este "script" define uma série de procedimentos para a construção da estrutura geral da interface principal (janela de usuários) do VIC e das funcionalidades relacionadas a ela. A partir deste "script" é feito o controle da exibição do fluxo de vídeo, do número de participantes da sessão, inclusão/remoção de participantes, atualização periódica dos dados estatísticos (taxa de recepção, quadros recebidos, percentual de perda, etc.) de cada participante. A cada fluxo de vídeo é atribuído um "consolidador" que dependendo do tamanho e da relação entre os componentes YUV, é o responsável pela montagem da imagem. A janela de configuração da instância local do VIC também é chamada a partir da interface principal, a partir do procedimento *build.menu* no momento em que o botão de menu é acionado pela primeira vez.

ui-ctrlmenu.tcl : define a estrutura da janela de *menu* (ou de configuração) local a partir do procedimento *build.menu* que é chamado da janela de participantes. A este "script" estão associados uma série de parâmetros a serem configurados para a transmissão da "stream" de vídeo como, taxa de transmissão máxima e freqüência de quadros. Para ambos os parâmetros as funções que efetivamente as implementam estão definidas no arquivo *grabber.cc* que possui a classe *Grabber*, que é herdada por uma classe específica relacionada ao dispositivo de captura de vídeo. O VIC em sua versão mais recente (vic-2.8ucl4) possui suporte para um número expressivo de placas de captura. Os arquivos associados a elas são listados no código como *grabber-[nome-do-dipositivo].cc*, onde nome-do-dispositivo é uma referência à placa usada. Neste trabalho o arquivo responsável pela implementação das funções é o *grabber-rtvc.cc* e a classe que herda as característica da classe *Grabber* é a *RTVCGrabber*. Nesta classe por exemplo, a função *fps* traduz

o valor desejado da freqüência de quadros em um valor realizável. Como ainda a placa possui a capacidade de compressão de vídeo via "hardware" para o formato JPEG a classe *RTVCJpegGrabber* definida também neste arquivo computa o valor da fidelidade selecionada na interface de forma que a freqüência de quadros possa respeitar o desejo do usuário.

Vários algoritmos de compressão de vídeo podem ser selecionados a partir da interface. Os arquivos que implementam as características de cada um, tanto para codificação quanto para decodificação são listados no código como *encoder-[nome-do-algoritmo].cc* e *decoder-[nome-do-algoritmo].cc*, por exemplo no caso do algoritmo JPEG os arquivos são *encoder-jpeg.cc* e *decoder-jpeg.cc*.

A partir deste "script" também pode-se fazer o monitoramento dos participantes da sessão chamando o procedimento *build.srclist* definido no "script" *ui-srclist.tcl*, e também o monitoramento da estatística global dos dados enviados e recebidos chamando o procedimento *create_global_window* definido no "script" *ui-stats.tcl*. A chamada dos procedimentos é feita a partir de botões criados na interface.

source.cc : este arquivo é responsável tanto pela manipulação dos dados da instância local do VIC (dados a serem transmitidos), quanto pelo gerenciamento dos dados dos outros participantes da sessão (dados recebidos). Neste arquivo a recepção e transmissão de dados obedece a especificação do protocolo RTP. Nele são definidos quatro classes: "PacketData", "PacketHandler", "Source" e "SourceManager". As duas primeiras são as responsáveis pela manipulação dos dados da instância local, fazendo a construção dos pacotes de dados, de forma que eles fiquem prontos para serem transmitidos pela rede. A classe "Source" define uma série de funções que fazem o controle sobre os dados da instância local e algumas funções que também servem para fazer o controle das outras instâncias participantes da sessão. Um grande número de parâmetros são controlados por essa classe: taxa de quadros, taxa de transmissão, pacotes transmitidos, percentual de perdas de pacotes, pacotes duplicados, fora de ordem, etc. Esta classe também possui uma função para criar um sincronismo de vídeo e áudio (somente para o VAT [2]) caso a sessão faça uso das duas mídias.

[2]Video Audio Tool

A classe "SourceManager" mantém o controle sobre os participantes da sessão, checando o número de participantes e quais estão ativos, faz a adição e remoção, faz também o monitoramento pra saber se não existem instâncias duplicadas de um mesmo participante na sessão. Cabe a esta classe fazer a demultiplexação dos dados de forma que os pacotes que chegam sejam endereçados corretamente para o participante correto. A otimização de desempenho controle do fluxo de dados de cada participante é feito a partir de uma tabela "hash", assim cada pacote recebido é endereçado corretamente para a janela de exibição de cada um.

session.cc : este arquivo é o responsável por toda a implementação relativa ao protocolo RTCP. A partir dele feito a construção dos pacotes da instância local e do tratamento das informações dos outros participantes. De posse desses dados, estatísticas de desempenho são geradas, como por exemplo, a média do tamanho dos pacotes, o intervalo de tempo entre pacotes, etc. Este arquivo também implementa a demultiplexação dos pacotes de forma que o controle possa representar o desempenho de cada participante individualmente.

mbus.cc : este arquivo é o responsável pela transmissão e recepção tanto dos pacotes de dados quanto dos pacotes de controle através do barramento de conferência.

Referências Bibliográficas

[1] ABDELZAHER, T. and SHIN, K. G. End-host Architecture for QoS - Adaptative Communication. In: *IEEE 4th Real-Time Technology and Applications Symposium*, Denver, Colorado, USA, June, 1998

[2] AURRECOECHEA, C., CAMPBELL, A. T. and HAUW, L. A Survey of QoS Architectures. *Multimedia Systems Journal*, vol. 6, no. 3, pp. 138-151, May, 1998.

[3] BANIKAZEMI, M. *IP Multicasting: Concepts, Algorithms and Protocols* http://www.cis-ohio.edu/ jain.

[4] BOLOT, J.-C, TURLETTI, T. and WAKEMAN, I. Scalable Feedback Control for Multicast Video in the Internet. *ACM Computer Comm.* vol. 28, no. 1, pp. 4-16. January, 1988.

[5] BUSSE, I., DEFFNER, B. and SCHULZRINNE, H. Dynamic QoS of Multimedia Applications Based on RTP. In: *1st International Workshop on High Speed Networks and Open Distributes Plataforms*. St. Petesburg, Russia. May, 1995.

[6] CAMPBELL, A. T., AURRECOECHEA, C. and HAUW, L. A Review of QoS Architectures. In: *IFIP 4th International Workshop on Quality of Service (IWQoS'96)*, Paris, March, 1996.

[7] CAMPBELL, A. T., COULSON, G. and HUTCHISON, D. Transporting QoS Adaptive Flows. *Multimedia Systems Journal*, no. 6, pp. 167-178, 1998.

[8] CHAN, T. *Unix programming using C++*. Prentice Hall, 1997.

[9] DAVIES, N et al. Supoting Adaptive Services in a Heterogeneous Mobile Environment. In: *Workshop on Mobile Computing System and Application*, IEEE Computer Society Press, pp. 153-157. Los Alamitos, California, 1994

[10] FLADENMULLER, A., SENEVIRATNE, A. and HORLAIT, É. *A Hybrid Management Scheme.* Institut Blaise Pascal, Masi 96/10, May, 1996.

[11] FLADENMULLER, A. *QoS Management Scheme for Distributed Multimedia Applications in Best Effort Environments.* Institut Blaise Pascal, Masi 96/12, May, 1996.

[12] FRY, M., SENEVIRATNE, A., VOGEL, A. Vogel and WITANA, V. Delivering QoS Controlled Media on the World Wide Web. In: *IFIP 4th International Workshop on Quality of Service (IWQoS'96).* Paris. March, 1996.

[13] GECSEI, J. Adaptation in Distributed Multimedia Systems. *IEEE Multimedia*, pp. 58-95, April-June, 1997.

[14] HAFID, A. and BOCHMANN, G. v. Quality-of-Service Adaptation in Distributed Multimedia Applications. *Multimedia Systems Journal*, vol. 6, pp. 299-315, 1998.

[15] HAFID, A., BOCHMANN, G. v. and DSSOULI, R. Distributed Multimedia Application and QoS: A Review. In: *Eletronic Journal on Networks and Distributed Processing*, vol. 2, no. 5, pp. 1-50, http://rerir.univ-pau.fr, 1998.

[16] JACOBSON, V. Congestion Avoidance and Control. *ACM Computer Comm.* vol. 18, pp. 314-329. August, 1998.

[17] KAMBA, T., BHARAT, K. and ALBERTS, M. C. *The Krakatoa Chronicle - An Interactive Personalized Newspaper on the Web.* http://www.w3.org/pub/Conferences/WWW4/Papers/93/.

[18] KOLIVER, Cristian. *Uma Abordagem de Adaptação de QoS para Aplicações Multimídia Distribuídas.* Exame de Qualificação(Doutorado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina. Florianóplois, 1998.

[19] KOLIVER, C., FARINES, J-M., FRAGA, J. S. and REIS, H. L. dos. Um Modelo para Adaptação de QoS Orientado ao Usuário Final. *XVIII Simpósio Brasileiro de Redes de Computadores.* Belo Horizonte. Maio, 2000.

[20] KOLIVER, C., FARINES, J-M., FRAGA, J. S. Controle Dinâmico de QoS Baseado no Uso do Protocole RTCP e Lógica Difusa. *SBC VI Simposio Brasileiro de Multimidia e Sistemas Hipermidia/SBMidia'2000.* Natal - Brasil. Junho, 2000.

[21] KOLIVER, C. and REIS, H. L. dos. *Uma Abordagem para Adaptação de QoS em Aplicações Multimídia Distribuídas.* LCMI-DAS. Relatório Interno. Outubro, 1999.

[22] KOLIVER, C. and REIS, H. L. dos. *A Implementação de um Mecanismo de Adaptação de QoS Orientado ao Grau de Qualidade.* LCMI-DAS. Relatório Interno. Novembro, 1999.

[23] McCANNE, S. and JACOBSON, V. vic: A Flexible Framework for Packet Video. *ACM Multimedia*, San Francisco, CA, November, 1995.

[24] NAHRSTEDT, K. O. *An Architecture for End-to-End QoS Provision and Its Experimental Validation.* University of Pennsylvania, Pennsylvania, August, 1995.

[25] NAHRSTEDT, K. O., HOSSAIN, A. and KANG, S.-M. A Probe-Based Algorithm for QoS Specification and Adaptation. In: *IFIP 4th International Workshop on Quality of Service (IWQoS'96).* Paris. March, 1996.

[26] *http://www.netperf.org.*

[27] QUINN, B. *IP Multicast Application: Challenges and Solutions*, Internet Draft 1998. ftp://www.draft-quinn-multicast-apps-00.txt.

[28] REININGER, D., RAYCHAUDHURI, D. and OTT, M. A Dynamic Quality of Service Framework for Video in Broadband Networks. In: *IEEE Network*, vol. 12, no. 6, pp. 22-45. November/December, 1998.

[29] RICHARDS, A., ROGERS, G., ANTONIADES, M. and WITANA, V. Mapping User Level QoS from a Single Parameter. In: *2nd International Conference on Multimedia (MMNS'98).* November, 1998.

[30] SABATA, B., CHATTERJEE, S., DAVIS, M. and SYDIR, J. J. Taxonomy for QoS Specifications. *WORDS'97*, 1997.

[31] SHULZRINNE, H., CASNER, S. and FREDERICK, R. and JACOBSON, V. GMD Fokus. *RTP: A Transport Protocol for Real-Time Applications.* GMD Fokus. RFC 1889, Berlin, German, January, 1996.

[32] SISALEM, D. End-to-End QoS Control Using Adaptive Applications. In: *IFPI 5th International Workshop on Quality of Service.* New York. May, 1997.

[33] VOGEL, A., BOCHMANN, G. v., DSSOULI, R., GECSEI, J., HAFID, A. and KERHERVÉ, B. University of Montreal, *On QoS Negotiations in Distributed Multimedia Applications.* TR-891, Montreal, Canada, 1993.

[34] VOGEL, A., KERHERVÉ, B., BOCHMANN, G. v. and GECSEI, J. Distributed Multimedia Applications and Quality of Service: A Survey. *IEEE Multimedia*, vol. 2, no. 2, pp. 10-19, Summer, 1995.

[35] WALLACE, G. K. The JPEG Still-Picture Compression Standard. *Communications of the ACM.* vol. 34, no. 4, pp. 30-44. April, 1991.

[36] WELCH, B. B. *Pratical programming in Tcl/Tk.* Prentice Hall, 1997.

[37] ZIMMER, J. A. *Tcl/tk for programmers with solved exercises that work with Unix and Windows.* Computer Society, IEEE. Los Alamitos, California. 1998.