UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CLEBER MACHADO ORTIZ

Um Protocolo de Perfil *Lite* para Colaboração Visual que utiliza Transferência de Arquivos com T.127

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Profa. Dra. Liane Margarida Rockenbach Tarouco
Orientador

Porto Alegre, abril de 2005.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Ortiz, Cleber Machado Ortiz

Um Protocolo de Perfil Lite para Colaboração Visual que utiliza Transferência de Arquivos com T.127 / Cleber Machado Ortiz. – Porto Alegre: PPGC da UFRGS, 2005.

102f:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2005. Orientador: Liane Margarida Rockenbach Tarouco.

1. Formatação Videoconferência. 2. Colaboração Visual de Dados. 3. H323 4. T.120 I. Tarouco, Liane Margarida Rockenbach. II. Título

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. José Carlos Ferraz Hennemann
Vice-Reitor: Prof. Pedro Cezar Dutra da Fonseca
Pró-Reitora Adjunta de Pós-Graduação: Profa. Valquiria Linck Bassani
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Prof. Flávio Rech Wagner
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

“A maior mentira do mundo: em determinado momento
de nossa existência, perdemos o controle de nossas vidas,
e ela passa a ser governada pelo destino”
— PAULO COELHO

AGRADECIMENTOS

A conquista deste mestrado representa para min não apenas um título, mas principalmente uma conquista de vida, uma transformação e um grande amadurecimento em minha vida profissional e pessoal. Profissional por oportunizar o meu crescimento como pesquisador e cientista. No campo pessoal, representou o momento para repensar minha vida, minhas atitudes e mudar a maneira de enfrentar os obstáculos.

A agradeço a Deus por estar sempre presente em minha vida, iluminando meus caminhos e ajudando-me a superar os obstáculos da minha vida e de toda fase do mestrado.

Durante o período do mestrado, convivi com pessoas que, de alguma forma, contribuíram para a conclusão desta etapa. Em primeiro lugar, gostaria de agradecer a minha orientadora, Dra. Liane Margarida Rockenbach Tarouco, pela inspiração que me proporcionou, por ter apostado e acreditado em mim, Professora Liane, muito obrigado por tudo!

Também não poderia deixar de agradecer ao Dr. Lisandro Zambenedetti Granville, que sempre me auxiliou quando precisei nos momentos mais difíceis, sempre com carinho e amizade. Meu grande amigo, gostaria de dedicar essa dissertação a você.

Um agradecimento especial aos colegas de grupo de pesquisa, que hoje são amigos: Ricardo Neisse, Michelle Leonhardt, Andrey Andreoli, Márcio Ceccon, Leandro Vaguetti, Evandro Pereira, Tiago Fioreze, Rafael Huff, enfim, todo o pessoal do LABCOM.

Queria agradecer também pelo auxílio inicial na construção deste trabalho ao Ronald e ao André - foram pessoas chaves para o andamento deste.

À Nilce pelo apoio fundamental e toda a sua compreensão. Sabemos o que significa essa conquista e com certeza você faz parte dela. Aos meus tios, Mário e Loreni, por terem me acolhido, incentivado e estimulado a ter a garra para superar os obstáculos.

À minha mãe, Maricleusa, um agradecimento todo especial, pelo exemplo de vida e por nunca medir esforços para me proporcionar as melhores oportunidades.

Enfim, gostaria de agradecer a todas as pessoas que de alguma forma, direta ou indiretamente, contribuíram para mais essa conquista em minha vida.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ADSL	Assimetric Digital Subscriber Line
B-ISDN	Broadband Integrated Services Digital Network
GCC	Controle Genérico de Conferência
IETF	Internet Engineering Task Force
ISDN	Integrated Services Digital Network
ISO	Internation Organization for Standardization
ITU	International Telecommunications Unions
MCS	Serviço de Comunicação Multiponto
MCU	Multipoint Control Unit
PDU	Protocol Data Unit
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RFC	Request for Comments
RNP	Rede Nacional de Pesquisa
RTP	Real-Time Transport Protocol
SCN	Switched Circuit Network
TPDU	Transport Protocol Data Unit
TSAP	Transport Service Access Point
UML	Unified Modeling Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Atualmente, quando falamos em Internet, não podemos somente pensar em serviços que utilizam correio eletrônico ou acessam informações de sites através de um navegador. Dentre alguns serviços utilizados na Internet, podemos destacar a videoconferência.

Videoconferência contempla, além do intercâmbio de áudio e vídeo entre duas pessoas ou grupos, também o compartilhamento de dados. Os serviços e protocolos definidos nas recomendações H.323 (para videoconferência) e T.120 (para a colaboração de dados) do ITU são bastante complexos e ocorrem muitos problemas na sua utilização em redes de pacotes funcionando segundo o princípio de *best effort* da Internet.

Nesta dissertação de mestrado são apresentados alguns resultados do estudo realizado sobre o contexto de videoconferência, suas soluções e protocolos, com ênfase nos protocolos padronizados para colaboração de dados dentro de uma videoconferência e a sua estrutura.

Esta dissertação também apresenta uma proposta de solução para uma aplicação que utilize transferência de arquivos nos moldes do padrão ITU T.127, mas que atenda aos requisitos de menor complexidade (tráfego e processamento). A proposta utiliza as estratégias de simplificação dos protocolos usados no contexto de colaboração de dados em ambiente que ainda mantenha compatibilidade com o ambiente T.120 e, sobretudo utilizando a Internet.

Palavras-Chave: Videoconferência, Colaboração visual de Dados, H.323, T.120, Transferência de arquivos.

A Protocol of Perfil *Lite* fo Visual Collaboration using File Transfer with T.127

ABSTRACT

In the current days when we talk about Internet, we cannot think only about services that use electronic mail or access information on a myriad of sites through a browser. Among some services used on the internet, we can highlight the videoconference.

Videoconference comprises, besides audio and video exchange between two people or among groups, data sharing. The services and protocols defined in the recommendations H.323 (for videoconference) and T.120 (for data collaboration) of ITU are very complex and many problems related to their usage in networks of packages operating according to the principle of *best effort* of the Internet occur.

In the current master's degree dissertation some results of the study accomplished on the videoconference context, its solutions and protocols with emphasis on the protocols standardized for data collaboration of a videoconference and its structure are presented.

This dissertation also presents a solution proposal for an application that uses files transfer according to the pattern ITU T.127, but that complies to the requirements of less complexity (traffic and processing). The proposal uses the strategies of simplification of the protocols used in the context of data collaboration in an environment that still keeps compatibility with the T.120 environment and, above all using the Internet.

Keywords: Videoconference, Visual Data Collaboration, H.323, T.120, Files Transfer.

1 INTRODUÇÃO

Atualmente, a existência de novas alternativas para as redes de computadores, tais como Redes Metropolitanas de Alta Velocidade (METROPOA, 2004), Internet2 (INTERNET2, 2004) e RNP2 (RNP, 2004) viabilizam o desenvolvimento de aplicações avançadas como videoconferência, vídeo interativo, bibliotecas digitais e laboratórios virtuais. Esse fato tem colaborado em grande parte para o favorecimento da comunidade acadêmica e instituições de pesquisa, além do setor comercial; e tem provocado um crescimento no uso do computador como uma ferramenta para mediar a comunicação em tempo real entre indivíduos e grupos, aumentando o número de ferramentas disponíveis e também o número de usuários desses serviços (LEOPOLDINO, 2004).

No que se refere à videoconferência, uma grande vantagem é o fato de pessoas geograficamente distantes poderem interagir através de um computador pessoal, eliminando, com isso, despesas e a perda de tempo inerente aos deslocamentos "tradicionais". Para que esta vantagem seja atingida, muitas empresas começam a buscar soluções passíveis de utilização num contexto mais abrangente, com um número maior de produtos e infra-estruturas, investindo mais em soluções padronizadas em lugar de soluções proprietárias. Atualmente, uma grande variedade de soluções para sistemas de videoconferência está disponível e cada aplicação, de acordo com o seu fornecedor, pode ter necessidades diferentes com relação a equipamentos, à infra-estrutura de comunicação e à qualidade de serviço (QoS) (VIDEOCONFERENCING, 2004). Considerando esses fatores, um sistema de videoconferência deve se adequar da melhor forma possível aos recursos que a infra-estrutura de rede oferece (LEOPOLDINO; MOREIRA, 2004).

O padrão H.323 do ITU (ITU, 1998) estabelece um conjunto de serviços e protocolos para assegurar a interoperabilidade entre clientes e servidores de videoconferência nas trocas de áudio, vídeo e controle da videoconferência. As aplicações que usam o H.323 são bastante complexas, pois contemplam o objetivo de permitir que ocorra um ajuste nos serviços oferecidos, tendo em vista as condições de transmissão, derivadas de condições tanto do enlace como da própria estação cliente do serviço. Atualmente, diversos trabalhos abordam essa questão, como o que é tratado por Carrion (CARRION, 2002), em que o usuário tem a possibilidade de interagir com o sistema a fim de escolher o tipo de fluxo das informações e qualidade das mídias, podendo atribuir priorização a estas. Também é tratada por Korb (KORB 2003) a sinalização dos terminais H.323, onde esses são capazes de requisitar QoS para a rede IP, com o objetivo de buscar aumentar a qualidade percebida pelo usuário em relação aos sinais de mídia, utilizando, para isso, gerenciamento baseado em políticas para controle de QoS dos terminais H.323.

Uma outra maneira de tratar as questões de videoconferência é, segundo Zanin (ZANIN, 2001), o trabalho realizado pelo ADAPT (*A Low Cost Video Conference Model for Personal Computers Running on IP Networks*). O ADPT procura implantar um modelo ou solução de baixo custo para videoconferência. Esse modelo busca solucionar as diferenças nas características do ambiente ou aplicação, ou seja, levar em consideração a configuração das estações envolvidas e as condições da rede, para que essas possam se adaptar ao serviço utilizado no momento. O VRVS (*Virtual Rooms Videoconferencing System*) (VRVS, 2004) tem como objetivo prover serviços de videoconferência e trabalho colaborativo de baixo custo, procurando utilizar o mínimo possível de largura de banda, onde essa é limitada em uma videoconferência ao consumo máximo de 256 Kbps. Uma outra forma de diminuir os custos relacionados com as aplicações que usam o H.323, é o trabalho que vem sendo realizado pelo projeto OpenH323 (OPENH323, 2004), que tem por objetivo tornar as aplicações que utilizam esse padrão, livres do alto custo de licenças para implementação comercial do protocolo. Atualmente, o valor das licenças é um dos motivos que acabam inibindo um maior desenvolvimento e utilização na comunidade da Internet dos recursos e padronizações de videoconferência.

Como o H.323 utiliza somente recursos de áudio e vídeo, é natural em uma videoconferência a solicitação de recursos auxiliares de dados, como *chat*, arquivos, estruturas compartilhadas tais como quadro-branco e também permitir que uma aplicação, sendo executada na estação de trabalho de um dos participantes, seja acompanhada ou mesmo controlada por um outro participante (compartilhamento de aplicações). Essa classe de serviços ou recursos auxiliares, foi objeto de padronização pelo ITU e sua especificação publicada na série de recomendações conhecida como T.120 (ITU, 1996). A utilização de áudio e vídeo através do H.323 e dos serviços auxiliares de dados no T.120, acaba possibilitando que essas duas padronizações, trabalhem em conjunto dentro de uma videoconferência (POON et al 2000).

A série T.120 contém um conjunto de recomendações que oferecem suporte para o estabelecimento de serviços auxiliares em uma arquitetura multiponto e em tempo real. O T.120 apresenta um conjunto de protocolos de comunicação e aplicação, mostra como os protocolos se relacionam entre si para o estabelecimento de conferências e ainda, a arquitetura proposta para a troca de dados em um ambiente de conferência de dados (ITU, 1996). Os protocolos que compõem o T.120, possuem a capacidade de manipular mais de uma conferência ao mesmo tempo, assim como um determinado terminal pode participar em mais de uma conferência (GLASMANN 2002).

A série contempla a capacidade de serem manipuladas diferentes taxas de fluxo de informação numa mesma conferência de dados, dependendo dos limites impostos por cada uma das tecnologias de rede usadas pelas estações participantes. As aplicações que usam protocolos T.120 são bastante complexas, pois precisam também contemplar e permitir que ocorra um ajuste nos serviços oferecidos, tendo em vista as condições de transmissão, derivadas de condições tanto do enlace como da própria estação cliente do serviço. Em relação a isso, pode-se chegar a conclusão que esses fluxos de informação tanto de controle como de uma aplicação que use transferência de arquivos binários, ocasionam um grande tráfego na rede, um maior processamento das estações envolvidas e um número significativo de diversas chamadas entre as camadas que compõe o T120.

A série T.127 (ITU 1995a), que compõe parte do T.120, define um protocolo para suporte a troca de arquivos binários dentro de uma conferência interativa de grupo, trabalhando em um ambiente onde o T.120 é usado. No T.127 existem fatores que

influenciam no desempenho de uma conferência de dados; um exemplo é a presença de mensagens de compartilhamento de aplicações que não foram solicitadas. Essas chamadas geram um processamento desnecessário tanto na máquina como na rede, consumindo recursos destinados para aplicações que foram legitimamente acionadas pelo usuário (ORTIZ; GRANVILLE; TAROUCO, 2003a) (ORTIZ; GRANVILLE; TAROUCO, 2003e) .

Nesta dissertação é apresentada uma proposta de solução para uma aplicação que utiliza transferência de arquivos binários nos moldes do padrão ITU T.127 (ITU 1995a), mas que atenda aos requisitos de menor complexidade (tráfego e processamento). A solução proposta consiste em um modelo minimalista que contenha as funções mínimas para o funcionamento de uma aplicação de transferência de arquivos sobre o T.120, com o objetivo de diminuir a complexibilidade na implementação das chamadas e primitivas definidas na recomendação. A diminuição ou simplificação das chamadas deve permitir que uma aplicação T.127 seja executada com menos recursos de processamento da máquina do usuário. Aplicações que não utilizam tanto processamento, podem ser fundamentais para aquele usuário que não dispõe de uma máquina com os recursos mais sofisticados. Já no que diz respeito ao tráfego na rede, a simplificação das chamadas ocasiona um número menor de informações tanto de sinalização quanto de dados. Também é importante para essa proposta minimalista não perder algumas características desejáveis como: a escolha de uma solução padronizada, que não limite as suas opções a produtos desenvolvidos por somente um fabricante e que permita a comunicação através da Internet (ORTIZ; GRANVILLE; TAROUCO, 2003c) (ORTIZ; GRANVILLE; TAROUCO, 2003d).

O restante do trabalho está assim organizado: no capítulo 2, são apresentados os trabalhos relacionados, uma análise da videoconferência e os protocolos envolvidos nessa e na colaboração visual de dados. O capítulo 3 mostra algumas das contribuições desta dissertação, apresentando o modelo minimalista para transferência de arquivos e comentando algumas simplificações propostas. O capítulo 4 apresenta o protótipo para solução *lite* , os testes realizados e as monitorações obtidas através dessa solução. Por fim, no capítulo 5, são apresentadas as conclusões resultantes desta dissertação e os trabalhos futuros. .

2 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados os principais conceitos relacionados com os serviços de videoconferência e colaboração visual de dados, os protocolos utilizados nestas e também os trabalhos relacionados.

2.1 Videoconferência

Os sistemas de videoconferência permitem que se trabalhe de forma cooperativa e se compartilhem informações e materiais de trabalho sem a necessidade de locomoção geográfica. Através da videoconferência são criadas novas possibilidades para reuniões de pessoas distantes umas das outras, provendo diversos meios bem mais interessantes pela comunicação visual e a interação entre os demais participantes, superando, desta forma, o conhecido e-mail, os diversos canais de bate-papo e até mesmo o telefone. Sobre os tipos de interação em videoconferência, existem diversos sistemas disponíveis, classificados segundo a forma de comunicação que utilizam (LEOPOLDINO, 2002):

- Ponto-a-ponto: quando existe a comunicação entre duas pessoas ou mais. Nesta comunicação não é necessária a presença de um MCU (Unidade de Controle Multiponto), sendo usados apenas dois terminais de videoconferência.

- Modelo centralizado: Este modelo é baseado no modo de comunicação ponto-a-ponto ou *unicast.* No caso de existirem três ou mais pontos para se conectarem entre si, a comunicação é possível desde que haja um MCU. Nas sessões de videoconferência baseadas neste modelo, cada participante estabelece uma conexão com o MCU central e a distribuição do fluxo de áudio, vídeo e dados para cada participante são feitos pelo MCU que mescla os vários fluxos de áudio, seleciona o fluxo de vídeo correspondente e retransmite o resultado para todos os outros participantes.

- Modelo descentralizado: O modelo descentralizado compartilha características de controle comum com o modelo centralizado, mas o fluxo de mídia é manuseado diferentemente. Uma das entidades participantes deve ser um MC (Controlador Multiponto) que, independente do modelo de comunicação, provê o controle de três ou mais participantes durante uma sessão multiponto. O MC tipicamente é colocado com um dos participantes. Enquanto no modelo centralizado o MCU faz o processamento de mídia, no modelo descentralizado os fluxos de mídia são enviados e recebidos por todos os participantes. Não existe MCU para processar os múltiplos fluxos; cada participante é responsável por sua própria mesclagem de áudio e seleção de vídeo. A mídia pode ser

enviada entre todos os participantes utilizando *multicast*, ou múltiplos *unicast* se a rede não suportar *multicast*.

- Modelo híbrido: O modelo híbrido tenta mesclar o melhor dos dois modelos anteriores, mantendo a consistência dos dados através de um armazenamento centralizado e suportando visões individualizadas através do uso de *front ends*. Neste exemplo, um MCU separado é usado para manusear o áudio, dados e controle de funções, e o vídeo é distribuído por *multicast* conservando a largura de banda.

2.2 Padronização do serviço de videoconferência: H.323

O ITU-T *(Telecommunication Standardization Sector of International Telecommunication Unio*n) propôs a Recomendação H.323, que descreve uma arquitetura contendo terminais, equipamentos e serviços para comunicação multimídia sobre LANs *(Local Area Network*s), sem garantia de qualidade de serviço. Na verdade a H.323 é uma recomendação "guarda-chuva", referenciando outras recomendações ITU-T e até o protocolo RTP *(Real-Time Transport Protoco*l) do IETF. A primeira versão desta recomendação foi publicada em 1996 e a segunda, em 1998. Esta última contém melhorias no tocante à segurança, diminuição do tempo necessário para liberação do canal após a chamada ter sido atendida pelo destino, incorporação de serviços adicionais como transferência, redirecionamento, entre outros e maior integração com T.120 (protocolo de transmissão de dados multimídia) (LOGUINOV; RADHA, 2004) (CALYAM et al, 2004) (IMAI et al, 2003) .

Basicamente, destina-se à transmissão em ambiente com uma ou mais redes locais, sem garantia de qualidade de serviço. Como por exemplo: Ethernet (IEEE 802.3), Fast Ethernet (IEEE 802.10), FDDI (no modo sem garantia de QoS) e Token Ring (IEEE 802.5).

2.2.1 Componentes da arquitetura

O ITU-T define na recomendação H.323 a descrição dos seus componentes, visando a prover serviços de comunicação multimídia sobre redes de pacotes, tais como LANs, MANs e WANs incluindo a Internet (não garantem qualidade no serviço). Os componentes descritos compreendem terminais, *gateways, gatekeepers,* MCU (*Multipoint Control Units*) (Figura 2.1)

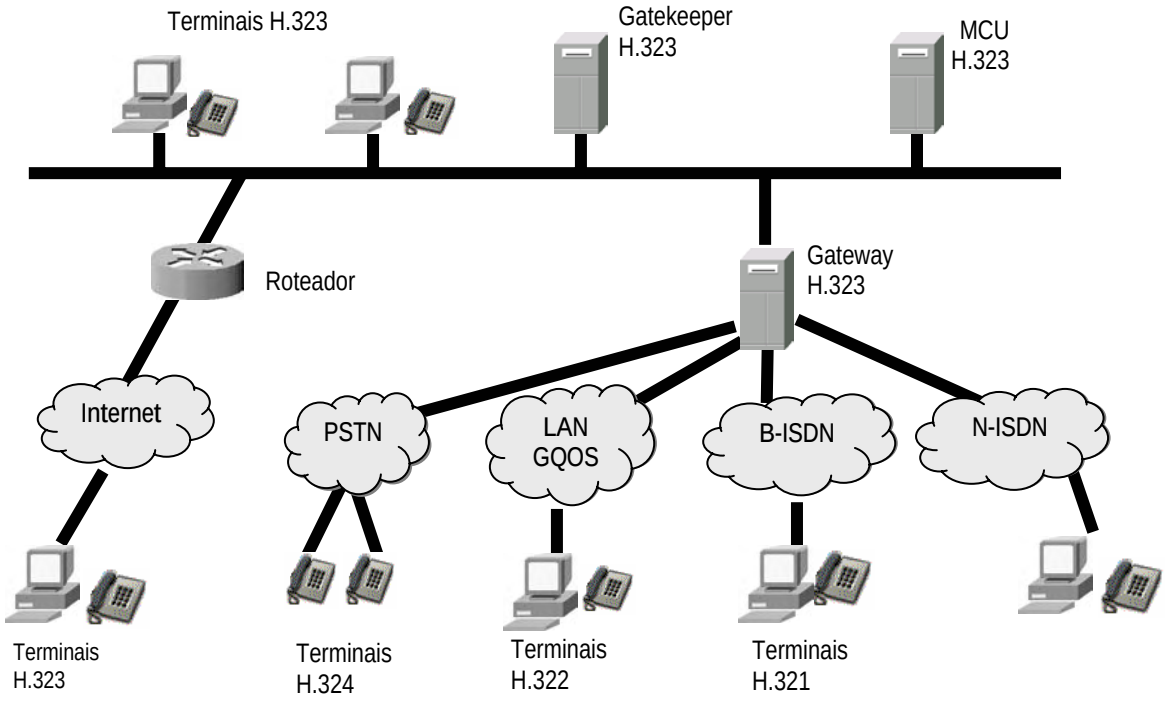

Figura 2.1: Componentes da arquitetura H.323

- Terminais : Por definição é um ponto de rede que provê comunicação bidirecional, em tempo real, com outro equipamento terminal, *gateway* ou MCU. Pode ser um equipamento telefônico com a funcionalidade de comunicação através do protocolo IP, um microcomputador com *software* específico e *hardware* multimídia para comunicação de voz ou qualquer outro tipo de equipamento que possa estar conectado em rede e permita a transferência apenas voz; voz e dados; voz e vídeo; ou voz, dados e vídeo, entre os usuários (BALDI; OFEK, 2000).

- Gateway: Converte, apropriadamente, diferentes formatos de mensagens (exemplo: H.225.0 / H.221 - estrutura de quadros em canais de 64 a 1920 Kbps em tele-serviços audiovisuais) e procedimentos de comunicação (exemplo: H.245 / H.242 - estabelecimento de comunicação entre terminais audiovisuais em canais digitais até 2 Mbps). A conversão entre diferentes formatos de áudio, vídeo ou dados também pode ser feita pelo *gateway*.

  Os terminais H.323 podem conversar entre si em uma mesma rede, sem o envolvimento de um *gateway*. Entretanto, quando um terminal H.323 desejar conversar com uma pessoa que possui apenas um telefone comum, conectado na rede pública de telefonia (PSTN) ou terminal H.320 e em linhas ISDN, isto deverá executar todas as traduções necessárias para que a comunicação aconteça. A sua função é refletir as características de um terminal na rede IP para um terminal na SCN e vice-e-versa.

- Gatekeeper: É um elemento opcional no sistema H.323, provendo serviços de controle de chamadas para os terminais. Possui como funções obrigatórias a tradução de endereços (permite o uso de apelidos (*alia*s) em lugar dos Endereços de Transporte), controle de admissão (autoriza o acesso de recursos usando por base critérios como: permissão de acesso em período predeterminado, banda disponível, entre outros), controle de banda (rege os pedidos de troca de banda em uso) e a gerência de zona (prove as funções acima para terminais, MCUs e *gateways* nele registrados) (SAJAL et tal 2002).

  Opcionalmente, o *gatekeeper* pode apresentar as funções de controle de sinalização de chamadas (pode controlar todo processo de sinalização de chamadas entre terminais, ou deixar que os próprios terminais tratem da sinalização), autorização de chamadas (através da sinalização prevista na H.225.0, o *gatekeeper* pode rejeitar chamadas de um terminal sem autorização adequada), gerência de banda (controla o número de acessos simultâneos de terminais H.323 à LAN, podendo rejeitar uma chamada por falta de banda disponível), gerência de chamadas (controla que terminais possuam chamadas estabelecidas, para o caso de uma nova chamada a um terminal ocupado ser informada aos módulos de direito).

  Um conceito importante, introduzido pela segunda versão do H.323 é o de Zona. Por definição, Zona abrange a coleção de todos os terminais, *gateway*s, e MCUs, gerenciados por um único *gatekeepe*r. Cada Zona possui apenas um *gatekeeper* e não precisa estar confinada geograficamente a mesma LAN (Figura 2.2).

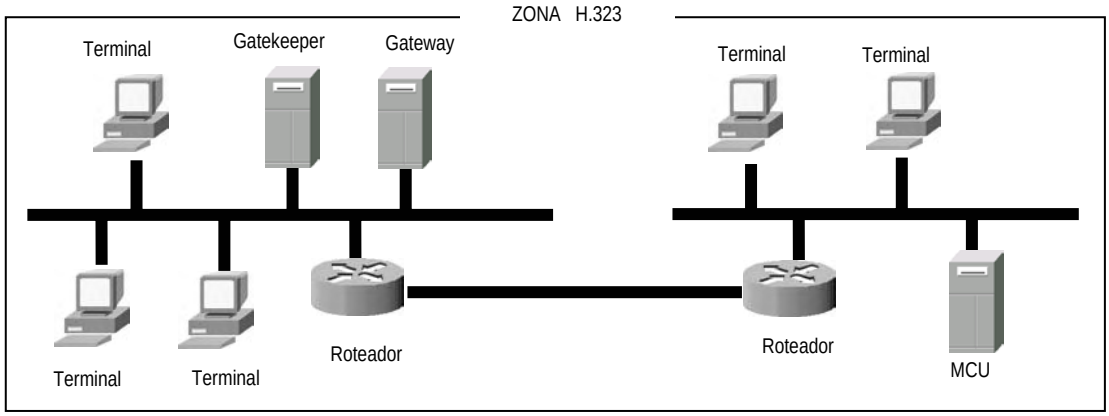

Figura 2.2: Zona H.323

- MCU: O MCU é um componente do H.323 que provê a capacidade para suportar conferências multiponto. Pode estar dividido em um MC (obrigatório) e MPs (não obrigatório).

  MCU suporta conferências multipontos centralizadas através de um MC e da agregação de sinais de áudio, vídeo e dados. Conferências descentralizadas constituem de um MC e suporte para agregação de dados através de um MP, que implementa as funções, conforme a recomendação T.120, usando capacidades de áudio e vídeo descentralizadas (PINHEIRO, 2002).

> Um *gateway* pode implementar um MCU no lado interno da rede. Um *gatekeeper* poderá também conter um MCU. Em qualquer dos casos, as funções devem estar separadas e não podem interferir uma na operação da outra. Isto significa que, embora as funções possam coexistir em um mesmo equipamento, elas devem ser implementadas separadamente e a ativação de um serviço não implica na ativação do outro e vice-versa.

2.3 Protocolos e colaboração visual de dados

Um dos maiores objetivos da utilização de padrões é permitir que uma mesma tecnologia possa ser utilizada por diversos fabricantes de equipamentos e softwares e que estes equipamentos (mesmo sendo de fabricantes diferentes) possam interoperar. Uma maior interoperabilidade faz com que a colaboração visual de dados possa atingir um número maior de usuários.

2.3.1 Colaboração visual de dados

Dentro do ambiente de colaboração visual de dados existem diversas ferramentas como: videoconferência, *video streamming*, compartilhamento e transferência de informações e imagens. A interação entre as pessoas que utilizam as ferramentas de colaboração visual pode ser assíncrona - onde ocorre a troca de informações em momentos diferentes, algo semelhante ao correio eletrônico ou síncrona, com interação em tempo real e da visualização on-line de uma conversa em tempo real (*chat*).

A colaboração visual de dados procura utilizar, desenvolver e aumentar a interação entre os seres humanos. Dentre as aplicações que podem ser úteis na colaboração visual de dados destacam-se: a educação à distância, pesquisa distribuída, atendimento a clientes em centrais de atendimento, desenvolvimento de novos produtos por especialistas que encontram-se separados pela distância, assistência médica a pacientes e telemedicina; são exemplos de muitas atividades que aumentam a comunicação entre as pessoas (KARAHASHI et al 1999).

Conforme os exemplos citados no parágrafo anterior, pode-se concluir que quanto mais a colaboração visual de dados for utilizada, maior o aumento de diferentes tipos de soluções de comunicação e colaboração, que vão desde videoconferências e conferência de dados até diferentes combinações com outras funcionalidades, como compartilhamento de *desktop* e transferências de arquivos.

2.3.2 O protocolo para conferência de dados

A recomendação T.120 contém uma série de recomendações que oferecem suporte para o estabelecimento de conferências multimídia de dados em uma arquitetura multiponto e em tempo real. Esta recomendação, apresenta um conjunto de protocolos de comunicação e aplicação, que são coletivamente referenciados como a Série T.120.

A padronização T.120 descreve as recomendações que compõe a série. Relata como as recomendações se relacionam entre si para o estabelecimento de conferências e ainda, a arquitetura proposta para a troca de dados em um ambiente de conferência multimídia. Dentre as recomendações da Série T.120, pode-se destacar (ITU 1996):

- Recomendação ITU-T T.121 – (1996) – Modelo genérico de aplicação (*Generic application template*)

- Recomendação ITU-T T.122 – (1996) – Serviço de comunicação multiponto para a definição de serviços de conferência audiovisual e audiográficas (*Multipoint communication service for audiographics and audiovisual conferencing service definition*)

- Recomendação ITU-T T.123 – (1996) – Pilha de protocolos para aplicações de teleconferência audiovisual e audiográficas (*Protocol stacks for audiographic and audiovisual teleconference application*)

- Recomendação ITU-T T.124 – (1996) – Controle de conferência genérico (*Generic conference control*)

- Recomendação ITU-T T.125 – (1996) – Especificação do protocolo de serviço de comunicação multiponto (*Multipoint communication service protocol specification*)

- Recomendação ITU-T T.126 – (1996) – Protocolo de anotação e de imagens estáticas Multiponto (*Multipoint still image and annotation protocol*)

- Recomendação ITU-T T.127 – (1996) – Protocolo para transferência multiponto de arquivo binário (*Multiponit binary file trasfer protocol*)

- Recomendação ITU-T T.128 – (1996) – Compartilhamento de aplicações multiponto (*Multiponit application sharing*)

Os protocolos que compõem a Série T.120 possuem a capacidade de manipular mais de uma conferência ao mesmo tempo, assim como, um determinado terminal pode participar em mais de uma conferência (Figura 2.3).

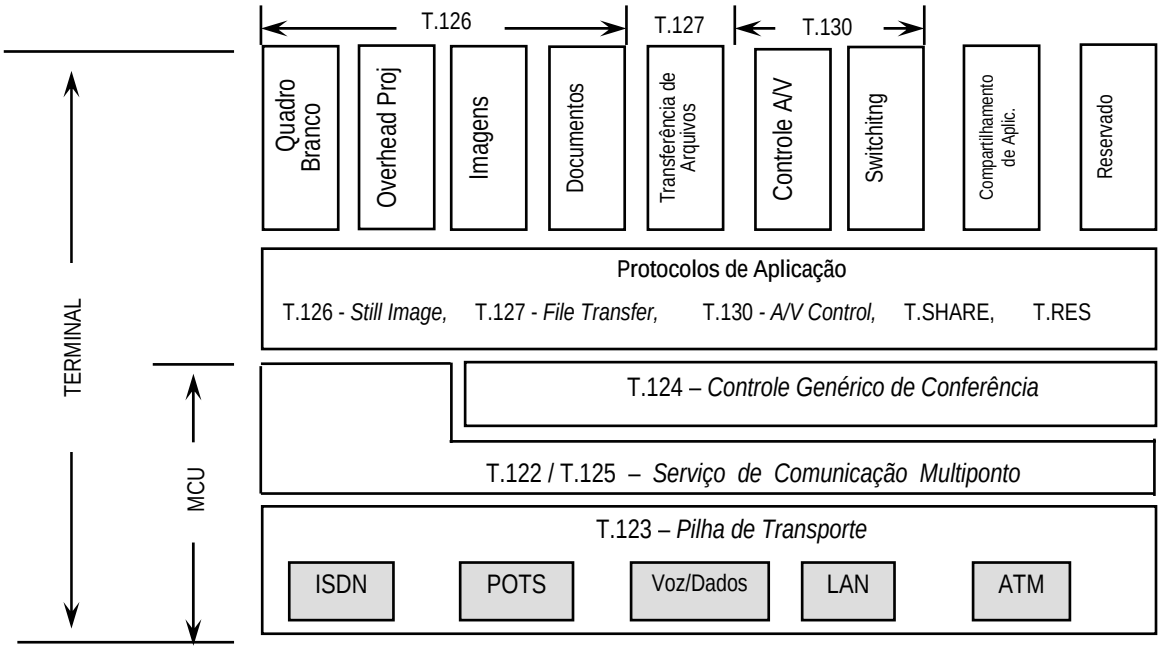

Figura 2.3: A arquitetura da recomendação T.120

A recomendação T.120 especifica como deve acontecer o estabelecimento e o gerenciamento de comunicações interativas de dados em uma configuração multiponto que pode envolver uma infinidade de redes diferentes. A função da recomendação é oferecer um serviço de comunicação de dados para os participantes de uma conferência. O serviço deve ser independente da tecnologia de rede que está sendo utilizada pelos participantes em cada um dos pontos que compõem a conferência. Entre os tipos de rede definidos na recomendação estão: PSTN, ISDN, CSDN, PSDN, B-ISDN e LANs.

Os protocolos definidos nas recomendações devem permitir que os participantes de uma conferência possam se comunicar entre eles. As recomendações devem também assegurar a interoperabilidade entre as aplicações envolvidas na troca dos dados, através da definição de alguns protocolos padronizados para implementação do quadro branco (*whiteboard*), transferência de arquivos e troca de imagens estáticas. Deve ficar claro que os protocolos de aplicação utilizados não precisam necessariamente ser padronizados pelas recomendações da série T.120, desde que todos os pontos participantes da conferência utilizem este mesmo protocolo para a comunicação (BOTIA; RUIZ; SKARMETA, 2004).

Os protocolos que compões a série T.120 possuem a capacidade de manipular mais de uma conferência ao mesmo tempo, assim como, um determinado terminal pode participar em mais de uma conferência. Não existe nenhuma imposição quanto à quantidade de informação a ser transmitida, nem em relação à taxa de transferência utilizada, uma vez que os protocolos propostos pelo T.120, possuem a capacidade de manipular as diferentes taxas de fluxo de informação, dependendo apenas dos limites impostos por cada uma das tecnologias de rede que participam da conferência. A segmentação dos dados é executada automaticamente em benefício das aplicações, mas

a remontagem das mensagens deve ser feita pela própria aplicação, quando os dados são entregues (ORTIZ;GRANVILLE;TAROUCO, 2003b).

Para que a troca de informações multiponto através de uma conferência T.120 ocorra, apenas as recomendações T.122, T.123, T.124, T.125 são obrigatórias e devem ser implementadas nos softwares ou dispositivos de conferência T.120. Todos os produtos que seguem a recomendações T.120, estão aptos a:

- estabelecer e manter conferências multimídia independente de plataforma e fabricantes de hardware ou software;
- gerenciar os múltiplos participantes e programas que compõe as conferências;
- trocar dados, de forma segura, entre os dispositivos ou aplicações que compõe a conferência.

A recomendação T.120 não impõe muitos requisitos sobre as conexões entre os nós que compõe a conferência, exceto que, os nós participantes de uma conferência devem estar organizados hierarquicamente e deve existir somente um nó no topo da árvore. A topologia na qual a conferência pode estar organizada pode variar entre estrela ou cadeia de nós ou uma combinação destas duas, entretanto, a comunicação entre os terminais não pode apresentar redundâncias (*loops*) e o nó que está no topo da conferência deve estar presente desde o início da conferência até o seu final. Se o nó do topo falhar ou deixar a conferência, esta é encerrada. Caso um nó intermediário deixar a conferência, apenas os nós abaixo dele serão eliminados da conferência (ORTIZ;GRANVILLE;TAROUCO, 2003f).

A recomendação T.120 trata uma conexão ponto-a-ponto como a forma mais simples de conexão multiponto e define duas possibilidades para o estabelecimento de conexões multiponto. A primeira através de terminais com múltiplas portas, onde cada porta deve implementar ao menos os protocolos especificados nas recomendações T.122, T.123, T.124, T.125, que são obrigatórias. O terminal pode atuar como uma ponte entre três ou mais nós, permitindo que uma comunicação multiponto possa ser estabelecida entre estes nós. A segunda opção para o estabelecimento de uma conexão multiponto é através de MCUs, que são dispositivos especializados no estabelecimento e gerenciamento de conexões multiponto, atuando como ponto de ligação entre as conexões dos vários nós (terminais ou outros MCUs) que compõem a conferência.

A arquitetura proposta pelo T.120 é composta de três partes principais: protocolos de aplicação, nó de controle e a infra-estrutura de comunicação. Cada uma das partes que compõem o modelo T.120 são brevemente discutidos na recomendação T.120, entretanto são amplamente discutidos individualmente em cada uma das recomendações. O modo genérico de funcionamento dos protocolos é semelhante ao empregado pelo modelo de referência OSI, onde cada camada oferece serviços à camada imediatamente acima e utiliza serviços da camada abaixo, através da troca de unidades de dados de protocolo (PDUs).

As aplicações de usuário não são objeto de padronização, mas deverão possuir capacidade para comunicação multiponto e deverão ser projetadas para a utilização da infra-estrutura T.120. As aplicações não têm influência direta no que diz respeito à infra-estrutura de interconexão e portanto, poderão ser dependentes de plataforma e sua

performance é extremamente dependente dos protocolos de aplicação que ela utiliza. O modelo T.120 foi projetado para dar suporte concorrente a múltiplas aplicações de usuário, oferecendo a elas mecanismos para o controle e gerenciamento dos recursos de comunicação.

No conjunto de recomendações proposto pela série T.120, existem algumas que tratam exclusivamente de protocolos de aplicação para comunicação em uma ambiente multiponto. Estas recomendações definem os requisitos mínimos que são necessários para assegurar a interoperabilidade e compatibilidade entre aplicações de diferentes fabricantes. A recomendação T.128 trata as questões envolvidas no compartilhamento de aplicações multiponto. A recomendação T.127, por sua vez, define como a transferência binária de arquivos deve ser implementada em um ambiente multiponto. Já a recomendação T.126 trata de outras facilidades, tais como, a visualização e modificação de imagens estáticas, quadro branco compartilhado.

É importante salientarmos que as aplicações podem utilizar qualquer combinação de protocolos padronizados ou não padronizados na sua implementação. O nó de controle é o elemento que cuida do gerenciamento T.120, tanto nos terminais quanto no MCU. Ele direciona comandos para o GCC (Controle Genérico de Conferência) que têm o poder de iniciar e controlar as sessões. Entretanto, o nó de controle não é alvo da série T.120 e suas interfaces foram definidas somente onde ele deve interagir com o GCC. A infra-estrutura de comunicação engloba quatro das recomendações da série T.120: T.122, T.123, T.124 e T.125. A infra-estrutura de comunicação tem a função de garantir uma comunicação segura em um ambiente multiponto, sendo capaz de acomodar concorrentemente, múltiplas aplicações independentes (IMAI et al 2003).

A conexão entre os nós que compõe a conferência pode acontecer em um ambiente composto de qualquer combinação de redes de pacotes ou de comutação por circuitos. Esta infra-estrutura de comunicação é composta por três elementos padronizados: o GCC, o MCS (serviço de comunicação multiponto) e perfis de protocolos de transporte associados com cada uma das tecnologias de redes suportadas.

O GCC é descrito na recomendação T.124 e define um conjunto de serviços para o estabelecimento e o gerenciamento de conferências multiponto. O GCC mantém e controla uma base de informações sobre o estado de cada conferência as quais ele está servindo. Através do GCC é possível consultar um determinado MCU e encontrar uma conferência específica. Desta forma, as aplicações criam, se juntam ou convidam outros para participarem de uma conferência (BAUER, KORTUEM, SEGALL 1999).

Para manter o estado atual das conferências, conforme cada aplicação inicia ou encerra sua participação em uma conferência, o GCC imediatamente atualiza a sua base de informações que pode ser usada para notificar os outros participantes a respeito destes eventos. O GCC também oferece facilidades para a incorporação de segurança e para o monitoramento dos recursos que estão disponíveis e que estão em uso no MCS.

O MCS oferece um mecanismo genérico que serve para a resolução de quase todos os problemas de comunicação multiponto entre aplicações, oferecendo um serviço de dados seguro orientado à conexão. Em uma conferência existem múltiplos pontos que estão logicamente conectados, isto é chamado na série T.120 como um domínio, mas que genericamente pode ser comparado ao conceito de conferência. O MCS reúne conexões ponto-a-ponto e as combina para formar um domínio multiponto. Dentro de cada domínio existe um grande número de canais que podem oferecer comunicação de dados um-para-um, um-para-muitos ou ainda muitos-para-muitos.

Cada um dos domínios possui somente um provedor principal, que fica localizado no topo da árvore daquele domínio, pois os nós que compõem a conferência devem estar organizados hierarquicamente. Desta forma, a entrega de dados segue o caminho mais eficiente até os nós que devem receber os dados. Entretanto, o MCS também oferece um mecanismo para garantir que os dados que são originados a partir de diferentes nós cheguem na mesma ordem em todos os nós que participam da conferência.

O MCS atua como um provedor de recursos para as camadas acima, mas é independente da tecnologia de rede, provendo canais lógicos de comunicação e tokens (mecanismo de sinalização) sob demanda, conforme as necessidades de cada conferência.

O último nível da arquitetura T.120 está especificada na recomendação T.123, que oferece suporte de transporte para protocolos de rede padronizados e não padronizados. A sua função principal é a entrega segura e seqüencial, ponto-a-ponto de PDUs vindos do MCS. Se houver necessidade de segmentação dos dados, esta recomendação também está apta para executar a tarefa, arranjando os novos segmentos em uma seqüência lógica coerente, para a reordenação dos dados no destino.

2.4 ADAPT

O trabalho elaborado por Zanin (ZANIN, 2001) propõe a elaboração de um modelo para videoconferência em computador pessoal sobre redes IP. Segundo o autor, o modelo é voltado para a videoconferência de baixo custo, beneficiando os usuários que nem sempre podem contar com um equipamento de última geração, podendo então contornar os problemas encontrados em uma videoconferência que utiliza equipamentos que não são de última geração.

A idéia do ADAPT parte de um sistema de videoconferência tradicional, utilizando a transmissão e recepção de áudio e vídeo conforme a recomendação F-730 do ITU-T (ITU, 1992). No ADAPT são somados ainda dois agentes de software para dar possibilidade do sistema modificar o tráfego (quantidade de bytes) gerado durante a sessão.

O Agente de Observação da Rede é o primeiro agente de software utilizado pelo ADAPT, tem a finalidade de obtenção dos dados da rede. O Agente de Videoconferência é o software responsável pelas alterações na qualidade da mídia.

O modelo elaborado por Zanin propõe a implementação dos dois agentes com o objetivo de melhorar a taxa de *frames* por segundo do vídeo, onde a nitidez deve ser mantida juntamente com a clareza do áudio. Para que as aplicações consigam realizar uma troca de informações relativas às condições da rede, capacidade de processamento ou às exigências do usuário. O modelo busca, através da transmissão periódica de pacotes, repassar essas informações.

As alterações de configurações das mídias, que são efetuadas pelo Agente de Videoconferência, são efetuadas com definições de uma série de regras pré-definidas no próprio software agente. A definição de regras é uma questão delicada, pois é difícil estabelecer uma regra estática que venha a servir para tipos diferentes de redes, computadores, câmeras de vídeo, placas e entre outros. O modelo apresentado não busca definir as regras estáticas, mas sim definir regras que sejam as mais abrangentes possíveis de atingir a todos os casos. Essa abrangência pode ser obtida, deixando

disponível para o usuário a possibilidade de poder definir características do equipamento a ser utilizado. Com posse dessas informações, o ADAPT pode fazer com que a aplicação local e a remota tenham conhecimento sobre como agir na transmissão, recepção e adaptação da conferência.

Com a posse das informações de configuração das mídias, obtidas pelo Agente de Observação da Rede e do usuário, o Agente de Videoconferência poderá alterar a configuração das mídias e adaptar o tráfego, com o objetivo de minimizar os problemas encontrados em uma videoconferência. Na figura 2.4, é apresentada a estrutura do modelo ADAPT.

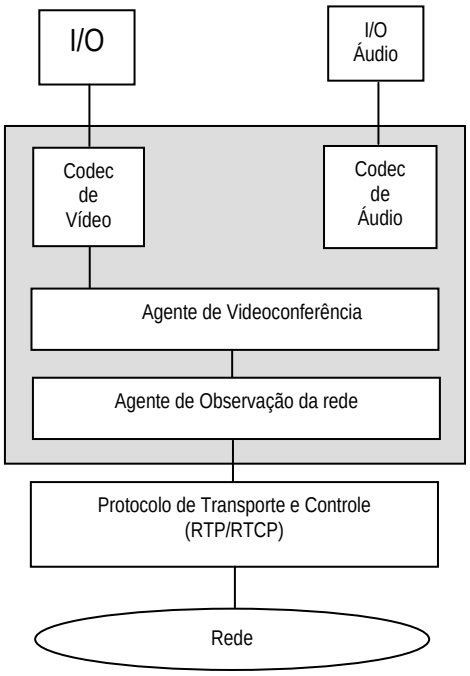

Figura 2.4: Arquitetura do ADAPT

Em relação ao modelo apresentado anteriormente, o processo de adaptação da transmissão das mídias é realizado a partir da interpretação dos dados de controle que são transmitidos entre os softwares clientes participantes da videoconferência. As regras que são definidas no Agente de Videoconferência, passam os parâmetros da transmissão para o receptor, que fica encarregado de se adaptar com os requisitos definidos antes da transmissão (qualidade da imagem por exemplo).

2.5 Um modelo orientado ao perfil da aplicação

O trabalho elaborado por Samuel (CARRION, 2001), propõe um mecanismo que tenha a capacidade de realizar a conformação do tráfego para o usuário. O trabalho foi focalizado no usuário doméstico e em pequenas empresas ou instituições de ensino que busquem o acesso a videoconferência através da Internet ou intranet, utilizando recursos de banda limitados e com uma finalidade definida.

Para que o trabalho atingisse uma maior satisfação do usuário para o sistema que está utilizando, foi definida para cada mídia uma prioridade no momento da conformação do tráfego, assim como os parâmetros que devem ser considerados para que ocorra a conformação. Na conformação pode existir uma diminuição da resolução

por congestionamento e um aumento da resolução por disponibilidade. Ambas as configurações têm somente um objetivo: que as estações envolvidas na conferência se interoperem da melhor forma possível.

Algumas funcionalidades são propostas no modelo definido por Samuel, como o Gerenciador de Perfil de Aplicação, onde a principal função é proporcionar ao usuário a possibilidade da escolha do perfil de transmissão (qualidade da mídia, supressão destas e opções de descarte por exemplo). No modelo elaborado, existe a possibilidade de armazenar, em uma base de dados, os perfis definidos pelos usuários e a utilização destes, no início de novas conferências.

Uma segunda funcionalidade proposta no trabalho é o Gerenciador de Prioridade, sua função é orientar a forma adequada de realizar a adaptação de acordo com o perfil escolhido pelo usuário, ou seja, qual a mídia que deve ser descartada se a adaptação não for possível dentro do perfil escolhido.

Conforme o modelo apresentado, tanto uma máquina que utilize uma conexão via ADSL quanto uma máquina com acesso discado, pode comunicar-se satisfatoriamente. O que ocorre, é que a segunda estação deverá gerar mais solicitações de conformação de tráfego do que a primeira, fazendo com que a estação que use acesso via ADSL receba o tráfego com a mesma limitação da estação com acesso discado.

2.6 Definição do problema

Como visto na introdução deste trabalho, um dos maiores problemas encontrados pelos usuários em videoconferência e colaboração visual de dados é o fato das aplicações consumirem toda ou boa parte dos recursos de uma rede, ocasionando uma grande incidência de problemas como: a queda da conexão dos usuários das conferências, maior processamento nas máquinas dos clientes, impossibilidade de utilizar recursos auxiliares como compartilhamento de aplicações e transferência de arquivos (GIRGENSOHN et al 2002).

O tráfego originado na rede muitas vezes é proveniente de diversos fatores como: a capacidade dos protocolos permitirem que ocorram diversos ajustes na comunicação (adição de novas estações na conferência, por exemplo), a troca de informações entre as camadas de um protocolo (MCS e GCC no T.120), as informações de controle entre as máquinas participantes e da própria conferência (serviços de áudio/vídeo e dados).

Um outro problema que pode ser encontrado nas soluções para videoconferência e colaboração visual de dados, refere-se ao uso de tecnologias proprietárias. Essas tecnologias limitam a utilização da videoconferência e colaboração visual de dados por equipamentos ou soluções de apenas um fabricante. Como conseqüência, sistemas produzidos por um fabricante não conseguem se comunicar com sistemas fabricados por outro fabricante, uma vez que cada produto funciona com tecnologias diferentes. A padronização assegura que os métodos e algoritmos usados para a compressão e descompressão dos dados sejam os mesmos, assim como os protocolos de comunicação. Portanto, quando o sistema estiver baseado em um padrão, qualquer equipamento ou software deverá funcionar neste sistema.

A vantagem de termos equipamentos e hardwares que interoperam usando padrões definidos, dá-nos a certeza de que em uma conferência não tenhamos problemas de incompatibilidade entre os sistemas. Mas o que ocorre atualmente é que aplicações que usam tanto o protocolo H.323, como o T.120 acabam esbarrando em problemas como o

valor de licenças para o uso nos equipamentos, o que acaba criando impecilhos para que as soluções para videoconferência e colaboração visual de dados não tenham um crescimento maior na utilização dentro da comunidade da Internet.

Existem problemas relacionados na colaboração visual de dados (quando é utilizado por exemplo, o serviço de transferência de arquivos - T.127). A utilização da transferência de arquivos provoca alguns problemas como: a queda da conexão dos usuários da conferência (grande volume de tráfego na rede) e o aumento de informações que devem ser processadas pelas máquinas dos clientes, o que em muitas ocasiões acaba sobrecarregando o processador em máquinas de usuários que não possuem os recursos mais sofisticados.

Assim, o problema a ser solucionado nesta dissertação é a diminuição da complexibilidade na implementação das chamadas e primitivas definidas na recomendação T.120, ocasionando, assim, um número menor de informações trafegando pela rede e uma quantidade menor de informações a serem processadas pelas máquinas dos usuários. A solução do problema será apresentada através de uma aplicação de transferência de arquivos sobre o T.120, com características minimalistas. Esta aplicação também deve solucionar questões como o custo de licenças de aplicativos colaborativos e não deve perder a característica de estar baseada em uma padronização e ser utilizada na Internet.

O capítulo seguinte apresenta a primeira contribuição desta dissertação: a apresentação do modelo minimalista para transferência de arquivos e os comentários sobre as simplificações propostas.

3 MODELO MINIMALISTA PARA TRANSFERÊNCIA DE ARQUIVOS

No capítulo anterior, foram identificados os padrões e protocolos utilizados nos serviços de videoconferência e colaboração visual de dados. Também foi objetivo de estudo a apresentação de alguns trabalhos que foram desenvolvidos com a finalidade de diminuir os custos envolvidos na videoconferência. Esses custos estão relacionados com: o grande número de chamadas existentes entre as máquinas dos clientes que ocasionam tráfego na rede, o processamento das informações nas máquinas dos usuários, o tráfego proveniente da própria videoconferência e colaboração visual de dados e as questões de soluções padronizadas e custos agregados com licenças.

O modelo minimalista de que trata este trabalho, busca apresentar algumas alternativas para a solução de alguns problemas comentados anteriormente através da minimalização ou simplificação que pode ser feita nos protocolos T.127, T.124 e T.125. As simplificações propostas, visam apresentar por sua vez, um modelo que possa fazer a transferência de arquivos no mesmo molde do que é realizado no T.127.

Inicialmente o capítulo apresenta simplificações nos protocolos T.124 e T.125. A seguir, são apresentadas as otimizações que podem ser realizadas sobre as chamadas ou primitivas de serviços que são inicializados mas não são usados em colaboração visual de dados com T.127. Por fim, questões de simplificações são propostas exclusivamente para o ambiente T.127.

3.1 Simplificações no nível do MCS e do GCC

Como descrito no capítulo anterior, a principal função do GCC é a definição de um conjunto de serviços para o estabelecimento e o gerenciamento de conferências multiponto, desta forma, é possível consultar um determinado MCU e encontrar uma conferência específica. Já o MCS, procura trabalhar com a resolução de problemas de comunicação multiponto e ponto-a-ponto entre aplicações.

Dentro da recomendação T.120, é considerável o número de trocas de informações entre o MCS e GCC, conseqüentemente, o alto número de chamadas e primitivas, ocasiona mais tráfego na rede e uma sobrecarga maior de processamento na máquina do usuário. Desta forma, é apresentada, a seguir, as simplificações propostas nas camadas GCC e MCS, para a criação do modelo *lite* do T.127:

3.1.1 Primitivas relacionadas ao Gerenciamento de Domínio

- MCS-Connect-Additional: A primitiva MCS-ConnectAdditional pertence à unidade funcional que trata do Gerenciamento de Domínio. O início de uma conexão no MCS ocorre através da primitiva MCS-Connect-Initial, onde alguns parâmetros são negociados como: capacidades, usuários e os canais a serem utilizados na conferência.

- A primitiva MCS-Connect-Additional trata de criar uma nova conexão adicional com algumas requisições não negociadas, como prioridades, direitos de acesso e canais. Em determinadas ocasiões, ocorrem pedidos de conexões adicionais, o que acaba influenciando no desempenho da comunicação entre as camadas MCS e GCC (maior troca de informações entre elas e processamento) e as estações participantes da conferência.

- O fato de não implementar uma chamada MCS-Connect-Additional vem de limitar o número de interações entra as camadas MCS e GCC, possibilitando um menor processamento e também um número reduzido de trocas de dados; não necessitando por sua vez, funções ou procedimentos para iniciar a chamada MCS-ConnectAdditional e nem uma função para responder essa solicitação. As mesmas negociações que deveriam ser realizadas pela chamada MCS-ConnectAdditional poderiam ser realizadas em um processo inicial, através da abertura da conexão, pela chamada MCS-Connect Initial, onde esta deveria negociar na fase inicial todos os recursos para o estabelecimento de uma conexão e suas prioridades, não necessitando assim, de intervenções de solicitações no andamento da conferência.

- MCS-ConnectResult: Na primitiva MCS-ConnectResult, pertencente à unidade funcional, que trata do Gerenciamento de Domínio, pode ser abstraída do modelo *lite* . O principal motivo é que a primitiva em questão, trata de um retorno do MCS-ConnectAdditional, que não deve ser implantado.

- MCS-PlumbDomainIndication: A primitiva MCS-PlumbDomainIndication é encarregada de fazer a comunicação com o T.124 e T.123, a fim de informar que um novo ciclo de conexão foi estabelecido. Conseqüentemente, as camadas GCC e MCS deveriam se adaptar trocando dados como domínio, prioridades e capacidades de transmissão. Como o processo de criação de novas conexões não será adotado no andamento de uma conferência, mas somente no início destas, esta primitiva não precisa ser tratada pelas camadas GCC e MCS no modelo *lite*.

- MCS-ErectDomainRequest: O MCS-ErectDomainRequest é encarregado de avisar ao provedor MCS sobre algumas mudanças na execução do

processamento como: um aumento no número de chamadas, autenticação de usuários, novas prioridades e direitos de acesso. A cada nova conexão criada, um novo provedor MCS deve ser associado. Os provedores MCS possuem uma hierarquia e são responsáveis pelo andamento da conferência. O primeiro provedor foi criado e inicializado no início da conferência e está situado no topo da hierarquia, recebendo o nome de provedor de topo (MCS Top Provider). A cada nova conexão, deve ser informado ao provedor de topo a inclusão de uma nova conexão. Esse aviso é realizado através do MCS-ErectDomainRequest. A inclusão de novos provedores não será realizada no andamento da conexão, evitando assim o aumento no número de chamadas circulando na rede e a diminuição do processamento na máquina do usuário provocado pela criação de conexões adicionais.

- MCS-MergeChannelsRequest: O MCS-MergeChannelsRequest é uma solicitação para realização de um descarte de canais com as mesmas características (utilização do mesmo canal estático, número de identificação de usuário idêntico, igual prioridade). A solicitação de descarte ocorre quando um usuário cria novas conexões em uma conferência. Com o objetivo de evitar que exista um provedor MCS para cada conexão ou canal idêntico, deve ser estabelecido um critério para descarte destes. O fato de não criarmos novas conexões acaba fazendo com que o MCS-MergeChannelsRequest não seja implementado e não sejam necessários, portanto, os mecanismos que fiquem monitorando constantemente a ocorrência de canais com as mesmas características.

- MCS-MergeChannelsConfirm: A primitiva MCS-MergeChannelsConfirm é uma complementação da MCS-MergeChannelsRequest. Como não haverá tratamento para a primitiva anterior, não será necessário a implantação do MCS-MergeChannelsConfirm.

- MCS-PurgeChannelsIndication: O MCS-PurgeChannelsIndication é uma indicação que é feita para as máquinas da conferência informando a ocorrência de canais com os mesmos atributos. Na presença de novas conexões, um provedor de topo é responsável pelo controle dos atributos de todos os canais como: direitos de acesso, permissões e capacidade do canal de dados a ser utilizado. Porém, existem canais que quando são criados, costumam usar determinados valores padrões, como por exemplo, usar sempre a mesma taxa do canal de dados. Como algumas solicitações acabam tornando-se comuns, o provedor de topo pode indicar que os demais provedores da hierarquia subordinados a ele, possam ser responsáveis pelo gerenciamento de alguns atributos mais comuns. Esse fato pode facilitar ou delegar poderes para os provedores subordinados, mas acarreta um grande fluxo de informações de controle na rede e um maior custo de processamento nas máquinas para o atendimento destas chamadas. É aconselhável que o provedor de topo continue atendendo a todas as requisições mais solicitadas por uma primitiva.

- MCS-MergeTokensRequest: A primitiva MCS-MergeTokensRequest é uma solicitação para unificação de tokens. A unificação ocorre quando canais possuem atributos em comum como: identificação de usuário, permissões, capacidade do canal de dados a ser utilizado; porém tokens ou direitos de acesso (momento em que será utilizado o canal de dados) diferentes. Esses tokens devem ser transformados em somente um direito de acesso, para que o provedor de topo possa gerenciar as conexões ou canais da conferência. O MCS-MergeTokensRequest não será implementado no modelo *lite* T.127, porque não existirão canais para serem unificados. Desta maneira, poderão ocorrer um tráfego menor de mensagens circulando na rede e um menor processamento nas máquinas para atender as camadas MCS e GCC.

- MCS-MergeTokensConfirm: O MCS-MergeTokensConfirm é uma resposta que é enviada pela camada MCS para a máquina que originou a chamada MCS-MergeTokensRequest, informando que possui conhecimento sobre o pedido de unificação de tokens. Como a primitiva anterior não será implantada, a MCS-MergeTokensConfirm por se tratar de uma resposta ao MCS-MergeTokensRequest, também não será implantado.

- MCS-PurgeTokensIndication: Quando ocorre um MCS-MergeTokensConfirm, é gerada uma primitiva MCS-PurgeTokensIndication para o provedor de topo, informando a possibilidade de unificação de tokens. A primitiva MCS-PurgeTokensIndication não será implantada por estar diretamente relacionada com as duas primitivas comentadas anteriormente.

- MCS-RejectUltimatum: A primitiva MCS-RejectUltimatum é gerada para informar algum erro em algum PDU entregue ou no protocolo. Essa chamada só tem a finalidade de informar o erro e não ativa ou se comunica com outra primitiva de tratamento de erro. Portanto, toda a função de gerar essa primitiva e de recebê-la, acaba gerando mais dados trafegando na rede e um custo de processamento no atendimento dessas mensagens nas máquinas dos usuários. Por tratar-se somente de uma primitiva de informação, adota-se pela não implantação do MCS- RejectUltimatum.

3.1.2 As primitivas relacionadas ao Gerenciamento de Canal

Segundo a norma T.120 (ITU, 1995a), as chamadas relacionadas ao Gerenciamento de Canal que devem obrigatoriamente constar em um nodo *T.127 lite* são: MCS-Channel-Join request, MCS-Channel-Join confirm e MCS-Channel-Leave request. A norma comenta ainda que as chamadas pertinentes ao Gerenciamento de Canal que não necessariamente precisam fazer parte de um nodo T.127 são:

- MCS-CHANNEL-CONVENE request: Primitiva que é enviada para o Provedor de Topo de MCS para solicitar que diferentes canais privativos de dados possam ser gerenciados por um mesmo usuário. O processo para que

um usuário possa gerenciar diferentes canais privativos gera um grande número de mensagens ou chamadas trafegando na rede, o que também acarreta um custo de processamento nas máquinas dos usuários. Como não será permitido que um usuário gerencie diferentes canais privativos, a primitiva em questão não será implantada.

- MCS-CHANNEL-CONVENE confirm: A chamada em questão trata de um retorno ao MCS-CHANNEL-CONVENE request, que não será implantado.

- MCS-CHANNEL-DISBAND request: A primitiva MCS-CHANNEL-DISBAND request procura desfazer a operação relacionada ao MCS-CHANNEL-CONVENE request. Como a primitiva MCS-CHANNEL-CONVENE request não será implementada, a chamada em questão não será tratada.

- MCS-CHANNEL-DISBAND indication: Primitiva que indica que a chamada MCS-CHANNEL-DISBAND request deve ser atendida.

- MCS-CHANNEL-ADMIT request: Chamada que é enviada ao Provedor de Topo de MCS solicitando a admissão de um usuário para mais de um canal privativo. Como um usuário não poderá gerenciar mais de um canal privativo no nodo T.127 *lite* , essa primitiva ou chamada não deverá ser implantada.

- MCS-CHANNEL-ADMIT indication: O MCS-CHANNEL-ADMIT indication é uma primitiva que informa a camada MCS, sobre a necessidade de atender ou responder a primitiva MCS-CHANNEL-ADMIT request. Por tratar-se de um retorno à primitiva anterior, o MCS-CHANNEL-ADMIT indication não será implantado.

- MCS-CHANNEL-EXPEL request: Trata-se de uma primitiva que solicita informações ao Provedor de Topo de MCS sobre quais os usuários não foram aceitos para gerenciar mais de um canal privativo. Como a adição de mais de um canal privativo para um mesmo usuário não será tratado, a primitiva em questão não deve ser implantada.

- MCS-CHANNEL-EXPEL indication: A chamada em questão, é uma indicação para a camada MCS que a primitiva MCS-CHANNEL-EXPEL request deve ser atendida. A primitiva em questão também não deverá ser implantada.

3.1.3 As primitivas relacionadas ao Gerenciamento de Token

Para a norma T.120 (ITU, 1995a), as seguintes primitivas que estão relacionadas ao Gerenciamento de Token, não necessitam ser implementadas em um nodo T.127 *lite*:

- MCS-TOKEN-GRAB request: Primitiva que é enviada para o Provedor de Topo de MCS, solicitando que um determinado usuário possa utilizar mais de um token. Como o número de informações que trafegam na rede e que são processadas pela máquina do usuário tende a aumentar na solicitação de tokens "adicionais", a primitiva em questão não será implantada no nodo T.127 *lite* .

- MCS-TOKEN-GRAB confirm: A chamada ou primitiva em questão trata de informar a camada MCS que a primitiva MCS-TOKEN-GRAB request dever ser atendida e por tratar-se de um retorno à primitiva anterior; não será implantada.

- MCS-TOKEN-INHIBIT request: A primitiva em questão é enviada ao Provedor de Topo de MCS para solicitar que o token "adicional", passe para o estado de "utilizado" para as demais estações. Passando o direito de uso do token para a estação que enviou o MCS-TOKEN-GRAB request. Como não serão implantados tokens "adicionais", a primitiva MCS-TOKEN-INHIBIT request não será implantada também.

- MCS-TOKEN-INHIBIT confirm: Primitiva que trata em dar uma resposta ao MCS-TOKEN-INHIBIT request. Por estar relacionada à primitiva anterior, não será implantada para o nodo T.127 *lite* .

- MCS-TOKEN-GIVE request: A primitiva em questão é enviada para o Provedor de Topo de MCS solicitando que o token "adicional" capturado, tenha o seu estado alterado para "utilizado" pela primitiva que solicitou o MCS-TOKEN-GRAB request. Como não serão utilizados tokens "adicionais" para cada usuário, esta primitiva não será implantada.

- MCS-TOKEN-GIVE indication: A chamada MCS-TOKEN-GIVE indication é uma indicação que a primitiva anterior deve ser atendida e retornada uma resposta com o MCS-TOKEN-GIVE response.

- MCS-TOKEN-GIVE response: O MCS-TOKEN-GIVE response é uma resposta à solicitação de MCS-TOKEN-GIVE request.

- MCS-TOKEN-GIVE confirm: O MCS-TOKEN-GIVE confirm, indica que a primitiva MCS-TOKEN-GIVE request foi respondida. A primitiva em questão, também não será implantada.

- MCS-TOKEN-PLEASE request: Nessa primitiva, é feita uma solicitação ao Provedor de Topo de MCS para avisar sobre a ocorrência de tokens "adicionais"

a serem capturados pelas estações. Como tokens adicionais não serão utilizados nos nodos T.127 *lite* , a primitiva em questão não deverá ser implantada.

- MCS-TOKEN-PLEASE indication: Primitiva que informa ao Provedor de Topo de MCS, que existe uma primitiva MCS-TOKEN-PLEASE request a ser atendida.

- MCS-TOKEN-RELEASE request: O MCS-TOKEN-RELEASE request é uma solicitação que é feita ao Provedor de Topo de MCS, para saber quais os próximos tokens poderão ser utilizados. Desta forma, estações podem utilizar os tokens que desejarem assim que estes ficarem disponíveis.

- MCS-TOKEN-RELEASE confirm: A primitiva em questão indica o recebimento pelas estações das informações pertinentes aos tokens que podem futuramente estar disponíveis para serem utilizados.

- MCS-TOKEN-TEST request: O MCS-TOKEN-TEST request é uma solicitação para descobrir a identificação do usuário que está utilizando o token "adicional". Desta forma, todas as mensagens que estão relacionadas à adição de tokens, geram muito tráfego na rede, fato que acaba diminuindo de certa forma a performance desta. A primitiva em questão também não deverá ser implantada para os nodos do T.127.

- MCS-TOKEN-TEST confirm: O MCS-TOKEN-TEST confirm procura confirmar o recebimento das informações solicitadas pela primitiva comentada anteriormente.

3.1.4 Primitivas relacionadas ao Estabelecimento da Conferência e Término

As primitivas que estão relacionadas ao estabelecimento da conferência e também ao término desta, que devem obrigatoriamente ser implementadas por um nodo *T.127 lite* são:

- GCC-CONFERENCE-CREATE request (primitiva que é enviada pelo GCC para solicitar a criação da conferência).

- GCC-CONFERENCE-CREATE indication (primitiva responsável pelo atendimento da primitiva anterior).

- GCC-CONFERENCE-CREATE response (primitiva que responde o recebimento do GCC-CONFERENCE-CREATE request).

- GCC-CONFERENCE-CREATE confirm (esta chamada é responsável por confirmar a criação da conferência).

- GCC-CONFERENCE-QUERY request (chamada que solicita a negociação de parâmetros da conferência, como: endereço de destino e origem – porta, dados do usuário, parâmetros de qualidade de serviço entre outros).

- GCC-CONFERENCE-QUERY indication (esta primitiva é responsável em sinalizar o atendimento da primitiva anterior).

- GCC-CONFERENCE-QUERY response (a chamada em questão, trata de responder a solicitação da primitiva GCC-CONFERENCE-QUERY request).

- GCC-CONFERENCE-QUERY confirm (primitiva encarregada de sinalizar ou fechar a negociação sobre os parâmetros negociados para a conferência).

- GCC-CONFERENCE-JOIN request (esta primitiva faz o pedido para conectar o usuário na conferência)

- GCC-CONFERENCE-JOIN indication (a primitiva em questão indica o recebimento da chamada anterior para que o GCC possa atender).

- GCC-CONFERENCE-JOIN response (trata-se de um retorno ou resposta da primitiva GCC-CONFERENCE-JOIN request)
- GCC-CONFERENCE-JOIN confirm (confirmação sobre o recebimento da chamada anterior)
- GCC-CONFERENCE-INVITE indication (chamada que faz o convite para que um usuário faça parte da conferência)

- GCC-CONFERENCE-INVITE response (a primitiva em questão, responde com o aceite ou não do usuário, para ingressar na conferência)

- GCC-CONFERENCE-DISCONNECT request (solicitação para que a conexão ou conferência seja finalizada)

- GCC-CONFERENCE-DISCONNECT indication (primitiva que indica o recebimento do pedido de desconexão da conferência).

- GCC-CONFERENCE-DISCONNECT confirm (esta primitiva responde ao pedido de desconexão)

- GCC-CONFERENCE-TERMINATE indication (primitiva que indica o término da conferência).

- GCC-CONFERENCE-EJECT-USER indication (indica que os parâmetros negociados na criação da conexão e no andamento desta devem ser repassados para o provedor de topo de GCC para não serem mais utilizados).

As primitivas que, segundo a norma T.120 (ITU, 1995a) não necessitam ser implantadas no estabelecimento da conferência e também no término, são citados a seguir:

- GCC-CONFERENCE-ADD request: o GCC-CONFERENCE-ADD request faz um pedido para o MCU, solicitando a inclusão na conferência da estação que desejar fazer parte da conferência. Esse processo deverá ser realizado no início da comunicação e não durante esta. Desta forma, a primitiva em questão não necessita ser implementada no T.127 *lite*.

- GCC-CONFERENCE-ADD indication: a primitiva em questão é uma indicação que o GCC-CONFERENCE-ADD request deve ser atendido pela camada GCC.

- GCC-CONFERENCE-ADD response: na chamada em questão, a camada GCC envia uma resposta para a máquina que enviou o GCC-CONFERENCE-ADD request.

- GCC-CONFERENCE-ADD confirm: o GCC-CONFERENCE-ADD confirm, é uma primitiva que tem a finalidade de informar ao GCC, que a primitiva GCC-CONFERENCE-ADD request foi respondida.

- GCC-CONFERENCE-LOCK request: a primitiva em questão, é enviada ao MCU para solicitar o "fechamento" da conferência. Desta forma, nenhuma outra estação pode ingressar na conferência e conseqüentemente, nem será convidada a fazer parte desta. Como não será permitido que novas estações façam parte da conferência em andamento, não é necessária a preocupação por parte da camada GCC em colocar a conferência no estado de "fechado", diminuindo com isso, muito o tráfego de informações que circulam na rede.

- GCC-CONFERENCE-LOCK indication: o GCC-CONFERENCE-LOCK indication é um aviso para que a primitiva de GCC-CONFERENCE-LOCK request recebida seja atendida pelo GCC.

- GCC-CONFERENCE-LOCK response: para a primitiva GCC-CONFERENCE-LOCK request, é enviado uma resposta através da primitiva GCC-CONFERENCE-LOCK response.

- GCC-CONFERENCE-LOCK confirm: a primitiva em questão informa que a solicitação GCC-CONFERENCE-LOCK request foi respondida.

- GCC-CONFERENCE-UNLOCK request: o GCC-CONFERENCE-UNLOCK request, é uma primitiva que é enviada para o MCU, solicitando que a conferência seja "destrancada" e novos usuários possam ser convidados a fazer parte da conferência. Como não será permitido que novos clientes ingressem na conferência em andamento, a chamada em questão não deve também, como as anteriores, ser implantada.

- GCC-CONFERENCE-UNLOCK indication: o GCC-CONFERENCE-UNLOCK indication é uma informação a camada GCC de que a primitiva GCC-CONFERENCE-UNLOCK request deve ser atendida.

- GCC-CONFERENCE-UNLOCK response: na primitiva GCC-CONFERENCE-UNLOCK response é enviada uma resposta para a estação que enviou o GCC-CONFERENCE-UNLOCK request indicando se a conferência vai ser "destrancada" para que novos usuários possam ingressar. Conseqüentemente, a primitiva em questão não será implantada como as anteriores relacionadas ao ingresso ou convite de novos usuários em conferências.

- GCC-CONFERENCE-UNLOCK confirm: a chamada em questão é responsável em informar que a primitiva GCC-CONFERENCE-UNLOCK request foi respondida.

- GCC-CONFERENCE-LOCK-REPORT indication: toda vez que a chamada GCC-CONFERENCE-LOCK request é enviada para o MCU é gerado um relatório aos nodos da conferência indicando que a conferência está em estado de "fechado".

- GCC-CONFERENCE-DISCONNECT request: a primitiva GCC-CONFERENCE-DISCONNECT request é um pedido de desconexão. Dentro de uma conferência, quando uma estação deseja desconectar-se, ela envia um pedido, a estação que recebe este pedido responde. Para o modelo T.127 *lite* , a estação que deseja desconectar-se da conferência, simplesmente encerra sem a necessidade de realizar um pedido. Desta forma, a primitiva em questão não deve ser implementada.

- GCC-CONFERENCE-DISCONNECT indication: o GCC-CONFERENCE-DISCONNECT indication trata de uma indicação de que a primitiva GCC-CONFERENCE-DISCONNECT request deve ser atendida pela camada GCC.

- GCC-CONFERENCE-DISCONNECT confirm: no GCC-CONFERENCE-DISCONNECT confirm, a estação que recebeu o GCC-CONFERENCE-DISCONNECT request informa que recebeu o pedido de desconexão.

- GCC-CONFERENCE-TERMINATE request: a primitiva em questão é enviada para solicitar o término de uma conferência. O fato do nodo *T.127 lite* não implementar esta primitiva, acarreta um número menor de mensagens circulando na rede e um custo menor de processamento, pois as camadas GCCs das máquinas envolvidas não necessitam executar e atender as primitivas relacionadas ao encerramento da conexão. No modelo T.127 *lite* , quando uma estação deseja finalizar uma conferência ela simplesmente encerra esta, sem aviso prévio às demais estações.

- GCC-CONFERENCE-TERMINATE confirm: o GCC-CONFERENCE-TERMINATE confirm trata de uma indicação de que a mensagem GCC-CONFERENCE-TERMINATE request deve ser atendida.

- GCC-CONFERENCE-EJECT-USER request: a primitiva em questão é utilizada pelo Controlador de Nodos a "forçar" uma estação a desconectar-se da conferência. No nodo T.127 *lite* , esta primitiva não é necessária, quando o Controlador de Nodo deseja que uma estação se desconecte da conferência, ele simplesmente encerra a conexão da estação.

- GCC-CONFERENCE-EJECT-USER confirm: o GCC-CONFERENCE-EJECT-USER confirm é uma primitiva que indica que existe uma chamada GCC-CONFERENCE-EJECT-USER request a ser atendida.

- GCC-CONFERENCE-TRANSFER request: no GCC-CONFERENCE-TRANSFER request, o Controlador de Nodos faz uma solicitação para que uma determinada estação encerre sua conexão e una-se a outra conferência. No nodo T.127 *lite* , cada estação tem a possibilidade de finalizar uma conexão e não necessita receber, enviar pedidos de encerramento de conexão e término.

- GCC-CONFERENCE-TRANSFER indication: GCC-CONFERENCE-TRANSFER indication indica que existe uma primitiva GCC-CONFERENCE-TRANSFER request para ser atendida.

- GCC-CONFERENCE-TRANSFER confirm: a primitiva em questão tem a finalidade de indicar o recebimento da primitiva GCC-CONFERENCE-TRANSFER request.

3.1.5 Primitivas relacionadas ao Roteamento da Conferência

As primitivas que segundo a norma T.120 (ITU, 1995a) não necessitam ser implantadas no Roteamento da Conferência, são citados a seguir:

- GCC-CONFERENCE-ROSTER-INQUIRE request: o GCC-CONFERENCE-ROSTER-INQUIRE request é uma solicitação para que o provedor de GCC obtenha uma lista de todas as conexões criadas no andamento da conferência, independente do Provedor de GCC local. O provedor de uma conferência pode obter informações sobre as conexões criadas em uma outra conferência, para que futuramente estas possam interagir. A desvantagem na implementação destas chamadas é o fato de consumirem grande capacidade de processamento das máquinas dos usuários para atender solicitações. Existe também uma grande troca de mensagens na rede pertinentes a esse serviço, o que também acarreta um maior tráfego de informações.

- GCC-CONFERENCE-ROSTER-INQUIRE confirm: esta primitiva indica que foi recebido pela máquina destino, uma chamada de GCC-CONFERENCE-ROSTER-INQUIRE request.

- GCC-APPLICATION-INVOKE request: primitiva que é enviada para solicitar a associação de uma Entidade de Aplicação de Protocolo – APE na conferência. O provedor de topo GCC solicita informações de outras conferências que ocorrem em paralelo. As informações podem, futuramente, ser utilizadas para integrar alguns serviços, como a utilização do canal de dados pelas demais conferências. Ao longo de uma conferência, várias primitivas GCC-APPLICATION-INVOKE request podem ser acionadas e enviadas pela rede. Estas chamadas podem aumentar significativamente o número de mensagens e também o desempenho das máquinas dos usuários, pois serão necessárias rotinas para ativar essas funções, confirmar o recebimento e enviar novamente uma primitiva informando sobre o processamento do GCC-APPLICATION-INVOKE request. Toda a associação da Entidade de Aplicação do Protocolo que venha a ocorrer, poderia ser realizada no início do estabelecimento da conferência ou em determinados casos, nem ocorrer este serviço por ser constatado que muitas vezes essas operações de associação entre as diferentes conferências não ocorrem, restando à primitiva GCC-APPLICATION-INVOKE request a função de informar que existem conferências em paralelo. Conseqüentemente, esta primitiva pode ser desconsiderada na implantação de um nodo T.127 *lite* .

- GCC-APPLICATION-INVOKE indication: chamada que indica para a camada GCC o recebimento da primitiva GCC-APPLICATION-INVOKE request, sendo retornada com uma confirmação de recepção. Como se trata de uma chamada que está relacionada com a anterior, não necessita ser implantada.

- GCC-APPLICATION-INVOKE confirm: esta primitiva tem a finalidade de avisar que a mensagem GCC-APPLICATION-INVOKE request foi realmente recebida e tratada pela camada GCC da máquina de destino. Conseqüentemente a primitiva GCC-APPLICATION-INVOKE confirm não será inserida no nodo T.127 *lite* .

3.1.6 Primitivas relacionadas ao Registro de Aplicação

As primitivas que segundo a norma T.120 (ITU, 1995a) não necessitam ser implantadas no Registro de Aplicação para um nodo *T.127 lite* são citados a seguir. O fato de não serem implantadas justifica-se por todos os processos de registro de aplicação, serem definidos no início da conferência ou conexão.

- GCC-REGISTRY-REGISTER-CHANNEL request: pedido para que o canal a ser utilizado pela aplicação seja registrado. O processo já é realizado na criação da conexão, no canal de dados que será utilizado pelo usuário.

- GCC-REGISTRY-REGISTER-CHANNEL confirm: chamada que indica o recebimento pela camada GCC do GCC-REGISTRY-REGISTER-CHANNEL request.

- GCC-REGISTRY-ASSIGN-TOKEN request: pedido para que um tokem seja referenciado para uma aplicação. Este processo também pode ser realizado no início da conexão, informando ao Provedor de Topo cada tokem associado a cada canal.

- GCC-REGISTRY-ASSIGN-TOKEN confirm: chamada confirmando o recebimento do GCC-REGISTRY-ASSIGN-TOKEN request.

- GCC-REGISTRY-SET-PARAMETER request: trata-se de uma solicitação para que os parâmetros de identificação do usuário, canal e aplicação devem ser atribuídos e registrados. Esta função é definida também no estabelecimento da conexão.

- GCC-REGISTRY-SET-PARAMETER confirm: chamada que confirma o recebimento da primitiva GCC-REGISTRY-SET-PARAMETER request

- GCC-REGISTRY-RETRIEVE-ENTRY request: chamada que faz a solicitação para a revisão dos registros solicitados nas chamadas GCC-REGISTRY-SET-PARAMETER request, ou seja, confirmar os valores antes negociados.

- GCC-REGISTRY-RETRIEVE-ENTRY confirm: a primitiva em questão, indica o recebimento do GCC-REGISTRY-RETRIEVE-ENTRY request.

- GCC-REGISTRY-DELETE-ENTRY request: primitiva que envia uma solicitação para retirada da base de dados dos registros negociados no GCC-REGISTRY-SET-PARAMETER. Os pedidos de retirada e confirmação podem gerar processamento nas máquinas dos clientes e tráfego na rede, não necessitam ser implementados.

- GCC-REGISTRY-DELETE-ENTRY confirm: trata-se de uma confirmação da chamada anterior informando que foi realizada a retirada das bases de dados dos registros negociados.

- GCC-REGISTRY-MONITOR request: solicitação que é enviada para a camada GCC indicando a habilitação ou desabilitação das monitorações relacionadas ao Registro de Aplicação. Esta chamada despende um custo de processamento nas máquinas dos usuários e gera tráfego na rede, pois várias primitivas GCC-REGISTRY-MONITOR request são executadas dentro de uma conferência.

- GCC-REGISTRY-MONITOR indication: primitiva que indica o recebimento de uma chamada GCC-REGISTRY-MONITOR request.

- GCC-REGISTRY-MONITOR confirm: retorno que é passado para a máquina de origem que enviou o GCC-REGISTRY-MONITOR request.

- GCC-REGISTRY-ALLOCATE-HANDLE request: primitiva que é enviada para solicitar a alocação de um cabeçalho para a Entidade de Protocolo de Aplicação. Com isso, a conferência poderá alocar uma aplicação para futuramente ser utilizada. Esta chamada não necessita ser implantada porque cada canal da conferência recebe um atributo – como identificação do usuário, não necessitando a alocação para a aplicação. Este processo é realizado na definição do canal de dados para o usuário.

- GCC-REGISTRY-ALLOCATE-HANDLE confirm: primitiva que confirma o recebimento do GCC-REGISTRY-ALLOCATE-HANDLE request.

3.1.7 Primitivas relacionadas à Condução da Conferência

Sobre as primitivas relacionadas na Condução da Conferência, a norma T.120 (ITU, 1995a) indica que as seguintes primitivas não necessitam ser implantadas para um nodo T.127 *lite* :

- GCC-CONDUCTOR-ASSIGN request: primitiva que envia uma solicitação para examinar se a conferência pode ser conduzida ou não. Esta chamada não necessita ser implementada, pois a garantia da condução da conferência pode ser definida no estabelecimento da conexão, eliminando assim os custos de processamento nas máquinas dos usuários e o aumento do número de dados circulando na rede.

- GCC-CONDUCTOR-ASSIGN indication: primitiva que indica o recebimento da chamada GCC-CONDUCTOR-ASSIGN request.

- GCC-CONDUCTOR-ASSIGN confirm: primitiva que envia um retorno sobre o recebimento da chamada GCC-CONDUCTOR-ASSIGN request.

- GCC-CONDUCTOR-RELEASE request: primitiva que é enviada para solicitar que um determinado nodo faça a condução da conferência. Esta primitiva não é implementada, quem faz a condução da conferência é o nodo que iniciou a conexão.

- GCC-CONDUCTOR-RELEASE indication: a primitiva em questão, indica que o GCC-CONDUCTOR-RELEASE request deverá ser respondido

- GCC-CONDUCTOR-RELEASE confirm: primitiva que confirma o recebimento da chamada GCC-CONDUCTOR-RELEASE request.

- GCC-CONDUCTOR-PLEASE request: primitiva que é enviada ao nodo de destino para solicitar que esse dê permissão de controle da conferência. A alternância no controle da conferência não será implantada. No nodo *T.127 lite* a estação que iniciou a conexão mantém o controle desta.

- GCC-CONDUCTOR-PLEASE indication: primitiva que indica que o nodo de destino deve responder a um GCC-CONDUCTOR-PLEASE request.

- GCC-CONDUCTOR-PLEASE response: a chamada em questão tem a finalidade de responder um GCC-CONDUCTOR-PLEASE request.

- GCC-CONDUCTOR-PLEASE confirm: primitiva que envia uma confirmação sobre a solicitação de GCC-CONDUCTOR-PLEASE request recebida.

- GCC-CONDUCTOR-GIVE request: chamada que é enviada para perguntar se o nodo que recebe esta chamada, deseja realizar a condução da conferência. Existe um custo de processamento na troca de informações referentes à troca da condução de uma conferência, como o nodo que iniciou a conferência é por direito quem a conduz, não é necessário a troca de administradores por ocasionar muitas mensagens na rede e um aumento no processamento nas máquinas dos usuários.

- GCC-CONDUCTOR-GIVE indication: primitiva que indica que deve ser enviada uma resposta ao nodo que enviou um GCC-CONDUCTOR-GIVE request.

- GCC-CONDUCTOR-GIVE response: retorno da primitiva GCC-CONDUCTOR-GIVE request.

- GCC-CONDUCTOR-GIVE confirm: primitiva que informa o recebimento do GCC-CONDUCTOR-GIVE request.

- GCC-CONDUCTOR-INQUIRE request: primitiva que é enviada por um controlador de nodo para descobrir qual o nodo é o condutor da conferência. Esta primitiva não necessita de implementação, o nodo controlador é aquele que inicia a conexão da conferência.

- GCC-CONDUCTOR-INQUIRE confirm: confirmação por parte dos nodos da conferência sobre o recebimento do GCC-CONDUCTOR-INQUIRE request.

- GCC-CONDUCTOR-PERMISSION-ASK request: primitiva que é enviada solicitando para um determinado nodo exercer as funções de condução da conferência. Como o nodo que inicia a conexão é quem realmente faz a condução, a primitiva em questão não será implementada.

- GCC-CONDUCTOR-PERMISSION-ASK indication: a primitiva em questão, informa que existe uma primitiva GCC-CONDUCTOR-PERMISSION-ASK request para ser atendida.

- GCC-CONDUCTOR-PERMISSION-ASK confirm: confirmação de que a primitiva GCC-CONDUCTOR-PERMISSION-ASK request foi realmente respondida.
- GCC-CONDUCTOR-PERMISSION-GRANT request: primitiva que é enviada para solicitar a renovação dos direitos de condução da conferência por um nodo. Esta primitiva não necessita ser implementada, uma vez definido o nodo que irá gerenciar a conferência, este permanece com os mesmos direitos do início da conexão.

- GCC-CONDUCTOR-PERMISSION-GRANT indication: a primitiva em questão indica que deve ser respondido o GCC-CONDUCTOR-PERMISSION-GRANT request.

- GCC-CONDUCTOR-PERMISSION-GRANT confirm: primitiva que indica que a chamada GCC-CONDUCTOR-PERMISSION-GRANT request foi respondida.

3.1.8 Primitivas relacionadas a Funções Variadas

Sobre as primitivas relacionadas a Funções Variadas, a norma T.120 (ITU, 1995a) indica que as seguintes primitivas não são obrigatórias para um nodo T.127 *lite* :

- GCC-CONFERENCE-TIME-REMAINING request: a chamada em questão, é enviada pelo Controlador de Nodo para os nodos participantes da conferência, solicitando o tempo em que cada estação estiver conectada na conferência. A chamada pode não ser implementada, pois o número de informação que é gerado para as estações na rede é grande e a quantidade de informação para processar também é considerável.

- GCC-CONFERENCE-TIME-REMAINING indication: chamada que indica que a primitiva GCC-CONFERENCE-TIME-REMAINING request deve ser atendida.

- GCC-CONFERENCE-TIME-REMAINING confirm: retorno que deve ser dado à máquina que enviou o GCC-CONFERENCE-TIME-REMAINING request.

- GCC-CONFERENCE-TIME-INQUIRE request: primitiva que pode ser enviada por qualquer nodo para descobrir quanto tempo os nodos estão conectados na conferência. Para a primitiva em questão, não é necessária a sua implementação devido ao número de informações que podem ser geradas na rede e a quantidade de dados a ser processados na máquina do cliente.

- GCC-CONFERENCE-TIME-INQUIRE indication: a primitiva em questão indica que o GCC-CONFERENCE-TIME-INQUIRE request deve ser atendido. Como a primitiva anteriormente comentada na será implementada, esta por ser uma seqüência da anterior também não deve ser implantada.

- GCC-CONFERENCE-TIME-INQUIRE confirm: retorno para a máquina que enviou o GCC-CONFERENCE-TIME-INQUIRE request.

- GCC-CONFERENCE-EXTEND request: primitiva que solicita um tempo maior para que determinada estação permaneça na conferência. Esta chamada gera um maior tráfego na rede e um custo de processamento das mensagens. Quando o número de estações que fazem parte de uma conferência é alto, é grande a quantidade de informações solicitando a permanência das estações na rede.

- GCC-CONFERENCE-EXTEND indication: a primitiva em questão tem por objetivo receber o GCC-CONFERENCE-EXTEND request.

- GCC-CONFERENCE-EXTEND confirm: retorno que é dado para a estação que enviou o GCC-CONFERENCE-EXTEND request.

- GCC-CONFERENCE-ASSISTANCE request: a chamada em questão, trata de uma solicitação que uma estação pode fazer para receber maiores informações a fim de continuar fazendo parte da conferência. O retorno desta solicitação não é previsto pela norma T.120 e por este motivo, não será implantada.

- GCC-CONFERENCE-ASSISTANCE indication: indica que a primitiva GCC-CONFERENCE-ASSISTANCE request deve ser atendida. Como a norma não prevê a primitiva anterior, o GCC-CONFERENCE-ASSISTANCE indication também não deverá ser implantado.

- GCC-CONFERENCE-ASSISTANCE confirm: retorno que é dado à máquina que enviou o GCC-CONFERENCE-ASSISTANCE indication.

- GCC-TEXT-MESSAGE request: solicitação que é enviada para as estações participantes da conferência a fim de enviar mensagens que não foram especificadas durante o estabelecimento da conexão. Estas mensagens são enviadas por broadcast para todas as máquinas e devem ser exibidas por algum meio não especificado na recomendação T.120. Como existem outras formas de comunicação de mensagens de texto definidos na norma T.120, esta primitiva não será implantada.

- GCC-TEXT-MESSAGE indication: indicação que a primitiva GCC-TEXT-MESSAGE request deve ser atendida. Para um nodo T.127 *lite*, esta primitiva não deve ser implantada.

- GCC-TEXT-MESSAGE confirm: como as duas primitivas comentadas anteriormente não serão implantadas, a primitiva em questão também não deve ser implementada por tratar-se de um retorno das duas anteriores.

3.2 Simplificações nas chamadas não pertinentes ao serviço T.127 utilizado

No âmbito de uma videoconferência e colaboração visual de dados, é comum a solicitação de recursos auxiliares, como a transferência de arquivos. Um dos problemas encontrados em monitorações realizadas sobre tráfegos de conferências com T.127, é a presença de chamadas não pertinentes ao serviço utilizado no momento. Essas chamadas foram encontradas tanto na comunicação ponto-a-ponto entre duas máquinas realizando uma transferência de arquivos, como em uma comunicação multiponto com presença de um MCU.

3.2.1 Simplificação na primitiva: T.128 Legacy AS Section: RequestActivePdu

A primitiva RequestActivePdu, faz uma solicitação para as estações envolvidas na conferência a fim de ter conhecimento das capacidades que podem ser negociadas na conferência e se existem outros nodos ativos nesta.

Quando a estação envia um RequestActivePdu e não recebe nenhum retorno, o próximo passo a ser tomado pela estação que deseja compartilhar um aplicativo é enviar um DemandActivePDU. Desta forma, a estação já está tomando o passo inicial para o compartilhamento de aplicações, que é a preparação do canal e a definição das capacidades a colaborar.

O fato de encontrarmos primitivas do serviço T.128 em tráfegos de uma conferência com T.127 vem a constatar que as camadas T.123, T.124 e T.125 da arquitetura T.120 buscam “preparar” todo o ambiente colaborativo para uma possível colaboração visual de dados, com compartilhamento de aplicações.

Na figura 3.1, é apresentada um trecho de uma captura de pacotes de uma conferência com T.127. Na monitoração, foi detectada a primitiva RequestActivePdu que pode comprometer o desempenho de uma conferência com transferência de arquivos (T.127). Nesta monitoração, pode ser visualizada além da chamada do T.125, a

presença de mensagens T.128 (compartilhamento de aplicações) não solicitadas. Estas chamadas geram um processamento desnecessário tanto na máquina como na rede, consumindo recursos destinados para aplicações que foram legitimamente acionadas pelo usuário.

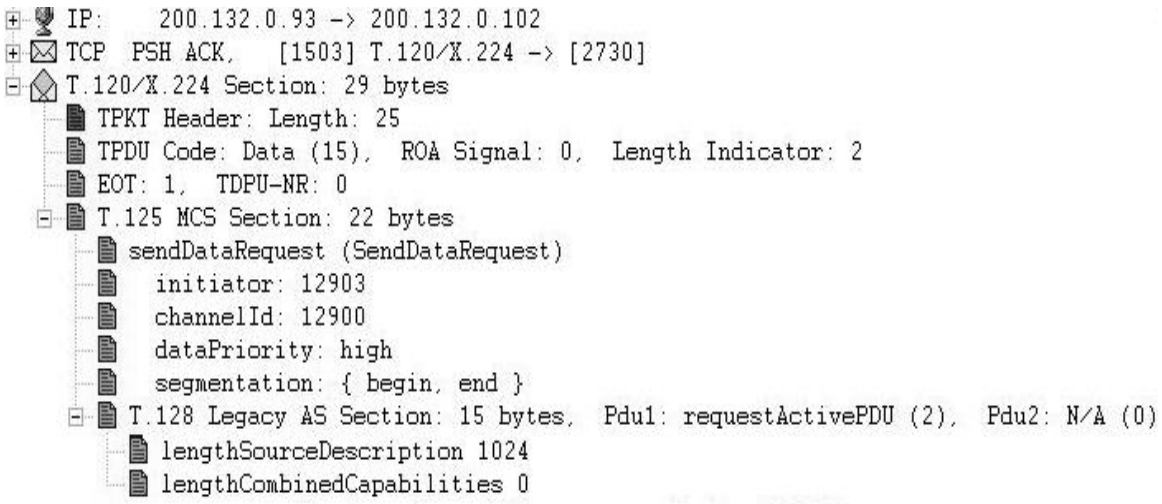

Figura 3.1: Primitiva RequestActivePdu referente ao serviço T.128

Como a primitiva RequestActivePdu está relacionada especificamente com o compartilhamento de aplicações, não é pertinente que a primitiva em questão faça parte do nodo T.127 *lite* . Desta forma, o fato de não implantar o RequestActivePdu vem a diminuir o número de mensagens circulando na rede e também o custo de processamento das mensagens nas máquinas dos usuários participantes da conferência.

3.2.2 Simplificação na primitiva: T.128 Legacy AS Section: pduTypeFlow

A primitiva pduTypeFlow é responsável pelo início do processo de transmissão ou compartilhamento da aplicação a ser escolhida pelo usuário. Para a primitiva em questão existem atributos que podem definir diferentes tipos de fluxos.

O fluxo de informações de teste é um exemplo de informações que circulam através do atributo flowTest para indicar se o canal ou conexão estão aptos ao compartilhamento de dados. Já o flowStop é um atributo que pode indicar que os dados a serem enviados no pduTypeFlow não podem ser enviados no momento. Este tipo de impedimento pode, muitas vezes, estar relacionado com a capacidade do canal, priorizações ou até mesmo com o fato de uma estação receptora de dados não aceitar o compartilhamento.

Por fim, o atributo FlowResponsePdu da primitiva pduTypeFlow indica o tipo de informação a ser negociado entre as estações, como por exemplo, o fluxo de dados ou de teste, antes do compartilhamento.

A figura 3.2 apresenta o trecho de um pacote capturado em uma conferência onde o T.127 é utilizado. Na figura a ser apresentada, existem dados referentes à camada MCS ou T.125 e uma comunicação desta com o T.128 para troca de algumas informações.

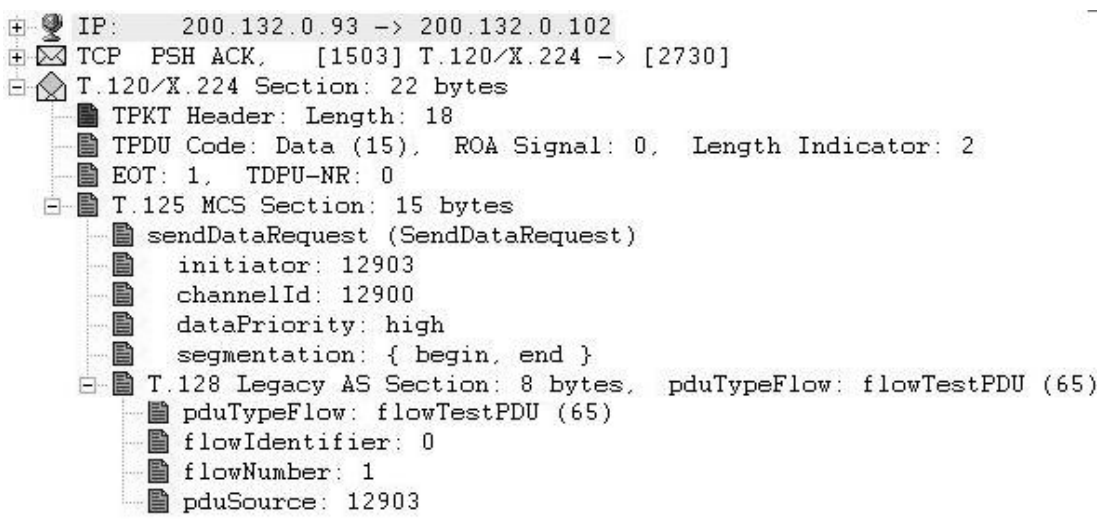

Figura 3.2: Primitiva pduTypeFlow referente ao T.128

A primitiva em questão também procura fazer uma interação com as camadas T.123, T124 e T.125 da arquitetura T.120 para, de certa forma, "preparar" a conferência para uma futura colaboração de dados com compartilhamento de aplicações. Como o serviço que foi legitimamente acionado pelo usuário é o T.127, mensagens pertinentes ao T.128 acabam gerando um número maior de informações circulando na rede e um custo para serem processadas pelas máquinas dos usuários. O fato de não implementar o pduTypeFlow, colabora para um melhor desempenho da rede e das máquinas envolvidas na conferência.

3.2.3 Simplificação na primitiva: T.128 Legacy AS Section: ShareControlHeader

A primitiva em questão, é responsável em passar informações sobre o cabeçalho das mensagens ou dados a serem compartilhados. As informações que são negociadas estão relacionadas à versão do protocolo, o tipo de PDU a ser transmitido (dados ou controle) e a origem do PDU (máquina e porta).

Ao negociar informações no ShareControlHeader, as camadas T.123, T.124 e T.125 procuram tomar conhecimento sobre possíveis trocas de mensagens relacionadas ao compartilhamento de aplicações. Desta forma, as camadas do T.120 envolvidas em uma conferência podem estar aptas futuramente a utilizarem os demais serviços que não foram acionados pelo usuário na conferência em andamento, como compartilhamento de aplicações ou chat.

Um trecho de um pacote capturado em uma conferência com T.127 é ilustrado na figura 3.3. Na ilustração a ser apresentada a seguir, foram feitas algumas abstrações nas informações que aparecem no pacote capturado. Desta forma as informações ficam restritas somente ao T.120.

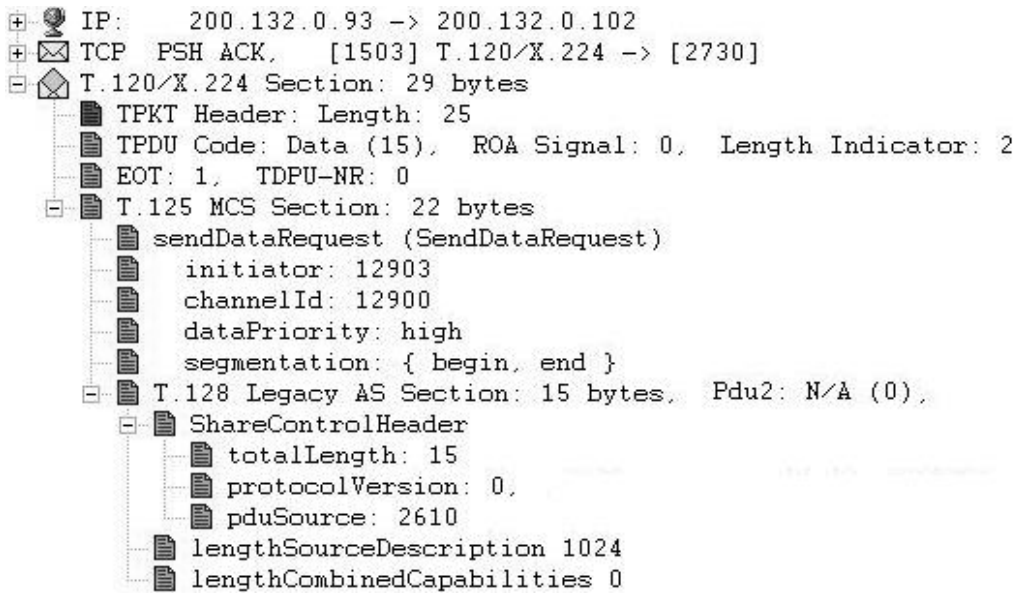

Figura 3.3: Primitiva ShareControlHeader referente ao T.128

A primitiva em questão também busca informar e "preparar" as camadas da arquitetura T.120 sobre uma possível utilização no futuro do serviço T.128. As mensagens de ShareControlHeader que são enviadas através da rede, geram tráfego e um custo de processamento nas máquinas que devem receber e enviar a primitiva. O fato de não implementar o ShareControlHeader vem de acordo com a necessidade de diminuir o tráfego na rede e o processamento das máquinas.

3.2.4 Simplificações na primitiva: T.128 Legacy AS Section: FlowMarker

Na primitiva FlowMarker são realizadas trocas de informações entre as camadas do T.120 com o objetivo de marcar a direção do fluxo, ou seja, qual a estação inicia o processo de compartilhamento e envio especificamente dos dados da aplicação a ser utilizada.

Com a definição da estação que deve iniciar o processo de compartilhamento de aplicações, o T.120 faz com que a estação defina seus atributos e configurações, como por exemplo, a adaptação do canal de dados entre as máquinas da conferência para que as mesmas possam se adaptar à quantidade de dados compartilhados.

O processo de troca de informações ou dados, que força as máquinas a se adaptarem na conferência, pode ser bem compreendido quando, em uma conferência, duas máquinas desejam iniciar o processo de compartilhamento ou transferência de arquivos. As duas máquinas que estão fazendo parte da conferência possuem algumas características que tornam bem restrito o uso dos recursos, como por exemplo, a máquina A possui uma conexão discada e pouca capacidade de processamento, já a máquina B, tem uma boa capacidade de processamento e utiliza uma conexão de banda larga como ADSL. Com o conhecimento das limitações de cada máquina, pode ser estabelecida uma conferência onde ambas as máquinas possam enviar e receber dados.

Na figura 3.4, é mostrado um trecho de um pacote capturado em uma conferência com T.127, onde é constatada a presença de primitivas ou mensagens pertinentes ao

T.128. A ilustração a seguir possui algumas abstrações para limitarmos a visualização somente a dados referentes ao T.120.

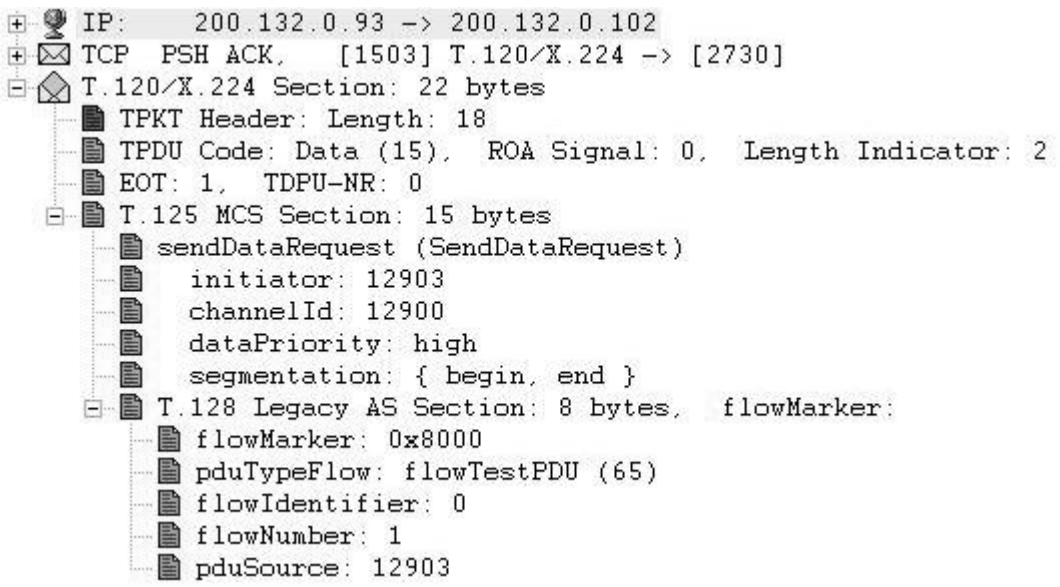

Figura 3.4: Primitiva FlowMarker referente ao T.128

A primitiva FlowMarker realiza algumas trocas de dados pertinentes ao compartilhamento de aplicações mesmo em uma conferência com transferência de arquivos; desta forma o uso da primitiva também busca "preparar" a conferência em uma futura utilização do T.128. Existe a desvantagem das trocas de mensagens do T.128 interferirem no desempenho da rede, gerando um maior tráfego. Além de mais informações geradas na rede, a utilização de primitivas do T.128 em uma conferência com T.127 também acarreta um custo de processamento nas máquinas dos usuários.

3.3 Simplificações propostas para a camada T.127

Como foi comentado nos tópicos anteriores, foram estabelecidas simplificações (abstrações) tanto no nível do MCS e GCC como também em chamadas não relacionadas ao serviço de colaboração visual de dados utilizado na conferência em andamento. Além dos dois tipos de simplificações apresentados, existe uma terceira etapa de simplificação para se definir um nodo T.127 *lite* . A terceira simplificação que pode ser estabelecida é realizada sobre o protocolo ou camada T.127 (ITU, 1995b).

Com o objetivo de implantar um serviço auxiliar de colaboração visual de dados que contenha um perfil minimalista, são sugeridas algumas abstrações dentro da padronização T.127. Essas abstrações não podem impedir o funcionamento de uma estrutura já definida, mas sim, diminuir a complexibilidade. A seguir, é apresentada a relação das chamadas ou primitivas que podem ser minimizadas no T.127.

3.3.1 Private-Channel-Join-InvitePDU (Private-Channel-Join-InvitePDU)

No método Private-Channel-Join-InvitePDU**,** pode-se ter primitivas responsáveis pela identificação do canal, identificação dos dados do canal ou até mesmo o modo (por exemplo broadcast ou ponto a ponto).

Pode-se propor abstrações no Private-Channel-Join-InvitePDU quando houver comunicação com o File APE. O File APE é a parte da aplicação de transferência de arquivos que envia aspectos que não têm nenhum efeito direto na interconexão (por exemplo, interface de usuário), podendo ser implementada em qualquer plataforma. Por ser executada na máquina cliente, a recomendação T-127 não cria restrições para a implementação. Uma aplicação de usuário confia nos serviços de uma APE para se comunicar com pontos de aplicações em outros nodos (ITU, 1995b). Na aplicação do usuário, não há comunicação com MCS ou GCC; isto é feito pelo File APE. A aplicação de usuário inicia uma sessão de transferência de arquivo por seu File APE e especifica as capacidades de aplicação e modo de sessão. Uma vez estabelecida a sessão, todas as transações específicas do MBFT (Protocolo para Transferência de Arquivos Binários Multi-ponto) são executadas pelo File APE em nome da aplicação de usuário. Na figura 3.5, é apresentado o trecho de um pacote capturado em uma conferência T.127 contendo o Private-Channel-Join-InvitePDU.

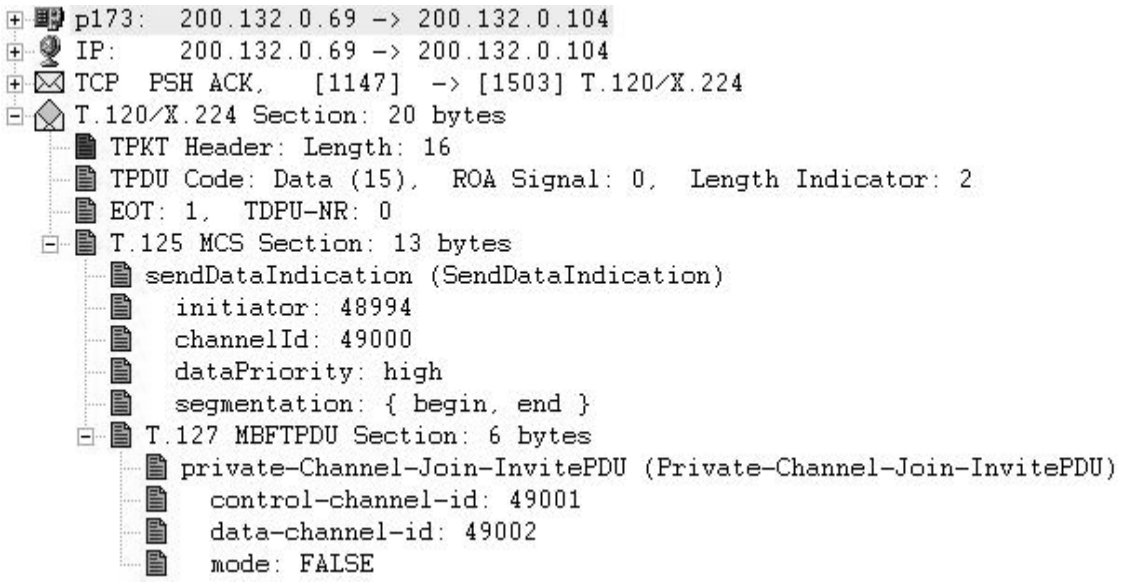

Figura 3.5: Primitiva Private-Channel-Join-InvitePDU do T.127

Quando se tratar da Recepção de Arquivos apenas para a APE, a primitiva Private-Channel-Join-InvitePDU não exige a implementação do envio de PDUs, mas o nodo *T.127 lite* deve ser capaz de receber PDUs relacionados à Recepção de Arquivos para a APE.

Na Transmissão de Arquivos apenas para a APE, o Private-Channel-Join-InvitePDU do nodo *T.127 lite* deve obrigatoriamente receber PDUs, o envio de PDUs não necessita ser implantado.

Quanto estiver sendo tratado de comunicações ou transferências de arquivos para diversas máquinas ao mesmo tempo, a padronização T.127 refere-se ao nodo da aplicação do T.127 responsável pela Transmissão e Recepção de Arquivos para a APE. Para o nodo comentando, o método Private-Channel-Join-InvitePDU não necessita implantar o envio de PDUs para a APE, porém o nodo deve obrigatoriamente implantar o recebimento de PDUs para a APE.

O fato de não implementar o PDU para envio do Private-Channel-Join-InvitePDU e somente os relacionados ao recebimento, acaba proporcionando um número menor de mensagens circulando na rede e um custo menor de processamento das informações na máquina do usuário.

3.3.2 Private-Channel-Join-ResponsePDU (Private-Channel-Join-ResponsePDU)

O método Private-Channel-Join-ResponsePDU trata de uma resposta ao método anterior (Private-Channel-Join-InvitePDU) e conseqüentemente, está diretamente relacionado às primitivas de identificação de canal, identificação dos dados do canal, o modo que os dados trafegam no canal (*broadcast* ou ponto a ponto). Com isso, qualquer abstração que possa a ser feita no método Private-Channel-Join-InvitePDU pode ser seguida para o Private-Channel-Join-ResponsePDU. A figura 3.6 apresenta trecho de uma monitoração de uma conferência, onde o T.127 é utilizado e mostra a primitiva Private-Channel-Join-ResponsePDU.

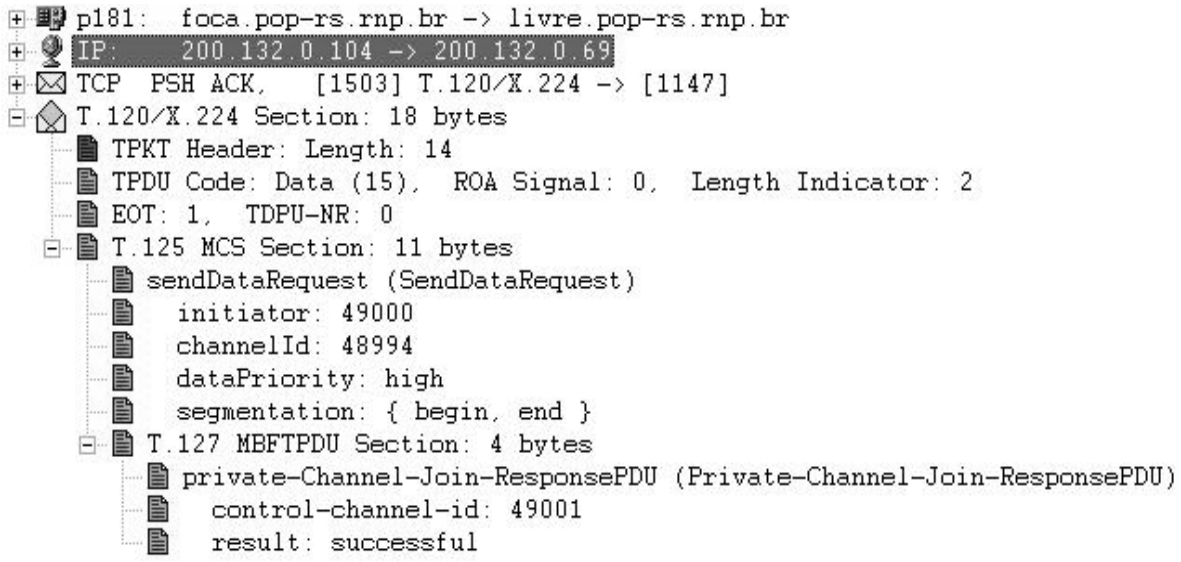

Figura 3.6: Primitiva Private-Channel-Join-ResponsePDU do T.127 encontrada

Para a Recepção de Arquivos apenas para a APE, a primitiva Private-Channel-Join-InvitePDU deve exigir a implementação do envio de PDUs, já o recebimento de PDUs não é exigido.

No que diz respeito à Transmissão de Arquivos apenas para a APE, o Private-Channel-Join-InvitePDU do nodo *T.127 lite* deve obrigatoriamente enviar PDUs, ao passo que o processo de receber PDUs não necessita ser implantado.

Para a comunicação ou transferências de arquivos para diversas máquinas ao mesmo tempo, a padronização T.127 refere-se ao nodo da aplicação do T.127 responsável pela Transmissão e Recepção de Arquivos para a APE. A primitiva Private-Channel-Join-ResponsePDU não necessita implantar o recebimento de PDUs para a APE, porém o nodo deve obrigatoriamente implantar o envio de PDUs para a APE.

Como foi comentado na primitiva anterior, ao realizar algumas abstrações no Private-Channel-Join-ResponsePDU, pode-se reduzir também o número de mensagens circulando na rede e dados a serem processados pelas máquinas dos usuários.

3.3.3 File-OfferPDU (File-OfferPDU)

A primitiva em questão pode ser simplificada na Recepção de Arquivos apenas para APE, onde não é exigida a implantação da primitiva para o envio de PDU. Como a APE é parte da aplicação de transferência de arquivos, não tem nenhum efeito direto na interconexão (por exemplo, interface de usuário) e pode ser implementada em qualquer plataforma. Portanto, os PDUS podem ser enviados sem levar em consideração a norma, seguindo um padrão de programação definido pelo próprio programador ou projetista do software T.127 *lite* . A Figura 3.7 mostra um trecho do pacote File-OfferPDU capturado em uma conferência T.127.

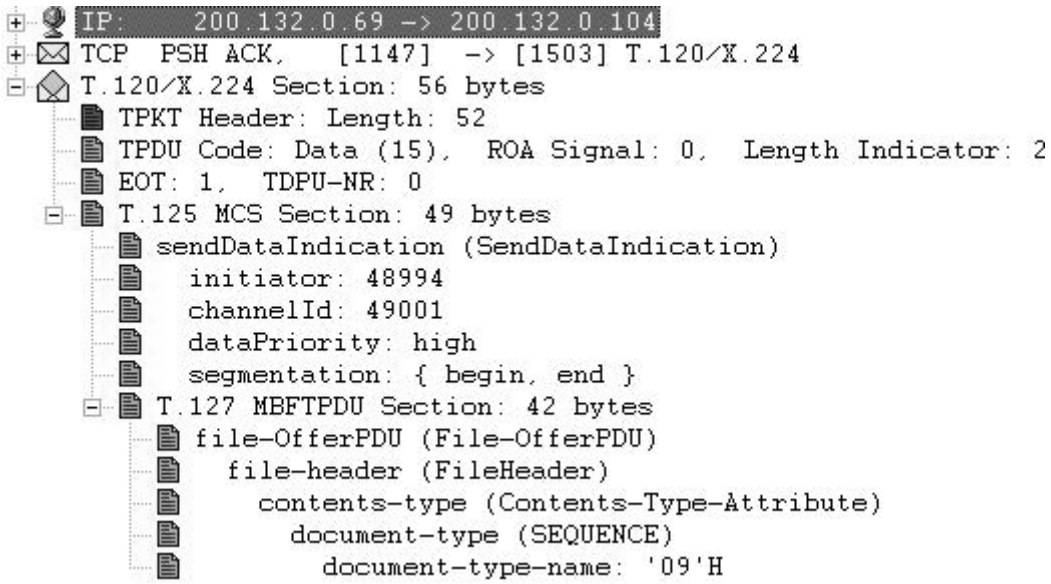

Figura 3.7: Primitiva File-OfferPDU encontrada em uma conferência T.127

3.3.4 File-AcceptPDU (File-AcceptPDU)

No método File-AcceptPDU não é exigida a implementação na recepção de arquivos para a APE quando relacionado ao recebimento de uma PDU, ou seja, o aplicativo não é obrigado a receber um arquivo, ele pode simplesmente descartá-lo sem realizar nenhum tratamento. Em se tratando de envio de PDU para APE, este método também não é obrigatório, visto que o recebimento também segue o mesmo padrão. Na Figura 3.8, é apresentado o trecho de um pacote contendo a primitiva File-AcceptPDU.

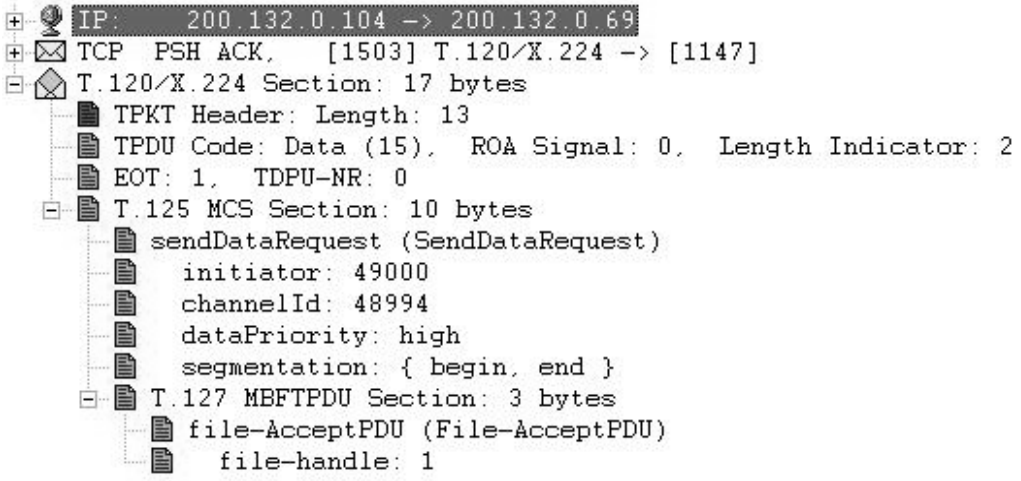

Figura 3.8: Trecho de um pacote de T.127 contendo o File-AcceptPDU

As mesmas considerações feita nos tópicos anteriores também servem para esse em que a APE não tem efeito "direto" na interconexão da rede e é portanto, livre de plataforma de implementação.

3.3.5 File-StartPDU (File-StartPDU)

Para o método File-StartPDU, quando a chamada for para o envio de PDU (recepção de arquivo apenas para APE) não é exigido a implementação. A idéia é a mesma para o recebimento de PDU (na questão de transmissão de arquivo apenas para a APE). A simplificação é possível por não haver uma comunicação direta com MCS ou GCC. Essa comunicação é realizada pelo File APE, voltando novamente ao termo sobre independência de plataforma. A figura 3.9 mostra trecho de um pacote capturado em uma conferência com T.127.

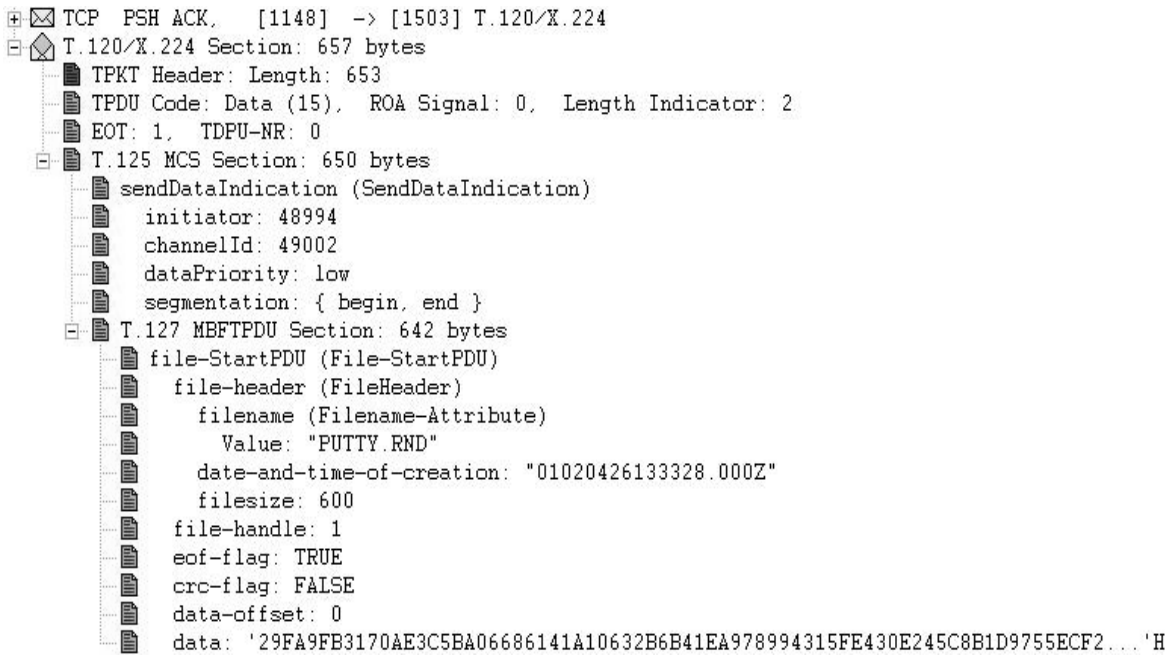

Figura 3.9: Trecho de um pacote onde a primitiva File-StartPDU é encontrada

3.3.6 Mbft-NonStandardPDU (MBFT-NonStandardPDU)

MBFT-NonStandardPDUs pode ser enviado a qualquer hora, mas em modo de administração só pode ser enviado por File APEs que foram concedidos o Privilégio de Não-padrão. É opcional este método segundo a especificação T.127, não é necessário implementá-lo na: recepção de arquivos para APE (envio e recebimento), transmissão de arquivos para APE (envio e recebimento) e transmissão e recepção de arquivo para APE. A Figura 3.10 apresenta o trecho de um pacote contendo a primitiva Mbft-NonStandardPDU encontrada em uma conferência T.127.

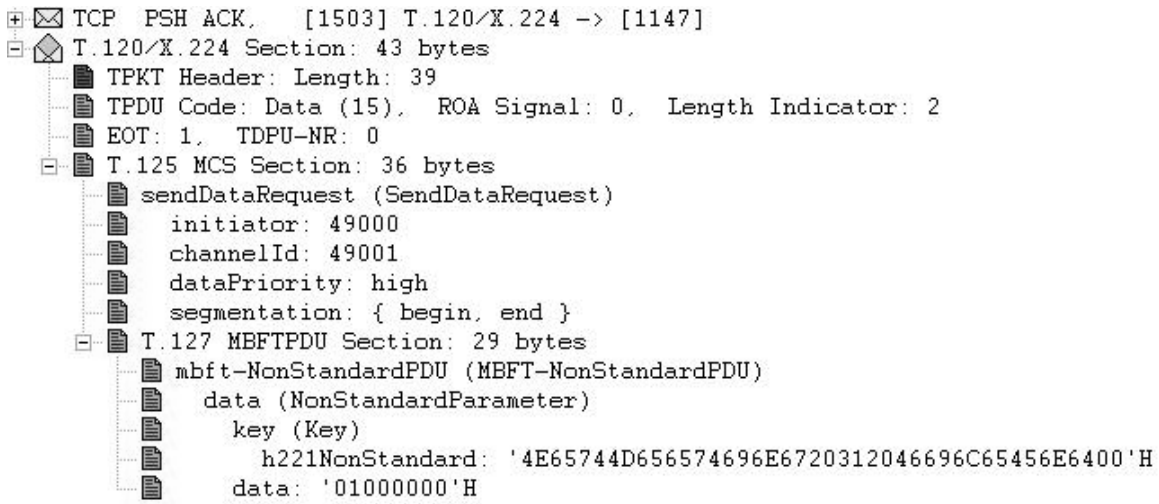

Figura 3.10: Pacote com a primitiva Mbft-NonStandardPDU

3.4 Seqüência de primitivas enviadas em uma conferência com T.127

Para a construção de uma solução T.127 *lite*, inicialmente é necessário o conhecimento sobre quais as primitivas devem ser trocadas entre estações que fazem parte da videoconferência ou colaboração visual de dados, com o uso do T.127 e a ordem em que estas ocorrem.

Na figura 3.11, são apresentadas as primitivas que são trocadas entre duas estações que estão utilizando o T.127. A figura pode dar uma noção sobre a quantidade de informações que são transferidas através da rede e processadas pelas máquinas dos participantes da conferência, assim como sua seqüência.

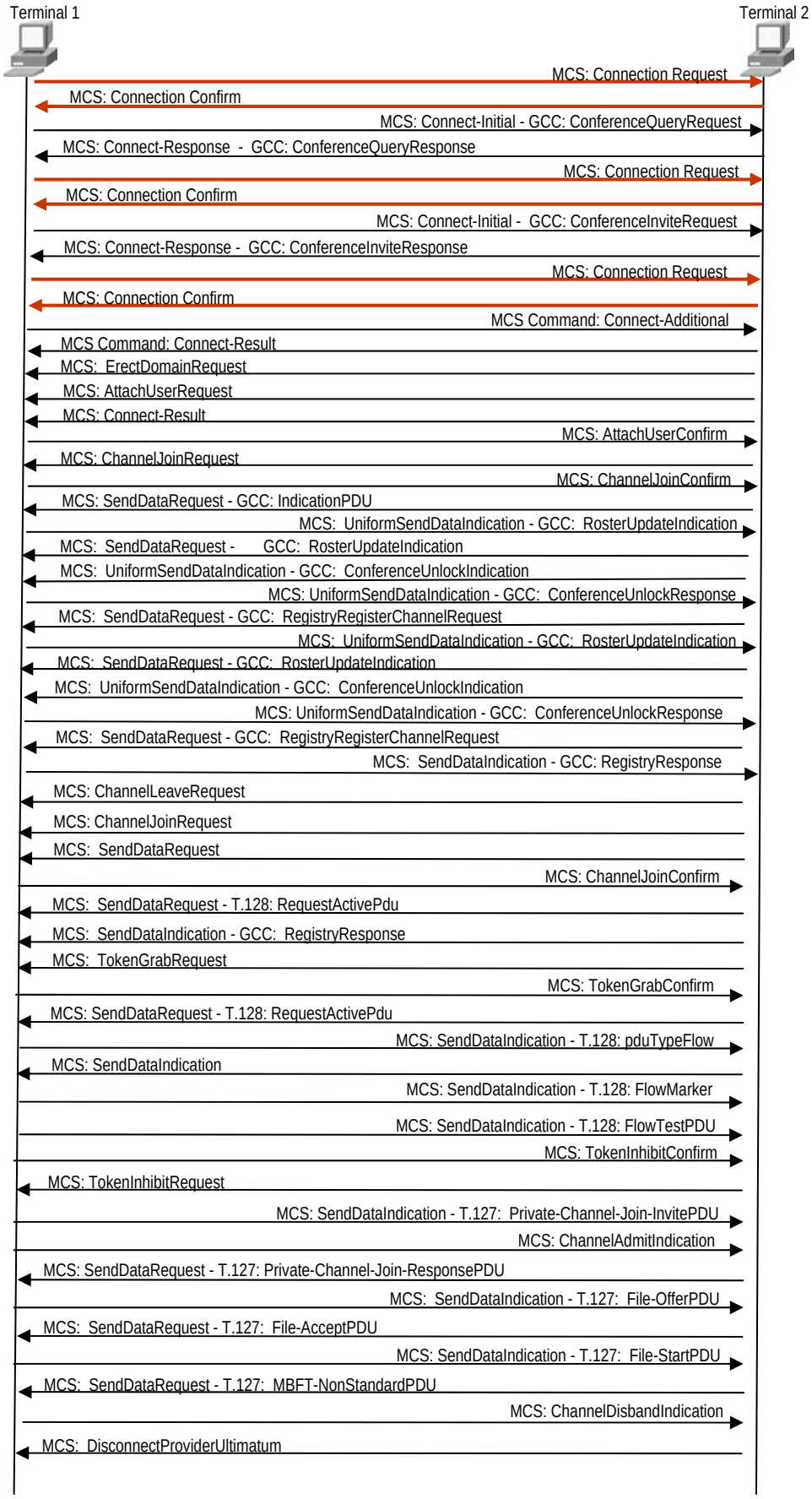

Figura 3.11- Seqüências de primitivas para o T.127

3.5 Seqüência de primitivas enviadas no T.127 *lite*

Tendo um conhecimento sobre quais as primitivas são trocadas e a sua ordem entre estações participantes de uma conferência, podem-se definir a seqüência e ordem das primitivas no T.127 *lite*.

Nos tópicos anteriores deste capítulo foram apresentadas algumas simplificações que podem ser adotadas para um nodo T.127 *lite*. A figura 3.12 busca sintetizar o que foi comentado anteriormente, apresentando a troca e a ordem das primitivas entre estações que executam o T.127 *lite*.

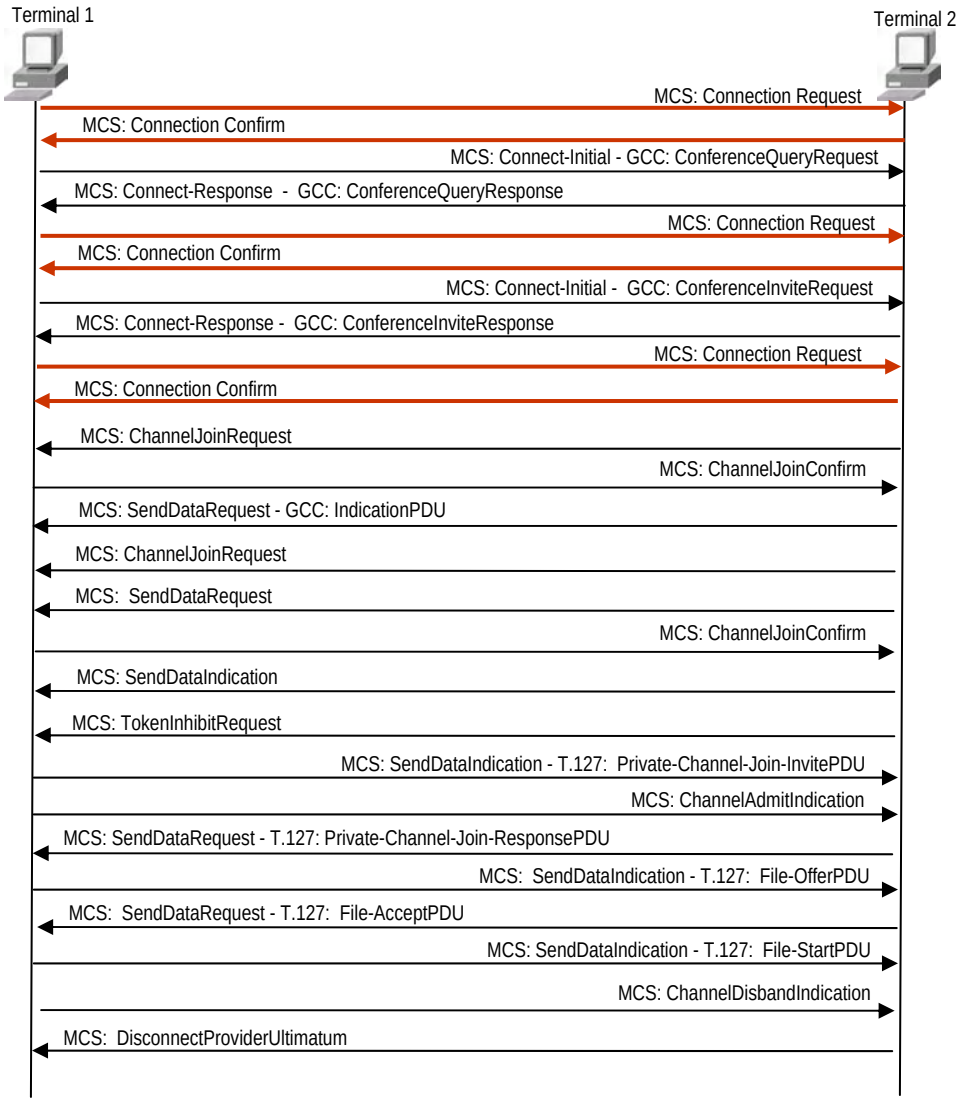

Figura 3.12 – Seqüências de primitivas para o T.127 *lite*

3.6 Considerações Finais Sobre o Capítulo

O presente capítulo visou apresentar algumas simplificações ou omissões que podem ser feitas na Arquitetura do T.120 para que seja possível a criação de um modelo *T.127 lite* que realize a transferência de arquivos. As simplificações ou omissões propostas

buscam diminuir alguns problemas que podem ocorrer em videoconferência com colaboração visual de dados, como por exemplo, um grande número de mensagens circulando pela rede e o custo de processamento do aplicativo na máquina do usuário, e atender as diversas solicitações e indicações de serviços.

No próximo capítulo, serão apresentados o protótipo para solução *lite* e os testes realizados e as monitorações obtidas através desta solução.

4 PROTÓTIPO PARA A SOLUÇÃO *LITE*

O capítulo anterior apresentou o modelo minimalista para transferência de arquivos, onde foram sugeridas diversas simplificações no nível do MCS e GCC, chamadas não pertinentes ao serviço T.127 utilizado e simplificações específicas da camada T.127. O fato de seguir algumas das simplificações sugeridas no capítulo anterior vem ao encontro da criação de um protótipo *lite* .

Esta proposta visa implementar, em parte, algumas chamadas ou descartar a utilização de outras, como por exemplo, a primitiva FlowMarker, referente ao serviço T.128 que não é utilizado.

O capítulo em questão procura apresentar ao leitor o ambiente utilizado para o desenvolvimento do protótipo, a modelagem dos módulos da arquitetura e as estruturas que são e devem ser enviadas pelo protótipo.

4.1 Ambiente de programação e recursos

O protótipo que procura demonstrar a solução *lite* do T.127 foi desenvolvido utilizando a Linguagem Java (sdk 1.4.1_02-b06) e a ferramenta JBuilder SE 7.0.1.55.0, que apresenta algumas facilidades para manipulação dos pacotes, classes e métodos utilizados.

A parte inicial do programa foi implementada baseado na RFC983 (ISO, 2002), o qual argumenta que se deve ter uma camada independente para definir o TSAP (Transport Service Access Point) que aparece ser idêntico ao serviço e interfaces oferecido pela ISO. O TSAP faz o uso de primitivas de serviço que são definidas em quatro níveis: pedido, indicação, resposta e confirmação. A figura 4.1 representa a comunicação estabelecida entre uma aplicação cliente no nível de transporte e o respectivo servidor no mesmo nível.

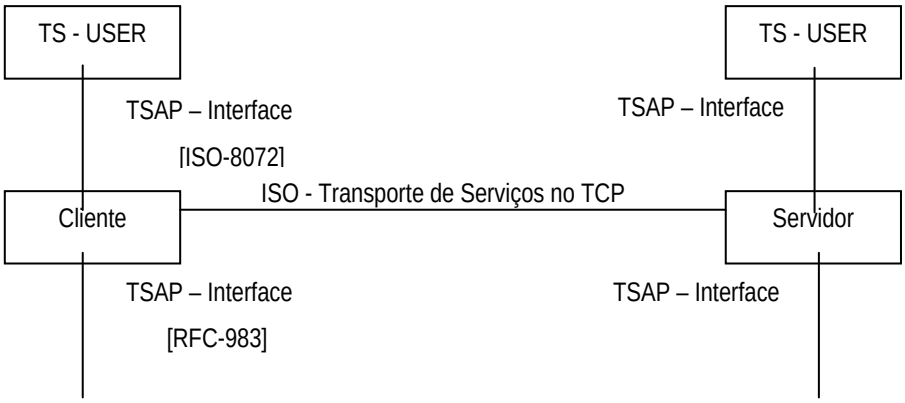

Figura 4.1: Representação da comunicação cliente/servidor através do TSAP

Através da utilização de um algoritmo denominado Singleton Pattern, pode-se iniciar o processo de estabelecimento de canal no nível de transporte. O algoritmo em questão garante um ponto de acesso global para toda a aplicação e desta forma, quando um canal é criado, o mesmo pode ser utilizado por outros pontos da aplicação e de forma sincronizada por cada ponto. Na figura 4.2, é apresentado o código da classe que utiliza o Singleton Pattern (GAMMA; HELM; JOHNSON, 1995).

```
1  /* Mantém sempre no mesmo socket fazendo com que seja respondido sempre pelo  mesmo
2   caminho na mesma textarea (garante que a mesma textarea no cliente e no servidor **/
3  package test;
4
5  import java.awt.*;
6  import java.awt.event.*;
7  import javax.swing.*;
8  import javax.swing.JTextArea;
9
10 //metodo que garante a sincronização pela mesma porta.
11 public class SingletonJTextArea {
12     private static SingletonJTextArea instance;
13     private JTextArea jTextArea;
14
15 //garante assim que um canal de comunicação após ser criado
16 //possa ser utilizado por outros pontos da aplicação e comunicação
17     public synchronized static SingletonJTextArea getInstance() {
18         if( instance == null ) {
19             instance = new SingletonJTextArea();
20         }
21
22         return instance;
23     }
24
25      /** faz this.obj(jTextArea) apontar para JTextArea criado em Frame1*/
26
27     public void setJTextArea(JTextArea jTextArea) {
28           this.jTextArea = jTextArea;
29     }
30
31     public void setTextTOJTextArea(String text) {
32           jTextArea.append(text + "\n");
33     }
34 }
```

Figura 4.2: Código do Singleton Pattern

Através da utilização do algoritmo Singleton Pattern, foi garantido que outros pontos de aplicação possam utilizar o canal estabelecido. Neste caso, isso pode ser exemplificado com a variedade de primitivas que são executadas entre as máquinas que fazem parte da colaboração visual de dados. As primitivas são associadas como pontos de aplicação, como por exemplo, a primitiva MCS-Connect-Initial, que solicita o estabelecimento de um canal entre as camadas MCS das máquinas participantes da conferência.

Por fim, tendo a garantia de que diversos pontos de aplicação possam utilizar a conexão, foi utilizado um *socket* para se obter a comunicação via rede. Inicialmente, o *socket* utiliza a porta 1503, que por definição é a porta do T.120.

4.2 Modelo de Arquitetura para o protótipo T.127 lite

Para o desenvolvimento e apresentação da proposta do protótipo T.127 *lite,* é sugerido um modelo de arquitetura onde são apresentados os módulos necessários para o andamento de uma videoconferência ou colaboração visual de dados.

Inicialmente, a arquitetura está dividia em três visões ou módulos: Protocolos de Conexão, Ambiente Visual e Módulo Interativo. A figura 4.3 representa a arquitetura proposta para o T.127 lite. Alguns dos módulos apresentados têm a mesma função das camadas que compõem a pilha de protocolos T.120 e em muitos casos, o mesmo nome. Como por exemplo, o Protocolo para teleconferência audiovisual, mais conhecido como T.123.

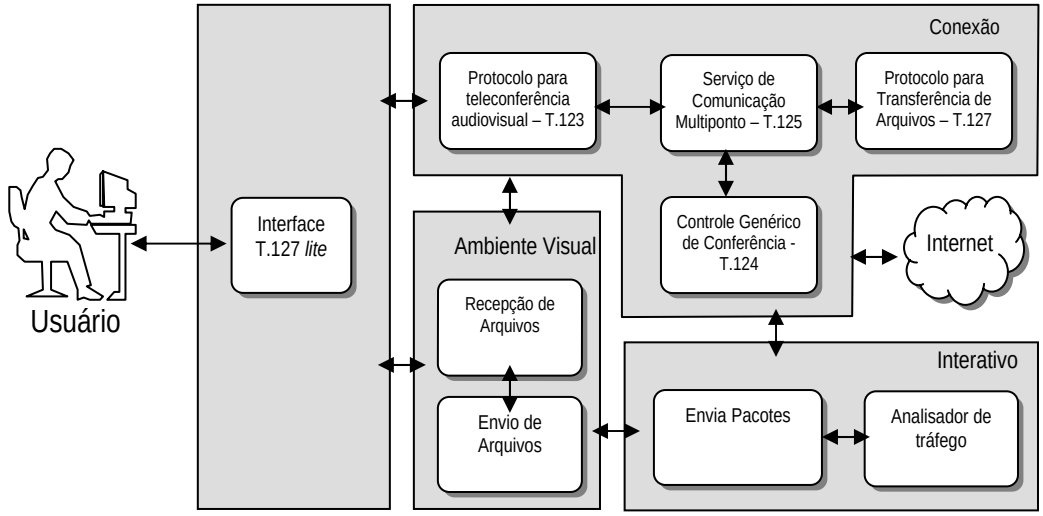

Figura 4.3: Arquitetura proposta para o T.127 *lite*

4.2.1 Módulo Conexão

No protótipo T.127 *lite* são definidos quatro níveis ou módulos para tratar do estabelecimento da conexão nas diversas camadas que compõem o T.120.

- Protocolo para teleconferência audiovisual – T.123: Este módulo corresponde à camada T.123 da arquitetura T.120 e é responsável por estabelecer a comunicação no nível de transporte. A troca dos TPDUs

Connection Request e Connection Confirm entre as máquinas participantes da conferência, estabelece a conexão no T.123.

Quando uma estação deseja estabelecer a conexão no T.123, ela deve enviar um pacote Connection Request para a máquina de destino. Ao receber este, ela responde com o envio do Connection Confirm, estabelecendo, portanto, a conexão.

Quando a máquina que executa o protótipo T.127 lite recebe o pacote Connection Request, o módulo de Conexão deve comunicar-se com o módulo Interativo, que tem a possibilidade de realizar a análise de pacotes, a fim de descobrir o tipo de primitiva.

O módulo Interativo detecta a TPDU Connection Request e comunica-se com o módulo Envia Pacotes, com o objetivo de solicitar o envio de um retorno no formato da TPDU Connection Confirm.

Por fim, o protocolo para teleconferência audiovisual, ou simplesmente T.123, repassa a informação de conexão estabelecida para o módulo Serviço de Comunicação Multiponto.

- Serviço de Comunicação Multiponto – T.125: o módulo Serviço de Comunicação Multiponto possui a mesma funcionalidade do MCS (T.125) da pilha T.120. Quando o módulo recebe o aviso do T.123 de que a conexão no nível de transporte já foi estabelecida, este pode tratar da comunicação no T.125.

  O primeiro passo para o estabelecimento da conexão no T.125 é o envio da primitiva MCS: Connect-Initial. Porém antes dessa primitiva ser enviada, ela deve ser repassada para o módulo Serviço de Comunicação Multiponto (ou GCC), para que sejam encapsuladas as informações para estabelecimento de comunicação no Controle Genérico de Conferência (T.124).

- Controle Genérico de Conferência – T.124: este módulo recebe as informações repassadas pelo módulo anterior e procura encapsular informações pertinentes a sua camada, idêntica ao T.124 da pilha T.120. Após as informações serem encapsuladas, ocorre novamente uma comunicação com o Módulo Interativo para que este envie a primitiva para a máquina de destino.

  Quando o módulo interativo recebe a resposta (MCS: Connect-Response) do pacote que foi enviado, ele retorna as informações para o módulo de conexão; neste caso, informando que uma nova primitiva deve ser encapsulada sobre o próximo pacote MCS a ser enviado.

  No momento que ocorrer a conexão no nível do T.124, através das trocas de primitivas MCS e GCC, pode ser estabelecida a conexão com o Protocolo para Transferência de Arquivos.

- Protocolo para Transferência de arquivos – T.127: quando a conexão no T127 é estabelecida, o Protocolo para Transferência de Arquivos encapsula suas informações sobre o MCS. Essas informações devem ter a finalidade de estabelecer a conexão no nível do T.127.

  Algumas primitivas são trocadas entre as camadas ou módulos participantes da conferência. Neste caso, sempre existe uma comunicação com o módulo Interativo a fim de que este possa analisar as informações recebidas e informar ao módulo de Comunicação que tipo de primitiva deve ser enviada pelo T.127.

  No momento em que todas as informações de estabelecimento de conexão no T.127 forem trocadas, deve ser iniciada a transferência do arquivo a ser escolhido pelo usuário.

  Para que possa ser enviado o arquivo, deve ocorrer uma comunicação inicialmente com o módulo de Ambiente Visual. Este módulo tem a finalidade de repassar ao T.127 o arquivo a ser enviado, ou seja, o usuário determina qual o arquivo.

4.2.2 Módulo Interativo

O módulo interativo está dividido em Envia Pacotes e Analisador de tráfego. Inicialmente esses dois módulos componentes do Módulo Interativo comunicam-se tanto com o módulo de Conexão quanto com o Ambiente Visual.

- Analisador de Tráfego: o módulo Analisador de Tráfego recebe os pacotes da conferência e procura analisá-los para que se possa determinar qual o tipo de informação (pacote) deve ser repassada para a máquina que inicialmente enviou a informação.

  Conhecendo o pacote recebido, o Analisador de Trafego solicita ao módulo de Conexão a "construção" do pacote que deverá ser enviado em resposta ao recebido.

- Envia Pacotes: o módulo Envia Pacotes tem a funcionalidade de enviar as primitivas para as estações participantes da conferência. Como por exemplo, quando for recebida a primitiva MCS-Connect_initial, o Analisador de Tráfego informa ao módulo de Conexão, o tipo de primitiva que ele deve "construir", neste caso o MCS-Connect-Response.

  Por sua vez, o módulo de Conexão repassa a primitiva ao módulo Envia Pacotes para que este possa enviar para a máquina de destino.

4.2.3 Ambiente Visual

O módulo Ambiente Visual tem a função de informar ao usuário o tipo de transação que está ocorrendo (transferência de arquivo ou envio). O ambiente visual está dividido em Envio de Arquivos e Recepção de Arquivos.

- Envio de Arquivo: o módulo em questão tem a funcionalidade de ativar uma opção (ou janela) para o usuário escolher o arquivo que deve ser transferido. Quando o usuário determina o arquivo, essa informação é repassada ao módulo de Conexão com o objetivo de transferir o arquivo para a máquina de destino.

- Recepção de Arquivos: este módulo tem a funcionalidade de informar ao usuário se o mesmo deseja receber determinado arquivo ou não. Caso o usuário desejar receber o arquivo, é ativada a opção para determinar o caminho (pasta ou diretório) e o nome do arquivo.

  O processo de transferência é encerrado quando o usuário não aceita o recebimento do arquivo.

4.3 Modelagem do Diagrama de Classes

Para a implementação e organização da solução T.127 *lite*, foi inicialmente mapeado o seguinte conjunto de requisitos para construção e enumerados quatro módulos da solução *lite* que são o T.123, T.124, T.125 e T.127. Cada módulo possui suas especificação (atributos) e as operações (métodos), que neste caso são todas as chamadas que a solução *lite* deve apresentar.

Uma das formas de representar a organização dos módulos foi através do Diagrama de Classes, que é um dos principais diagramas estruturais da UML (Unified Modeling Language). O Diagrama de Classes mostra a visão estática da estrutura do sistema e deve esboçar o que interage e não o que acontece quando as classes se comunicam (Figura 4.4).

Figura 4.4: Diagrama de Classes do T.127 *lite*

Na figura anterior, é mostrada a visão estática dos módulos da solução *lite*. O módulo T.125 (MCS) deve obrigatoriamente estar ligado ao módulo T.123, pois é este que é responsável por iniciar a comunicação via rede através do *socket* TCPs e estabelecer a conexão inicial através das chamadas Connection Request, Connection Confirm e Data. Por sua vez, o T.124 necessita comunicar-se diretamente com o T.125, pois somente após o estabelecimento da conexão no nível do MCS que podem trafegar os pacotes GCC entre as máquinas. Por fim, o T.127 comunica-se com o T.125 para poder enviar arquivos.

A partir do momento que temos uma visão estática e genérica do sistema, através do Diagrama de Classes, é possível partir para as especificações de cada módulo.

4.3.1 O módulo T.123

O módulo T.123 é o responsável pela criação da comunicação via rede, como comentado anteriormente, utilizando *socket* TCP para isso. Essa comunicação não está visível na estrutura definida no Diagrama de Classes, mas é inerente ao sistema. A figura 4.5 apresenta o módulo T.123 com seus atributos e métodos (que são as demais chamadas da camada MCS).

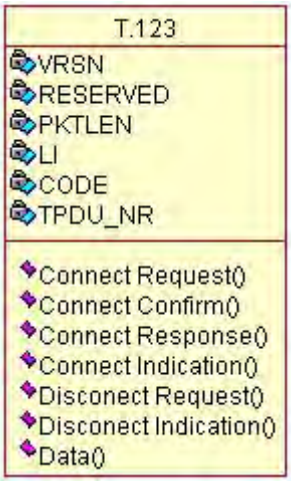

Figura 4.5: Representação do módulo T.123

Inicialmente a RFC983 (ISO, 2002) define que antes de ser enviado um pacote ou TPDU Connection Request, Connection Confirm ou Data, deve ser enviado o cabeçalho do pacote (*packet header*) que possui a seguinte estrutura (Figura 4.6):

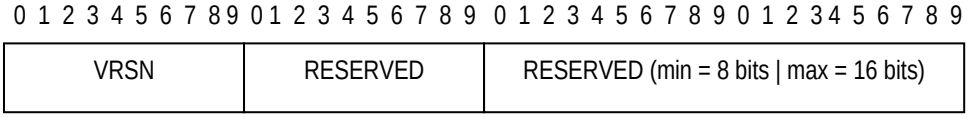

Figura 4.6: Formato do cabeçalho do pacote

No momento que o módulo T.123 envia o cabeçalho do pacote, os próximos atributos a serem enviados são: LI (identificador de tamanho – *length indication*), CODE (dependendo do valor, pode-se determinar se o pacote é um Connection Request, Connection Confirm ou Data) e o TPDU_NR (indica uma comunicação entre o módulo MCS). A figura 4.7 mostra os valores do cabeçalho do pacote que são enviados pelo T.127 *lite* .

```
1  package test;
2
3
4  public interface DataPackageHeader {
5
6     public static final int  VRSN = 0x03;// --> 07
7     public static final int  RESERVED =  0x00;
8     public static final int  PKTLEN = 0x00;
9     public static final int  _PKTLEN = 0xB5;//
10    public static final int  LI = 0x02;
11    public static final int CODE = 0xF0;//240; // --> 0 code
12    public static final int TPDU_NR = 0x80;//128;  //código do MCS
13 }
```

Figura 4.7: Cabeçalho do pacote enviado pelo protótipo

A figura 4.7 representou a estrutura que o *T.127 lite* utiliza para enviar os dados através da rede. Dependendo do valor atribuído para o TPDU_NR, este indicará o tipo de pacote MCS que estará trafegando na rede, por exemplo, quando o campo possuir o valor atribuído de 80 (0x80). Trata-se de um pacote MCS: Connect-Initial.

4.3.2 O módulo T.125

Se o algoritmo Singleton Pattern tem o objetivo de garantir um ponto de acesso global para toda a aplicação, o MCS faz o uso dessa garantia para reunir as conexões com os múltiplos pontos de aceso. Desta forma, o módulo faz a reunião das conexões ponto-a-ponto para assim criar sobre o mesmo canal, um domínio multiponto, conforme comentado anteriormente no capítulo 2.

A figura 4.8 mostra uma representação do módulo T.125 através da UML com seus atributos e as diversas chamadas ou pacotes que podem ocorrer em uma conferência T.127 *lite*, e que nesta figura são representadas como métodos.

Figura 4.8: Representação através da UML do T.125

O envio dos atributos do módulo T.125 só ocorre quando for recebido pela estação de destino o quadro cabeçalho do pacote e retornado a resposta para a máquina de origem. A partir deste ponto, tem-se a conexão no nível do T.123 e o módulo T.125 pode começar a enviar seus atributos. Na figura 4.9, é apresentado um dos oito atributos que devem ser enviados, o MCS_COMAND.

```
1  package test;
2
3
4  public interface MCS_COMMAND {
5
6     public static final int MCS_SECTION_01 = 0x7F;// 7F
7     public static final int MCS_SECTION_02 = 0x65;// 65
8     public static final int MCS_SECTION_03 = 0x81;// 81
9     public static final int MCS_SECTION_04 = 0xAA;// AA
10    public static final int MCS_SECTION_05 = 0x04;// 04
11    public static final int MCS_SECTION_06 = 0x04;// 04 END OF TRANSMISSION
12    public static final int MCS_SECTION_07 = 0x01;// 01
13    public static final int MCS_SECTION_08 = 0x00;// 00
14    public static final int MCS_SECTION_09 = 0x00;// 00
15    public static final int MCS_SECTION_10 = 0x10;// 10
16    public static final int MCS_SECTION_11 = 0x04;// 04 END OF TRANSMISSION
17    public static final int MCS_SECTION_12 = 0x04;// 04 END OF TRANSMISSION
18    public static final int MCS_SECTION_13 = 0x01;// 01
19    public static final int MCS_SECTION_14 = 0x00;// 00
20    public static final int MCS_SECTION_15 = 0x00;// 00
21    public static final int MCS_SECTION_16 = 0x10;// 10 DATA LINK ESCAPE
22    public static final int MCS_SECTION_17 = 0x01;// 01
23    public static final int MCS_SECTION_18 = 0x01;// 01
24    public static final int MCS_SECTION_19 = 0XFF;// FF
25 }
```

Figura 4.9: Atributo MCS_COMMAND

No mesmo formato que este atributo deve ser enviado pelo T.127 *lite* , os demais atributos que são: MCS_SECTION, CALLING_DOMAIN_SELECTOR, CALLED_DOMAIN_SELECTOR, UPWARD_FLAG, TARGET_DOMAIN_PARAMETRS, MAXIMUM_DOMAIN_PARAMETERS e MINIMUM_DOMAIN_PARAMETERS também devem ser enviados.

O MCS_COMMAND é o atributo que determina qual o método deve ser enviado através do canal de comunicação. Na figura anterior o método representado pelos valores atribuídos é o MCS: Connect-Initial.

4.3.3 O módulo T.124

Como comentado no capítulo 2, o GCC possui funcionalidades como manter listas de participantes da colaboração visual de dados e suas aplicações. Também procura seguir os recursos do MCS a fim de que não ocorram conflitos quando os participantes de videoconferências ou de colaboração visual de dados usem múltiplos protocolos de aplicação, como o T.126, T.127 ou T.128.

A figura 4.10 representa o módulo T.124 através da UML com os seus atributos que são: o SECTION, MAX_CHANNEL_IDS, NUM_PRIORITIES e MAX_MCSPDU.

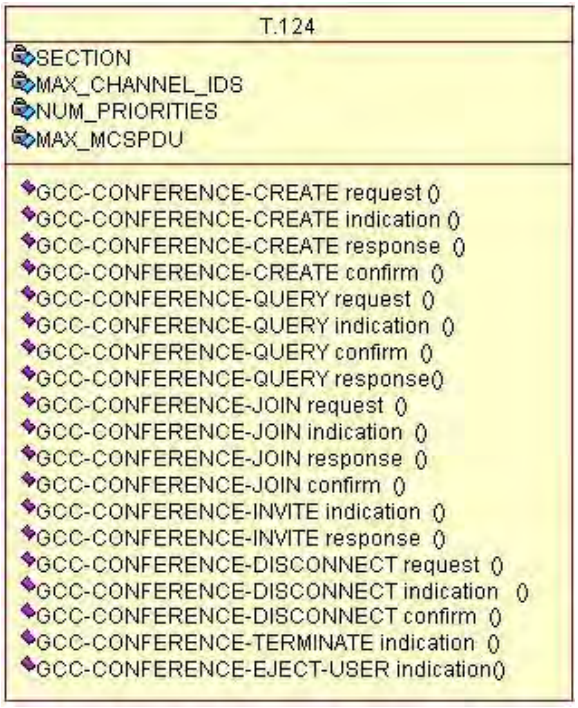

Figura 4.10: Representação através da UML do T.124

Além da especificação dos atributos, existem métodos (ou chamadas) que a solução *T.127 lite* deve implementar para garantir as funcionalidades do GCC. Dentre as diversas primitivas do GCC que a solução *T.127 lite* apresentou na ilustração anterior, é apresentada com mais detalhes na Figura 4.11, o código referente ao GCC_ConferenceInviteRequest.

```
1   package test;
2
3   public interface  GCC_ConferenceInviteResponse {
4
5     public static final int GCC_01 = 0x04; // Cabeçalho do GCC
6     public static final int GCC_02 = 0x3F; // Cabeçalho do GCC
7     public static final int GCC_03 = 0x00;// 00
8     public static final int GCC_04 = 0x05;// 05
9     public static final int GCC_05 = 0x00;// 00
10    public static final int GCC_06 = 0x14;// 14
11    public static final int GCC_07 = 0x7C;// 7C
12    public static final int GCC_08 = 0x00;// 00
13    public static final int GCC_09 = 0x01;// 01
14    public static final int GCC_10 = 0x37;// 37
15    public static final int GCC_11 = 0x26;// 26
16    public static final int GCC_12 = 0x6C;// 6C
17
18                    :
```

Figura 4.11: Primitiva GCC-CONFERENCE-INVITE response

A norma T.120 define uma série de primitivas, além do que foi modelado nesse trabalho para o T.124, porém, existe a possibilidade de não implementá-las por se caracterizarem segundo a norma, “não obrigatórias” ou ainda, primitivas que podem ser utilizadas em novas versões do protocolo.

Mesmo sendo primitivas “não obrigatórias”, foi constatado o uso destas através de monitorações que foram realizadas com softwares proprietários, como por exemplo o NetMeeting da Microsoft. Com isso, pode-se constatar que existem chamadas que trafegam desnecessariamente na rede, gerando um maior tráfego.

4.3.4 O módulo T.127

O módulo T.127 define como os arquivos são simultaneamente transferidos entre os participantes da conferência. É permitido que um ou mais arquivos possam ser selecionados e transmitidos de forma compactada ou descompactada para todos ou somente os participantes selecionados da conferência

A figura 4.12 apresenta através da UML os atributos MBFTPDU Section, Control Chanel Id, Data Channel, mode, Result, File Handle e Data. As primitivas que devem fazer parte do módulo *T.127 lite* são Private-Channel-Join-InvitePDU, Private-Channel-Join-ResponsePDU, File-OfferPDU, File-AcceptPDU e File-StartPDU.

Figura 4.12: Representação através da UML do T.127

Uma das primitivas do módulo T.127 que deve se enviada através da rede quando for realizada uma videoconferência ou colaboração visual de dados é o File-OfferPDU. Esta primitiva contém uma estrutura responsável para identificar qual é o arquivo que deverá ser enviado. Desta forma, o T.127 faz com que também as camadas T.123 e T.125 tomem conhecimento sobre o arquivo. A figura 4.13, representa os valores que devem ser passados para o T.123, T.125 e o T.127 das máquinas participantes da conferência através do protótipo *lite*

```
1  package test;
2
3  public interface File-OfterPDU{
4
5    public static final int File_Ofter_PDU_01 = 0x02;
6    public static final int File_Ofter_PDU_02 = 0x01;
7    public static final int File_Ofter_PDU_03 = 0x10;
8    public static final int File_Ofter_PDU_04 = 0x40;
9    public static final int File_Ofter_PDU_05 = 0x00;
10   public static final int File_Ofter_PDU_06 = 0x01;
11   public static final int File_Ofter_PDU_07 = 0x09;
12   public static final int File_Ofter_PDU_08 = 0x50;
13   public static final int File_Ofter_PDU_09 = 0x55;
14   public static final int File_Ofter_PDU_10 = 0x54;
15   public static final int File_Ofter_PDU_12 = 0x54;
16   public static final int File_Ofter_PDU_13 = 0x59;
17   public static final int File_Ofter_PDU_14 = 0x2E;
18   public static final int File_Ofter_PDU_15 = 0x52;
19   public static final int File_Ofter_PDU_16 = 0x4E;
20   public static final int File_Ofter_PDU_17 = 0x44;
21   public static final int File_Ofter_PDU_18 = 0x0F;
22   public static final int File_Ofter_PDU_19 = 0x30;
23   public static final int File_Ofter_PDU_20 = 0x31;
24   public static final int File_Ofter_PDU_21 = 0x30;
25   public static final int File_Ofter_PDU_22 = 0x32;
26   public static final int File_Ofter_PDU_23 = 0x30;
27   public static final int File_Ofter_PDU_24 = 0x34;
28   public static final int File_Ofter_PDU_25 = 0x32;
29   public static final int File_Ofter_PDU_26 = 0x36;
30   public static final int File_Ofter_PDU_27 = 0x31;
31   public static final int File_Ofter_PDU_28 = 0x33;
32   public static final int File_Ofter_PDU_29 = 0x33;
33
33 }
```

Figura 4.13: Primitiva File-OffterPDU

As diversas primitivas que devem ser enviadas no módulo T.127 possuem funções bem específicas no que diz respeito ao gerenciamento de envio de arquivos, canal e estrutura (como nome, extensão e tamanho).

Existem funções que devem ser consideradas como o fato de ser assegurado que ocorra no máximo uma transferência de arquivo via *broadcast*, evitando, assim, que a rede não fique sobrecarregada.

Também deve ser assegurado que exista no máximo um pedido de envio de arquivo, no canal, por vez, garantindo de determinada forma, que uma estação não detenha toda a capacidade de enviar e receber arquivos, mantendo certa justiça entre os participantes.

As primitivas que são enviadas entre as máquinas participantes da conferência também devem oferecer a opção ao usuário (aplicação) de rejeitar arquivos oferecidos em um canal de dados. O T.127 define que um canal de dados pode ser exclusivo (só o criador do canal pode enviar arquivos nele) ou compartilhado (qualquer participante pode enviar arquivos no canal).

4.4 Estabelecimento de conexão ou comunicação nas camadas T.120

O canal de comunicação a ser usado pelas máquinas pode ser obtido através do *socket* TCP na porta 1503 (específica do T.120). No momento que foi estabelecida a conexão com o *socket*, todas as mensagens que serão enviadas via rede usaram como origem e destino a porta 1503. As informações devem ser interpretadas pela camada T.123, que tomará a decisão sobre o que fazer com essa informação (pode responder, ignorar ou repassá-la para camada superior, neste caso o MCS ou GCC).

4.4.1 Comunicação na camada T.123

Antes do T.123 decidir o destino da informação, primeiramente deve ser estabelecida a conexão entre as camadas T.123 das máquinas participantes da conferência. Depois de estabelecida à conexão, já pode ser realizada a comunicação entre as camadas superiores que são o MCS, o GCC e por fim o T.127.

A RFC983 (ISO, 2002) define as seguintes TPDUs a serem utilizadas pelo T.123: Connection Request (CR), Connection Confirm (CC), Disconect Request (DR), Data (DT), Expedited Data e outros (que são reservados e não são inerentes ao problema em questão). A tabela 4.1, apresenta as TPDUs que devem ser utilizadas e os seus valores correspondes em decimal e binário.

Tabela 4.1: Relação dos TPDUs

Valor	Significado	Binário
14	CR – Connection Request	1110
13	CC – Connection Confirm	1101
8	DR – Disconnect Request	1000
15	DT – Data	1111
1	ED – Expedited Data	0001
Outros	Reservados	

Na figura 4.14 é exemplificado o formato dos quadros Connection Request e Connection Confirm pertinentes à comunicação que deve ser estabelecida entre as máquinas participantes da conferência.

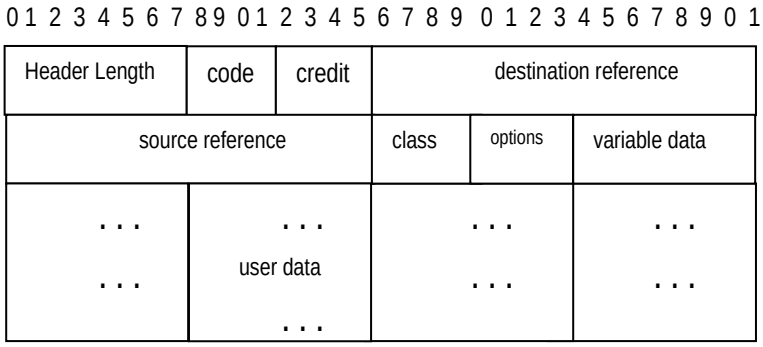

Figura 4.14: Formato dos quadros CR e CC

O formato do quadro exibido na figura anterior é o mesmo para o Connection Request e Connection Confirm, o que muda são valores de campos, como por exemplo, o campo *code* no Connection Request possui o valor 1110 (correspondente em binário) e no Connection Confirm, 1101.

O campo *credit* deve possuir sempre valor zero na saída (envio de quadro) e o mesmo é ignorado na entrada (recepção). Os campos *destination reference* (o valor no envio é sempre zero e na recepção o valor é ignorado) e *source reference* possuem o tamanho de 16 bits.

O campo *class* possui 4 bits, com o valor fixo em binário de 0100 no envio, porém na recepção ele é ignorado. O *option* também possui 4 bits, porém o valor fixo desses bits é 0000 no envio e é ignorado na recepção.

No campo *variable data* (com 6 octetos), existe a definição do tamanho da variável TPDU. Por fim, o campo *user data* possui o tamanho de 5 octetos, o valor deste campo depende do tipo de primitiva, por exemplo, a TPDU Data possui informações que devem ser direcionadas para o MCS, ao passo que o Connection Confirm e o Connection Request não.

Na figura 4.15, é ilustrado a estrutura que é enviada pelo protótipo T.127 *lite* , a fim de iniciar a conexão na camada T.123. O valor relacionado a cada variável tem o objetivo de gerar o TPDU Connection Request, que é o pedido de conexão no nível do T.123 solicitado para a máquina de destino.

```
1  package test;
2
3
4  public interface ConnectionRequestPackage {
5
6    public final static int VRSN         = 0x03; // --> 03
7    public final static int RESERVED     = 0x00; // --> 00
8    public final static int PKTLEN       = 0x00; // --> 00
9    public final static int _PKTLEN      = 0x19; // --> 11 packet length
10   public final static int  LI          = 0x14; // --> 06
11   public final static int CODE         = 0xE0; // --> 14 code
12   public final static int DST          = 0x00; // --> 00 dst
13   public final static int _DST         = 0x00; //--> 00  dst
14   public final static int SRC          = 0x04; // --> 00 dst
15   public final static int _SRC         = 0xF4; //--> 00  dst
16   public final static int CLASS_OPTION = 0x00; // --> 00 class_option
17
18   public final static int C_REQUEST12  = 0XC0; // C0
19   public final static int C_REQUEST13  = 0X01; // 01
20   public final static int C_REQUEST14  = 0X0D; // 0D
21   public final static int C_REQUEST15  = 0XC1; // C1
22   public final static int C_REQUEST16  = 0X09; // 00
23   public final static int C_REQUEST17  = 0X00; // 00
24   public final static int C_REQUEST18  = 0X00; // 00
25   public final static int C_REQUEST19  = 0X05; // 05
26   public final static int C_REQUEST20  = 0X00; // 00
27   public final static int C_REQUEST21  = 0X14; // 14
28   public final static int C_REQUEST22  = 0X7B; // 7B
29   public final static int C_REQUEST23  = 0X02; // 02
30   public final static int C_REQUEST24  = 0X01; // 01
31   public final static int C_REQUEST25  = 0X00; // 00
32 }
```

Figura 4.15: TPDU Connection Request enviada

Ao observar a figura anterior, no campo denominado de CODE (linha 11), foi atribuído o valor 14. Conforme o que foi colocado na tabela 4.1, trata-se do valor correspondente ao TPDU Connection Request.

Antes de ser enviada a TPDU Connection Request, foi executado e configurado o analisador de protocolos Observer (OBSERVER, 2004), a fim de capturar os pacotes que vão trafegar na rede. A figura 4.16, apresenta o resultado da monitoração efetuada no momento do envio da TPDU Connection Request pelo protótipo T.127 *lite* .

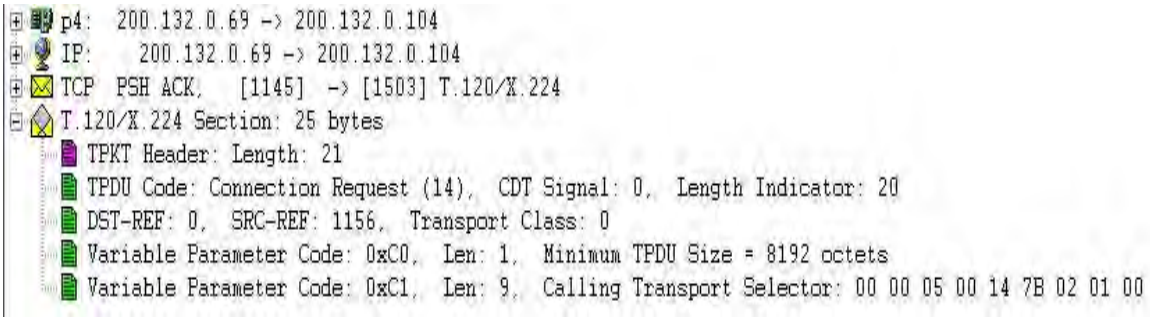

Figura 4.16: Connection Request enviado pelo protótipo

Observando a figura 4.17, na opção TPDU Code (linha 6), é verificado que foi enviada uma TPDU Connection Request. A máquina que recebeu a TPDU pode rejeitar ou aceitar a informação. No momento em que a máquina aceita a TPDU, ela deve enviar uma resposta com a TPDU Connection Confirm. A figura 4.16 apresenta um pacote capturado pelo analisador de protocolos Observer, que contém a resposta que foi enviada pelo NetMeeting à máquina que remeteu o TPDU Connection Request.

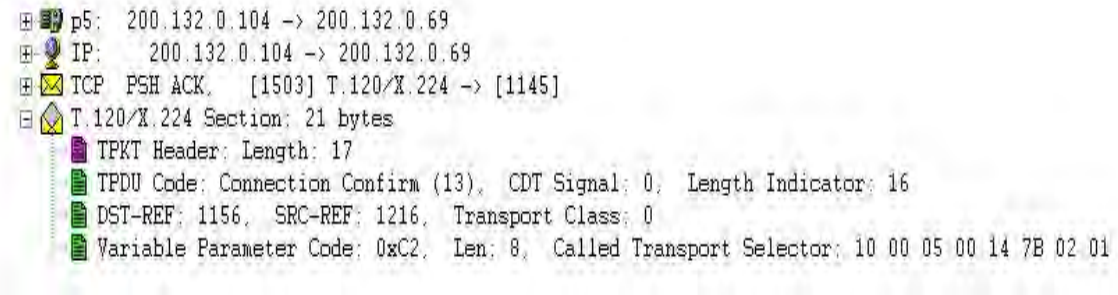

Figura 4.17: Connection Request enviado pelo NetMeeting

O protótipo T.127 *lite* deve ter a possibilidade de aceitar receber a TPDU Connection Request e a partir deste recebimento, responder através do envio do TPDU Connection Confirm. A figura 4.18 apresenta a condição onde o protótipo verifica o tipo de TPDU recebida e realiza a decodificação dos bits recebidos, dependo da análise, o programa determina qual TPDU deve ser enviada.

```
54        System.out.println( "[ 3 ] " + t );
55         // Converte o valor byte para hexa.
56         c = Parse.toDecimal(Integer.toBinaryString(k));
57
58
59        switch ( c )
60        {
61
62          case 1:
63             System.out.println("######### Client ***  case 1 ***");
64           break;
65
66          case 13:
67             System.out.println("######### Client ***  case 13 ***");
68             System.out.println("Dado no Servidor");
69             out.write(DT.senData());
70           break;
71
72          case 14:
73            System.out.println("######### Client **** case 14 ***");
74            //inBytes[ 5 ] = 13;
75            out.write(ConnectionConfirm.CC(inBytes, t));
76            System.out.println("######### Client **** Conexão estabelecida");
77            System.out.println(++i);
78           break;
79
80          case 15:
81            System.out.println("######### Client *** case 15 ***");
82            inBytes[ 5 ] = 13;
83           byte[] b = DecodificaDT.DecodificaDaTa(inBytes);
84            singleText.setTextTOJTextArea(DecodificaDT.transformaDaTa(b));
85           break;
86
87        default:
```

Figura 4.18: Decodificação do Connection Request feita pelo protótipo

A idéia do protótipo receber a TPDU Connection Request, identifica que a solução *T.127 lite* deve funcionar tanto como um cliente que solicita uma conexão, como também como servidor, recebendo o pedido de conexão e confirmando-o. Ao capturar o tráfego através do Observer, é verificado que o protótipo envia uma TPDU Connection Confirm, tanto para uma máquina com NetMeeting como para uma segunda máquina com o protótipo T.127 sendo executado (Figura 4.19).

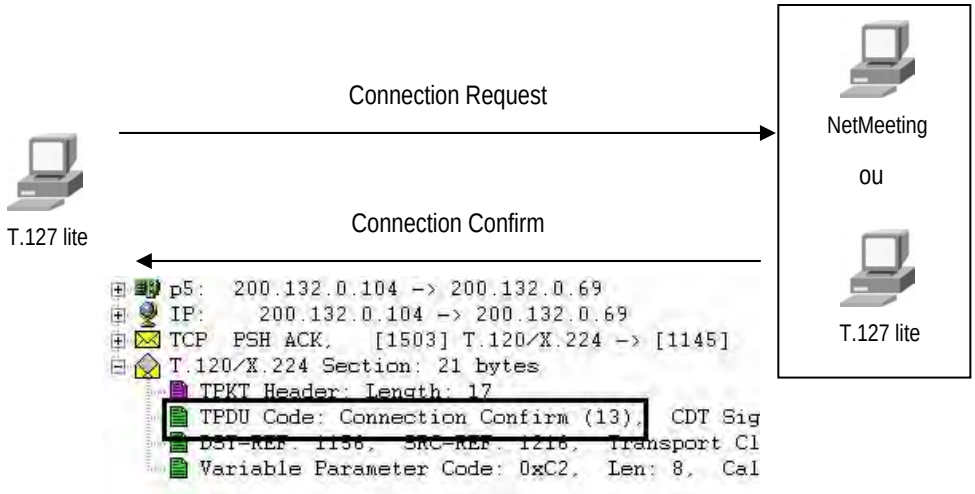

Figura 4.19: Connection Confirm enviado pelo NetMeeting ou T.127 *lite*

O protótipo *T.127 lite* ao verificar o pacote recebido e encontrar o valor 14, constata que foi enviado um Connection Request e deve ser enviado por sua vez, um pacote contendo o código 13, que corresponde ao Connection confirm, conforme é destacado na figura anterior. A partir deste ponto, já pode-se dizer que foi estabelecida conexão no T.123. Deste ponto em diante, o próximo pacote a ser enviado é o de código 15 (Data).

4.4.2 Comunicação na camada T.125

A partir do momento que uma máquina envia a TPDU Connection Request, a máquina que recebeu este pacote deve responder com a TPDU Connection Confirm. Deste ponto em diante, foi estabelecida a comunicação no T.123 e o próximo passo cabe a máquina que envia o Connection Request, enviar o TPDU Data.

A TPDU Data já possui parâmetros que são pertinentes ao MCS e significa que a conexão neste nível deve ser estabelecida. No que diz respeito à modelagem do T.125, entenda-se que deva ser enviado o próximo método que é responsável por iniciar a conexão no MCS, neste caso o MCS-Connect-Initial.

O protótipo utiliza a estrutura já definida sobre a TPDU Data (T.123) e encapsula informações sobre este pacote a fim de construir a primitiva MCS-Connect-Initial. A figura 4.20 representa a estrutura que é enviada para se obter um pedido de estabelecimento de conexão no nível do T.125.

```
package test;


public interface MCSInitial {

    public static final int MCS_SECTION_01 = 0x7F;// 7F
    public static final int MCS_SECTION_02 = 0x65;// 65
    public static final int MCS_SECTION_03 = 0x81;// 81
    public static final int MCS_SECTION_04 = 0xAA;// AA
    public static final int MCS_SECTION_05 = 0x04;// 04
    public static final int MCS_SECTION_06 = 0x04;// 04 END OF TRANSMISSION
    public static final int MCS_SECTION_07 = 0x01;// 01
    public static final int MCS_SECTION_08 = 0x00;// 00
    public static final int MCS_SECTION_09 = 0x00;// 00
    public static final int MCS_SECTION_10 = 0x10;// 10
    public static final int MCS_SECTION_11 = 0x04;// 04 END OF TRANSMISSION
    public static final int MCS_SECTION_12 = 0x04;// 04 END OF TRANSMISSION
    public static final int MCS_SECTION_13 = 0x01;// 01
    public static final int MCS_SECTION_14 = 0x00;// 00
    public static final int MCS_SECTION_15 = 0x00;// 00
    public static final int MCS_SECTION_16 = 0x10;// 10 DATA LINK ESCAPE
    public static final int MCS_SECTION_17 = 0x01;// 01
    public static final int MCS_SECTION_18 = 0x01;// 01
    public static final int MCS_SECTION_19 = 0XFF;// FF
}
```

Figura 4.20: Estrutura do MCS-Connect-Initial enviado pelo protótipo T.127 *lite*

Como o formato dos dados que são enviados através da rede está baseado no uso de uma pilha de protocolos como o T.120, ao utilizar o analisador de protocolos Observer, é verificado que as informações pertinentes ao MCS estão encapsuladas sobre a TPDU Data, que de certa forma diz respeito ao T.123. Este fato vem a comprovar que sem essa estrutura, seria inviável, por exemplo, a comunicação do *T.127 lite* com o NetMeeting. A figura 4.21 mostra um pacote capturado pelo Observer onde é apresentado o pacote MCS-Connect-Initial.

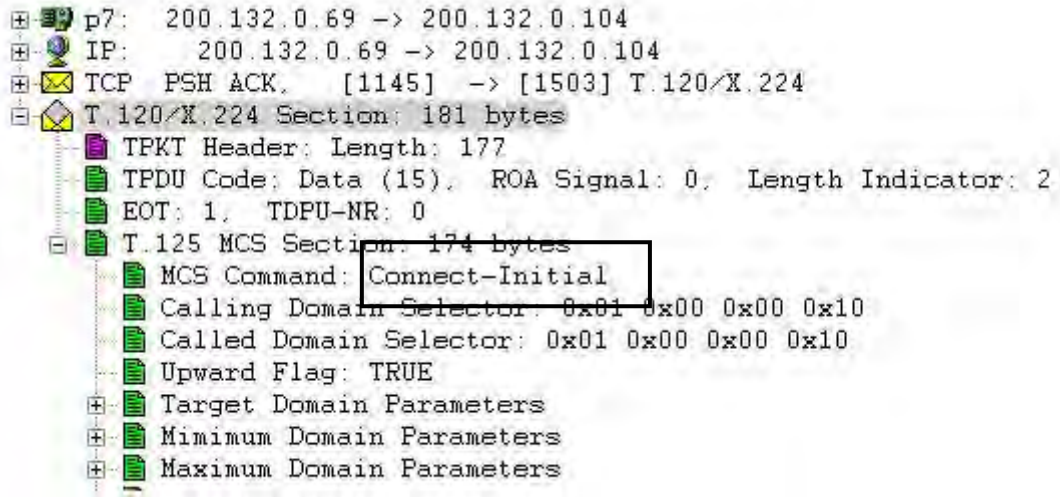

Figura 4.21: Monitoração do MCS-Connect-Initial enviado pelo protótipo

A figura anterior apresentou o pacote MCS-Connect-Initial que foi enviado pela rede através do protótipo T.127 *lite* . A partir deste ponto, a máquina que recebe o MCS-Connect-Initial deve analisar essa informação e em seguida responder através do envio do MCS-Connect-Response. O protótipo deve ser capaz de enviar e receber as duas primitivas, o que permite que a solução possa interoperar tanto com o NetMeeting por exemplo, como entre máquinas que executam o protótipo T.127 *lite* . Na figura 4.22, é ilustrado o pacote MCS-Connect-Response que foi capturado através do Observer na conferência com o T.127 *lite* .

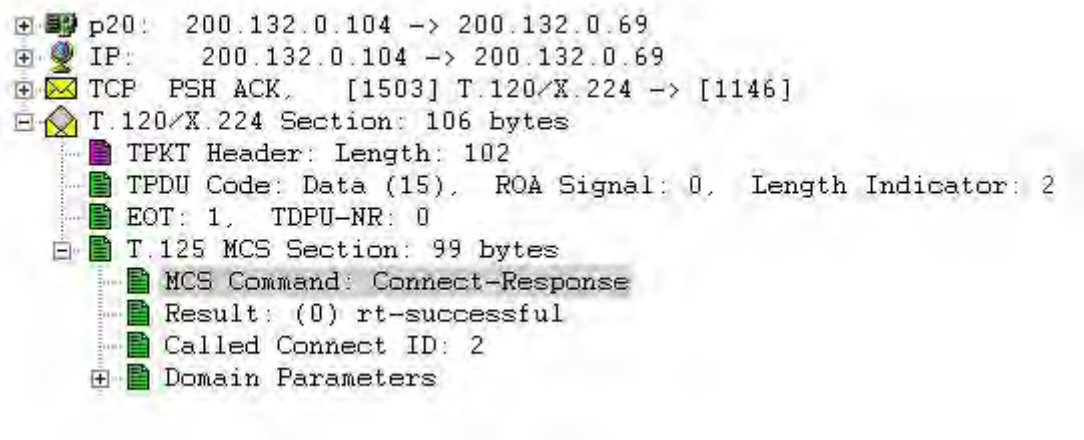

Figura 4.22: Monitoração do MCS-Connect-Response na conferência com T.127 *lite*

No momento em que é estabelecida a conexão entre as camadas MCS das máquinas participantes da conferência, devem ser trocadas mais primitivas pertinentes ao T.125 como:

- MCS: AtachUserRequest
- MCS: AtachUserConfirm
- MCS: ChanelJoinRequest
- MCS: ChanelJoinConfirm
- MCS: SendDataRequest
- MCS: UniformSendDataIndication
- MCS: SendDataIndication
- MCS: ChannelLeaveRequest
- MCS: ChannelAdmitIndication

- MCS: ChannelDisbandIndication

- MCS: DisconectPrivateUltimation

As primitivas citadas anteriormente são as necessárias para o funcionamento de uma conferência quando o T.127 for utilizado. Por tratar-se de uma solução mais simplificada no que diz respeito ao número de pacotes circulando na rede, acredita-se que essas primitivas também são as necessárias para o funcionamento de conferências com o uso T.126 e do T.128.

4.4.3 Comunicação na camada T.124

No momento que é estabelecida a conexão no T.123 através da troca das primitivas Connection Request e Connection Confirm, o MCS pode iniciar o processo de estabelecimento de conexão através do envio e recebimento de primitivas MCS-Connect-Initial e MCS-Connect-Response.

No momento em que são negociados os atributos para estabelecimento de conexão no MCS, também são negociados os valores para entrada na conferência pelo GCC. O procedimento ocorre da seguinte maneira: quando o T.125 envia a primeira primitiva que é o MCS-Connect-Initial, são encapsuladas as informações para o envio da primitiva GCC-ConferenceQueryRequest referente ao T.124. A figura 4.23 representa o pacote que é capturado através do Observer contendo as primitivas do MCS e do GCC, este pacote foi enviado através do protótipo T.127 *lite* .

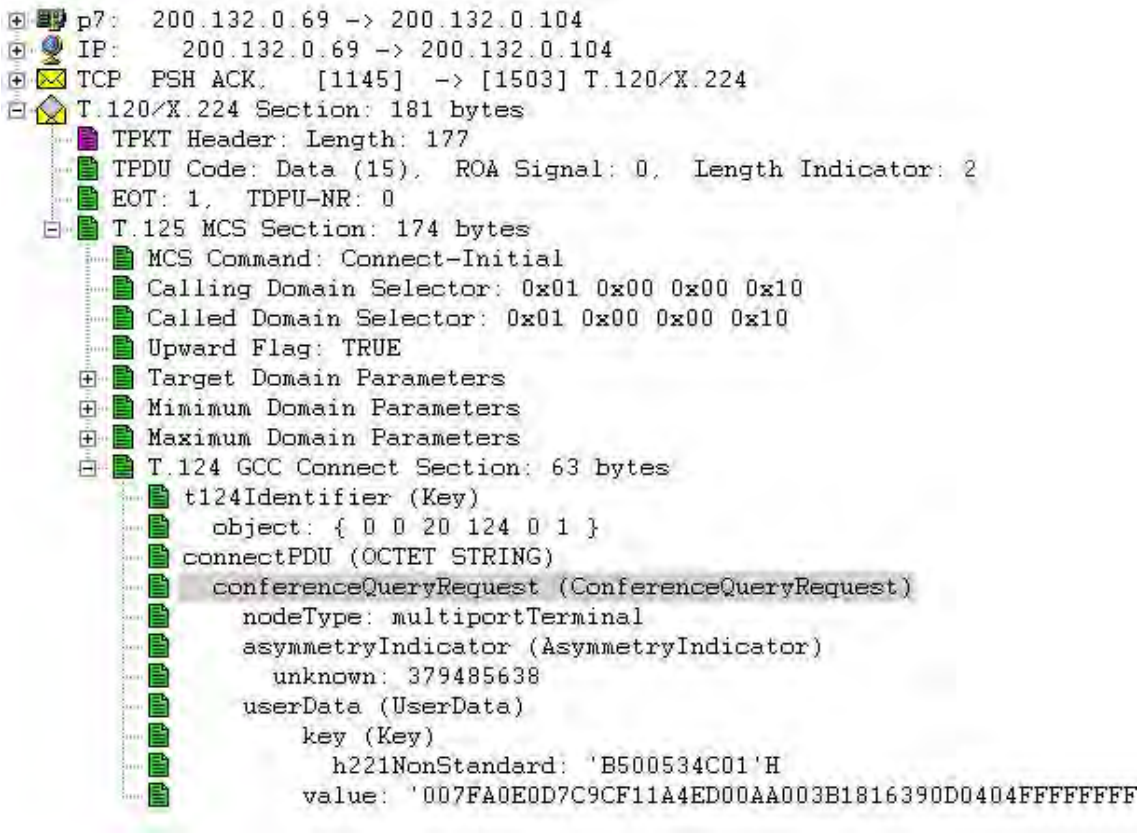

Figura 4.23: Primitivas MCS e GCC enviados pelo protótipo T.127 *lite*

O protótipo procura utilizar a mesma estrutura já definida na TPDU Data (T.123). Sobre essa estrutura são encapsulados as informações do MCS-Connect-Initial e do GCC-ConferenceQueryRequest. A utilização dessa estrutura, onde temos camadas encapsuladas, nos dá a precisão exata de que a norma foi seguida, caso contrário, na utilização em conjunto de um software como o NetMeeting não poderia ocorrer troca de dados, pois qualquer mudança na estrutura dos pacotes acaba inviabilizando o entendimento entre as camadas dos softwares participantes da conferência.

Na figura 4.24, é apresentado a estrutura que o protótipo utiliza para enviar o GCC-ConferenceQueryRequest para uma máquina que pode estar executando tanto o NetMeeting como uma cópia do protótipo T.127 *lite* .

```
1   package test;
2
3
4  public interface GCC {
5
6     public static final int GCC_01 = 0x04; // Cabeçalho do GCC
7     public static final int GCC_02 = 0x3F; // Cabeçalho do GCC
8     public static final int GCC_03 = 0x00;// 00
9     public static final int GCC_04 = 0x05;// 05
10    public static final int GCC_05 = 0x00;// 00
11    public static final int GCC_06 = 0x14;// 14
12    public static final int GCC_07 = 0x7C;// 7C
13    public static final int GCC_08 = 0x00;// 00
14    public static final int GCC_09 = 0x01;// 01
15    public static final int GCC_10 = 0x37;// 37
16    public static final int GCC_11 = 0x26;// 26
17    public static final int GCC_12 = 0x6C;// 6C
18  //public static final int GCC_13 = 0x6C;//Numero sequencial valido
19  //public static final int GCC_14 = 0x26;//Numero sequencial valido
20  //public static final int GCC_15 = 0x59;//Numero sequencial valido
21  //public static final int GCC_16 = 0x2C;//Numero sequencial valido
22    public static final int GCC_17 = 0x01;// 01
23    public static final int GCC_18 = 0xC0;// C0
24    public static final int GCC_19 = 0x40;// 40
25    public static final int GCC_20 = 0xB5;// B5
26    public static final int GCC_21 = 0x00;// 00
27    public static final int GCC_22 = 0x53;// 53
28    public static final int GCC_23 = 0x4C;// 4C
29    public static final int GCC_24 = 0x01;// 01
30    public static final int GCC_25 = 0x28;// 28
31    public static final int GCC_26 = 0x00;// 00
32    public static final int GCC_27 = 0x7F;// 7F
33    public static final int GCC_28 = 0xA0;// A0
34    public static final int GCC 29 = 0xE0;// E0
```

Figura 4.24: Estrutura da primitiva GCC-ConferenceQueryRequest

Na figura anterior, que representa os valores das variáveis que deveram gerar o GCC-ConferenceQueryRequest, é atribuído na linha 16 o valor 26, que corresponde ao código da primitiva em questão.

Tanto o protótipo como o NetMeeting ao fazer a análise dos campos do pacote, devem responder com a primitiva GCC-ConferenceQueryResponse. O pacote capturado pelo Observer contendo a primitiva em questão é apresentado na figura 4.25

Figura 4.25: Primitiva GCC-ConferenceQueryResponse

Quando o protótipo receber um pacote GCC-ConferenceQueryResponse, deve ser capaz de detectar esse tipo de informação a fim de que possa enviar a próxima primitiva, neste caso a GCC-ConferenceInviteRequest, que é responsável em convidar uma estação a fazer parte da conferência.

A verificação do tipo de pacote não pode ser no mesmo nível do GCC, mas sim no MCS, pois a primitiva GCC-ConferenceQueryResponse está encapsulada dentro da estrutura definida como MCS-Connect-Response. Em suma, devem ser analisadas a estrutura, a procura do valor 66 em hexadecimal, que é representado pelo valor 102 em decimal. No momento em que o código procurado é encontrado, o programa pode encapsular as informações pertinentes ao GCC-ConferenceInviteRequest dentro do MCS-Connect-Initial e enviar para a máquina de destino. A figura 4.26 representa parte do código que tem a função de detectar o tipo de primitiva.

```
112
113             case 15:
114
115      //varre o inBytes para verificar se é um MCS Connect-Response
116      for ( int i = 0; i < inBytes.length; ++i)
117       if (inBytes[i] == 102) {
118          System.out.println("#####!!!!!! TCLIENTC2 -> Achado o Connect-Response " +i);
119              s.close(); //fecha conexão
120        }
121
122    singleText.setTextTOJTextArea(DecodificaDT.transformaDaTa(DecodificaDT.DecodificaDaTa(inBytes)));
123             break;
124
125             default:
126               System.out.println("Default");
127             break;
128
129           }//fim switch
130          }
131
132        }catch(Exception err){}
133      }
134
135    public void dataSingleton(String jText)
136    {
137       System.out.println( "###### TClientC2 - datasingleton" );
138      try{
139
140        SingletonSockets single = SingletonSockets.getInstance();
141
142        System.out.println( "###### TClientC2 - datasigleton - single.getSocket " );
143
144       ArrayList arrList = single.getSocket();
145
```

Figura 4.26: Condição que verifica se o pacote é MCS-Connect-Response

Observando a figura 4.26, na linha 117, é realizada uma comparação da variável inBytes com o valor 102. A variável inBytes trata de uma variável multidimensional (matriz) que contém os bytes referentes ao pacote MCS-Connect-Response. Por fim, como comentado no parágrafo anterior, o valor 102 é o correspondente a 66 em hexadecimal, que foi descoberto através de monitorações realizadas com o Observer. A estrutura de como o protótipo deve enviar a próxima chamada, que é o GCC-Invite-Request, é apresentada na figura 4.27.

```
138
139
140  //**********GCC-INVITE-REQUEST*********
141
142
143  public static final int GCC_01 = 0x00;
144  public static final int GCC_02 = 0x05;
145  public static final int GCC_03 = 0x00;
146  public static final int GCC_04 = 0x14;
147  public static final int GCC_05 = 0x7C;
148  public static final int GCC_06 = 0x00;
149  public static final int GCC_08 = 0x01;
150  public static final int GCC_09 = 0x80;
151  public static final int GCC_10 = 0xB1;
152  public static final int GCC_11 = 0x67;
153  public static final int GCC_12 = 0x68;
154  public static final int GCC_13 = 0x00;
155  public static final int GCC_14 = 0x10;
156  public static final int GCC_15 = 0x13;
157  public static final int GCC_16 = 0x50;
158  public static final int GCC_17 = 0x65;
```

Figura 4.27: Estrutura do GCC-Invite-Request

Como a estrutura do GCC-Invite-Request é um pouco extensa, foram apresentadas na figura anterior somente algumas linhas do código com os valores atribuídos a essa estrutura. A monitoração realizada na conferência através do Observer mostra o MCS-Connect-Initial com a primitiva GCC-Invite-Request encapsulada (figura 4.28).

```
T.120/X.224 Section: 306 bytes
  TPKT Header: Length: 302
  TPDU Code: Data (15),  ROA Signal: 0,  Length Indicator: 2
  EOT: 1,  TDPU-NR: 0
  T.125 MCS Section: 299 bytes
    MCS Command: Connect-Initial
    Calling Domain Selector: 0x02 0x00 0x00 0x00
    Called Domain Selector: 0x02 0x00 0x00 0x00
    Upward Flag: FALSE
    Target Domain Parameters
    Mimimum Domain Parameters
    Maximum Domain Parameters
    T.124 GCC Connect Section: 186 bytes
      t124Identifier (Key)
        object: { 0 0 20 124 0 1 }
      connectPDU (OCTET STRING)
        conferenceInviteRequest (ConferenceInviteRequest)
          conferenceName (ConferenceName)
            numeric: "1"
            text: "Personal Conference"
          nodeID: 48988
```

Figura 4.28: Primitiva GCC-Invite-Request

Após a camada GCC receber de outra máquina o GCC-Invite-Request, as demais informações ou primitivas podem ser trocadas entre as máquinas e as suas respectivas camadas GCCs. Independente de quem recebeu o GCC-Invite-Request, tanto o NetMeeting como o protótipo *T.127 lite* devem responder com um GCC-Invite-Response.

O GCC-Invite-Response é uma primitiva do GCC que deve estar encapsulada dentro de uma estrutura do MCS. O pacote capturado dessa comunicação é apresentado na figura 4.29.

Figura 4.29: Primitiva GCC-Invite-Response

A figura anterior que mostra o resultado da monitoração do GCC-Invite-Response representa que o mesmo está encapsulado dentro da primitiva MCS-Connect-Response. As duas últimas figuras apresentadas recebem destaque por serem primitivas necessárias no estabelecimento da conexão tanto no nível MCS como no GCC, onde inclusive uma máquina pode convidar uma determinada estação a fazer parte da conferência. No momento em que é estabelecida a comunicação ou conexão no GCC, devem ser trocadas as demais primitivas do GCC:

- GCC-CONFERENCE-CREATE request
- GCC-CONFERENCE-CREATE indication
- GCC-CONFERENCE-CREATE response
- GCC-CONFERENCE-CREATE confirm

- GCC-CONFERENCE-JOIN request
- GCC-CONFERENCE-JOIN indication
- GCC-CONFERENCE-JOIN response
- GCC-CONFERENCE-JOIN confirm
- GCC-CONFERENCE-DISCONNECT request
- GCC-CONFERENCE-DISCONNECT indication
- GCC-CONFERENCE-DISCONNECT confirm
- GCC-CONFERENCE-TERMINATE indication
- GCC-CONFERENCE-EJECT-USER indication

4.4.4 Comunicação na camada T.127

No momento que é estabelecida a conexão no nível do MCS através das primitivas MCS-Connect-Initial e MCS-Connect-Response, a camada T.127 deve iniciar o processo de estabelecimento da conexão no seu nível.

Quando são negociados e enviados os atributos do MCS, também devem ser encapsuladas no mesmo pacote T.125 as informações pertinentes ao T.127. A primitiva que deve ser primeiramente enviada é o MCS-SendDataIndication.

O T.127 inicia o estabelecimento da conexão através da primitiva Private-Channel-Join-InvitePDU, que é um convite paras as camadas T.127 das estações participantes da conferência a iniciarem um canal privado para envio de dados. A figura 4.30 representa a primitiva Private-Channel-Join-InvitePDU encapsulada sobre o MCS.

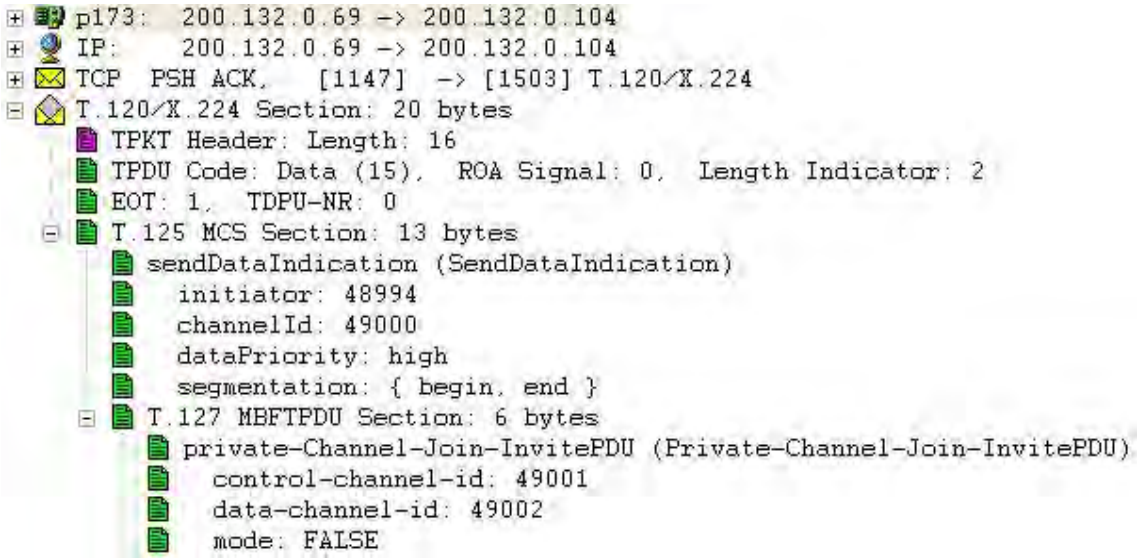

Figura 4.30: Private-Channel-Join-Invite encapsulado no MCS

O protótipo *T.127 lite* deve ser capaz de enviar ou detectar a primitiva Private-Channel-Join-InvitePDU. Desta forma, a solução pode trabalhar em conjunto com outros softwares como o NetMeeting. A estrutura que deve ser enviada através da rede referente ao Private-Channel-Join-InvitePDU é demonstrado na figura 4.31.

```
1   package test;
2
3   public interface T127_InvitePDU {
4
5
6   public static final int MBFTPDU_01  =  0x70;
7   public static final int MBFTPDU_02  =  0XBB;
8   public static final int MBFTPDU_03  =  0x80;
9   public static final int MBFTPDU_04  =  0xBB;
10  public static final int MBFTPDU_05  =  0x81;
11  public static final int MBFTPDU_06  =  0x00;
12
13  }
14
```

Figura 4.31: Private-Channel-Join-Invite PDU que deve ser enviado pelo protótipo

No momento em que é recebida a chamada Private-Channel-Join-InvitePDU por uma estação, a mesma deve responder ao convite de participar de uma sessão T.127. A resposta a primitiva Private-Channel-Join-InvitePDU deve ser Private-Channel-Join-ResponsePDU.

A primitiva Private-Channel-Join-ResponsePDU deve ser enviada e encapsulada à primitiva MCS-SendDataRequest. Tanto o protótipo *T.127 lite* quando outras soluções para colaboração visual de dados que usam o padrão T.120 devem ser capazes de receber e enviar a primitiva em questão junto ao MCS. A figura a seguir mostra a estrutura que deve ser enviada pelo protótipo a fim de gerar o pacote em questão (Figura 4.32).

```
1   package test;
2
3   public interface T127_ResponsePDU {
4
5
6   public static final int MBFTPDU_01  =  0x78;
7   public static final int MBFTPDU_02  =  0XBB;
8   public static final int MBFTPDU_03  =  0x80;
9   public static final int MBFTPDU_04  =  0xBB;
10
11  }
12
```

Figura 4.32: Private-Channel-Join-ResponsePDU encapsulado no MCS

O protótipo *T.127 lite* deve prover a recepção e o envio da primitiva File-OfferPDU que possui uma estrutura para identificar o arquivo a ser enviado. Quando o nodo T.127 identifica o arquivo a ser enviado através da primitiva em questão, cabe responder com o envio da primitiva FileAcceptPDU. O envio das duas primitivas deve ser realizado tanto pelo protótipo T.127 como pelo NetMeeting, por exemplo.

No momento em que um dos nodos da conferência recebe o FileAcceptPDU, pode-se iniciar o processo de envio do arquivo, pois nas primitivas anteriores já foram repassadas informações como identificação do arquivo, canal a ser utilizado, o nome do arquivo, sua extensão, o tamanho entre outros.

O protótipo *T.127 lite* deve ser capaz de enviar o FileStartPDU através da rede. A estrutura que deve ser enviada é apresentada na figura 4.33

```
1  package test;
2
3  public interface T127_StartPDU {
4
5  public static final int MBFTPDU_01  =  0x38;      // FileStartPDU
6  public static final int MBFTPDU_02  =  0X11;      // FileStartPDU - file header
7  public static final int MBFTPDU_03  =  0x04;      // file header
8  public static final int MBFTPDU_04  =  0x00;      // file Header
9  public static final int MBFTPDU_05  =  0x00;      // file name-Attributes
10 public static final int MBFTPDU_06  =  0x01;      // file name-Attributes
11 public static final int MBFTPDU_07  =  0x09;      // file name-Attributes
12 public static final int MBFTPDU_08  =  0x50;      // ********************* //
13 public static final int MBFTPDU_09  =  0x55;      //                       //
14 public static final int MBFTPDU_10  =  0x54;      //                       //
15 public static final int MBFTPDU_11  =  0x54;      //                       //
16 public static final int MBFTPDU_12  =  0x59;      //          value        //
17 public static final int MBFTPDU_13  =  0x2E;      //                       //
18 public static final int MBFTPDU_14  =  0x52;      //                       //
19 public static final int MBFTPDU_15  =  0x4E;      //                       //
20 public static final int MBFTPDU_16  =  0x44;      //                       //
21 public static final int MBFTPDU_17  =  0x0F;      // ********************//
22 public static final int MBFTPDU_18  =  0x30;      //                      //
23 public static final int MBFTPDU_19  =  0x31;      //                      //
24 public static final int MBFTPDU_20  =  0x30;      //                      //
25 public static final int MBFTPDU_21  =  0x32;      //                      //
26 public static final int MBFTPDU_22  =  0x30;      //                      //
27 public static final int MBFTPDU_23  =  0x34;      //                      //
28 public static final int MBFTPDU_24  =  0x32;      //  data e hora da      //
29 public static final int MBFTPDU_25  =  0x36;      //          criacao     //
30 public static final int MBFTPDU_26  =  0x31;      //                      //
31 public static final int MBFTPDU_27  =  0x33;      //                      //
32 public static final int MBFTPDU_28  =  0x33;      //                      //
33 public static final int MBFTPDU_29  =  0x33;      //                      //
34 public static final int MBFTPDU_30  =  0x32;      //                      //
35 public static final int MBFTPDU_31  =  0x38;      //                      //
36 public static final int MBFTPDU_32  =  0x5A;      //                      //
37 public static final int MBFTPDU_33  =  0x02;      // ******************** //
38 public static final int MBFTPDU_34  =  0x02;      //          tamanho     //
39 public static final int MBFTPDU_35  =  0x58;      // ******************** //
40 public static final int MBFTPDU_36  =  0x00;      //  flag                //
41 public static final int MBFTPDU_37  =  0x01;      //                      //
42 public static final int MBFTPDU_38  =  0x80;      // ******************** //
43 public static final int MBFTPDU_39  =  0x01;      //   data-offset         //
44 public static final int MBFTPDU_40  =  0x00;      //                       //
45 public static final int MBFTPDU_41  =  0x02;      // ********************//
46 public static final int MBFTPDU_42  =  0x58;      //        DATA           //
47 public static final int MBFTPDU_43  =  0x29;      //                       //
```

Figura 4.33: FileStartPDU que deve ser enviada pelo protótipo

A estrutura da primitiva FileStartPDU, que foi identificada na figura anterior, mostra alguns valores que devem ser comentados. Da linha 9 até a linha 21 da figura, os valores que são atribuídos nas variáveis dizem respeito ao nome do arquivo. Antes do envio, essa informação é exibida ao usuário que, por sua vez, pode aceitar o nome ou alterar. Por exemplo, o arquivo a ser enviado chama-se “teste.doc”, o usuário pode antes de receber o arquivo, definir o nome como “teste.html”.

Os valores referentes ao dia e hora da criação do arquivo também são repassados através das linhas 22 até a 36. Como é realizada uma transferência do arquivo, mesmo alterando o seu nome na hora do recebimento, é necessário ter conhecimento sobre quando este foi criado.

Também é necessário repassar a informação para a máquina de destino sobre o tamanho físico do arquivo, a informação sobre esse item é representada na figura 4.20, nas linhas 37 a 39.

A informação mais pertinente ocorre a partir da linha 45 que está relacionado com a própria transferência do arquivo (campo DATA), ou seja, é passado o conteúdo do arquivo, a sua estrutura, enfim, é o próprio arquivo que está sendo enviado. A quantidade de linhas a ser enviada pela estrutura está diretamente relacionada com o tamanho do arquivo, quanto maior ele for, maior o número de variáveis que a estrutura deve conter. A figura 4.34 apresenta o pacote capturado onde é transferido o arquivo de nome TESTE.TXT.

Figura 4.34: Primitiva FileStartPDU que deve ser gerada pelo protótipo

Quando o protótipo enviar a estrutura FileStartPDU, deve ser gerado o pacote conforme apresentado na figura 4.33. Os campos filename-Attribute e value são campos pertinentes ao nome do arquivo (“TESTE.TXT”). O campo data, conforme comentado no parágrafo anterior, é o próprio arquivo sendo enviado.

Caso uma nova transferência de arquivos ocorra, são executadas e enviadas as mesmas primitivas pelo protótipo. Se o usuário não desejar negociar mais nenhum arquivo, o processo (ou conexão) deve ser finalizado através da camada MCS. Com o envio da primitiva MCS-ChannelDisbandIndication. A máquina que receber a primitiva em questão deve responder com MCS-DisconnectProviderUltimatum, finalizando a conexão entre as máquinas.

O protótipo deve ser capaz tanto de receber a primitiva MCS-ChannelDisbandIndication, como também enviá-la, o mesmo procedimento vale para a primitiva MCS-DisconnectProviderUltimatum. A capacidade do protótipo enviar e receber uma mesma mensagem é fundamental para tornar a solução compatível com outras versões de aplicativos baseados no T.120.

4.5 Monitorações realizadas

No momento em que as estruturas ou chamadas eram definidas e implementadas no protótipo T.127 *lite* , era necessário verificar o comportamento dessas primitivas com outros aplicativos e até mesmo com o próprio protótipo.

A partir do envio das estruturas, foi necessário validar essas informações, e a melhor forma de comprovar o comportamento do protótipo T.127 *lite* foi através de monitorações nos diferentes tipos de comunicação estabelecidas.

4.5.1 Comunicação entre máquinas executando o protótipo T.127 *lite*

Ao executar o protótipo *T.127 lite* e informar o endereço IP da máquina que deve ser estabelecida à conexão, o protótipo envia as estruturas definidas e pertinentes ao T.123. Essa estrutura conforme já comentado anteriormente é o TPDU Connection Request. A figura 4.35 mostra a interface do protótipo *T.127 lite* onde deve ser digitado o IP da máquina que deve estabelecer a comunicação

Figura 4.35: Protótipo T.127 7iniciando uma conexão

No momento em que o usuário acessa no programa com o mouse o botão “Conectar”, é enviado a TPDU Connection Request para o endereço IP 200.132.0.69. A outra máquina que também executa o protótipo T.127 *lite* deve interceptar o pacote, analisá-lo e, em seguida, responder através de um TPDU Connection Confirm.

O recebimento do TPDU Connection Confirm indica que a conexão no nível do T.123 foi estabelecida. A partir deste ponto, o protótipo deve enviar o pacote responsável por iniciar a conexão no nível do MCS, que é MCS-Connect-Initial.

Junto com o MCS-Connect-Initial também vai encapsulada na estrutura a solicitação GCC-ConferenceQueryRequest, que é a comunicação que se inicia no nível da camada GCC. A figura 4.36 mostra trechos dos pacotes capturados nesse início de conexão entre T.123 e T.125 através do Observer.

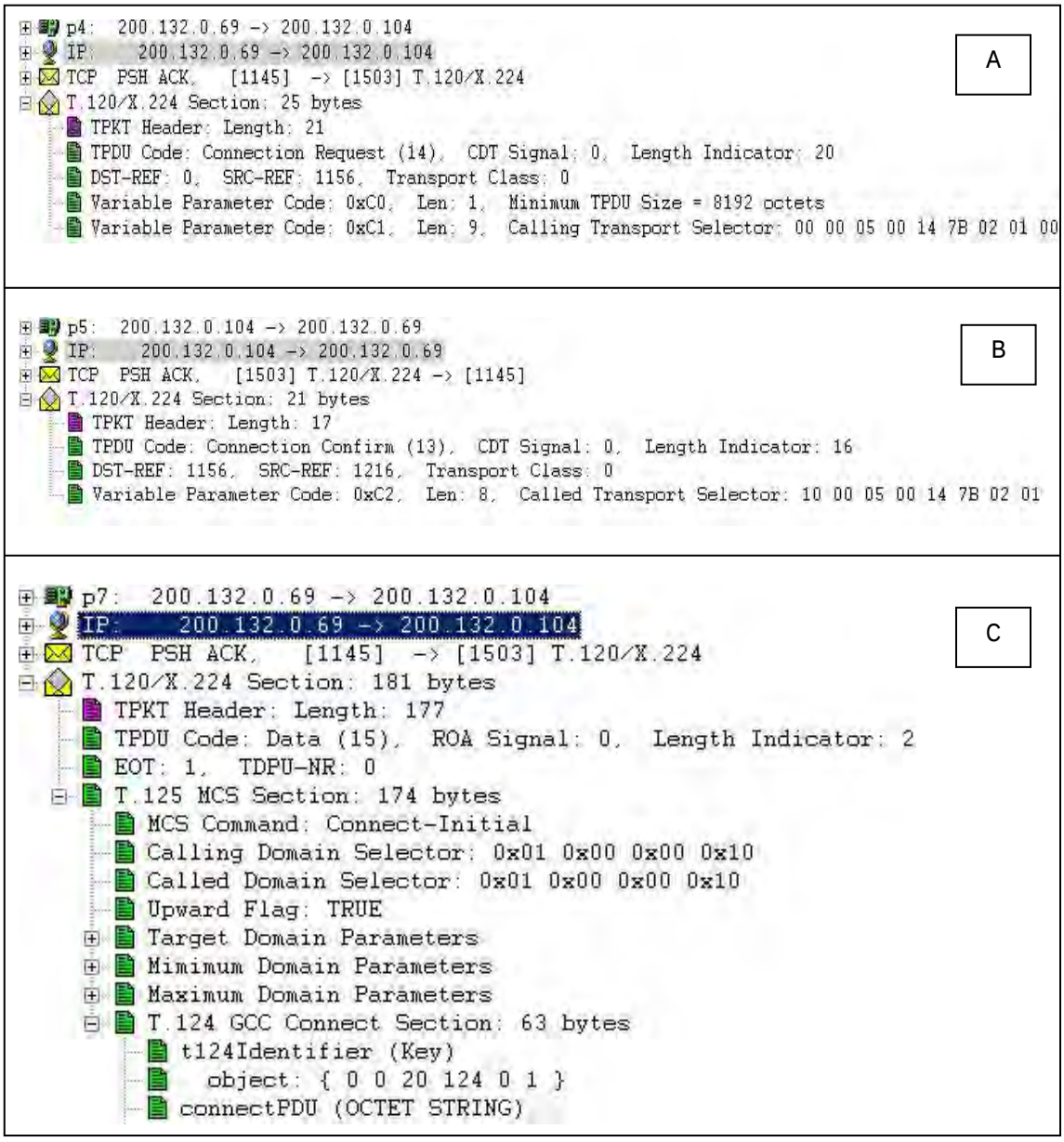

Figura 4.36: Monitoração entre os protótipos

A figura 4.36 representa os pacotes capturados no início da comunicação entre as máquinas que executam o protótipo. A letra “A” representa o estágio inicial, onde o usuário identifica o IP de origem da máquina de destino e acessa a opção “conectar”. A estrutura que é enviada neste momento é mostrada sobre forma de um pacote capturado, o Connection Request. Já a letra “B” representa a interpretação da máquina com IP 200.132.0.69, que responde ao Connection Request, enviando o pacote de resposta Connection Confirm para o IP 200.132.0.104. Na letra “C”, a máquina de IP 200.132.0.104 interpreta o Connection Confirm que recebeu; neste ponto, a estação conclui que foi estabelecida a conexão no T.123, portanto ela envia um pacote solicitando o início da conexão na camada MCS.

Com as monitorações que foram realizadas, pode-se constatar que o protótipo *T.127 lite* consegue comunicar-se com outra máquina que possui o mesmo programa executando.

4.5.2 Comunicação com o NetMeeting e o protótipo T.127 *lite*

O início da comunicação do protótipo T.127 com a máquina que possui o NetMeeting executando é semelhante ao que foi realizado no tópico 4.4.1. O programa envia uma mensagem para o IP 200.132.0.69 e é iniciado o processo de captura dos pacotes da conferência.

Ao ser analisada a monitoração, pode-se constatar que o NetMeeting enviou um TPDU Connection Confirm para a máquina que estava com o protótipo em execução. O protótipo interpretou o Connection Confirm e enviou por sua vez a estrutura MCS-Connetion-Initial responsável pelo início da comunicação no nível.

O NetMeeting por sua vez, responde enviando o pacote MCS-Connection-Response. Quando o protótipo interpreta a resposta do NetMeeting, ele envia mensagens pertinentes ao estabelecimento de comunicação no nível T.124. A figura 4.37 mostra o pacote capturado nessa comunicação.

Figura 4.37: Monitoração entre protótipo e NetMeeting

Na figura anterior, o protótipo envia um pacote ao NetMeeting convidando-o a estabelecer uma conferência no nível T.124. O procedimento adotado pelo NetMeeting é o de enviar um pacote GCC-ConferenceInviteResponse respondendo ao convite.

O procedimento de solicitação de uma conexão no nível do T.123 também pode ser iniciado pelo NetMeeting, o protótipo executará os mesmos procedimentos que foram efetuados pelo NetMeeting no início do tópico 4.4.2.

4.5.3 Comunicação entre o protótipo T.127 e o MCU

O processo de estabelecimento de conexão do protótipo T.127 com o MCU é semelhante às situações anteriores. Inicialmente para o estabelecimento de uma conexão no nível do T.123, o MCU envia o pacote Connection Request. O protótipo *T.127 lite* analisa o pacote recebido e envia ao MCU o Connection Confirm.

Quando é estabelecida a conexão no T.123, são iniciadas as trocas de pacotes para a negociação da comunicação com a camada MCS. O MCU envia o pacote MCS-Connet-Initial e um convite para ser iniciada à conferência no nível do GCC, através do GCC-ConferenceInviteRequest.

Por sua vez, o protótipo envia ao MCU o pacote MCS-Connet-Response com as informações do GCC-ConferenceInviteResponse encapsuladas. A figura 4.38 procura demonstrar através da capturada do pacote em questão, que o protótipo é capaz de interagir com o MCU.

Figura 4.38: Interação do protótipo *T.127 lite* com o MCU

A distribuição de dados em determinadas conferências pode ser realizado através do MCU. Muitas vezes, para realizar a distribuição de dados, o MCU costuma usar determinas salas, onde em cada uma podem existir vários clientes. A forma que o MCU

usa para informar suas salas é através de uma mensagem de *broadcast* que é enviada via rede.

O protótipo *T.127 lite* deve ser capaz de detectar e analisar essa informação, a fim de poder prosseguir com as trocas de dados e futuramente o envio do arquivo para uma determinada estação participante. Na figura 4.39, é apresentado um trecho do pacote de *broadcast* enviado pelo MCU as máquinas participantes da conferência

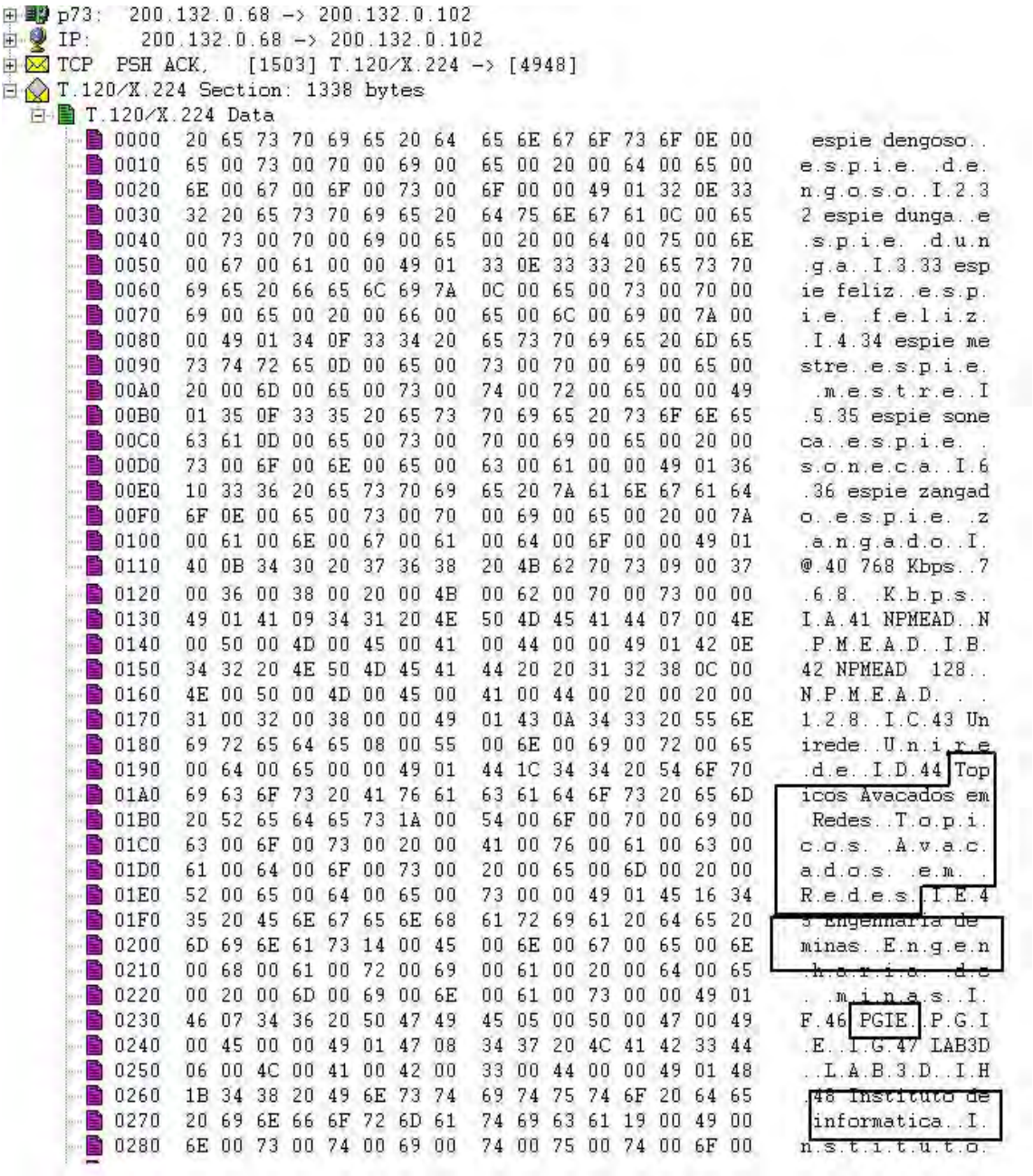

Figura 4.39: Mensagens *broadcast* enviada pelo MCU

Na mensagem *broadcast* enviada pelo MCU é apresentada uma série de salas em que as máquinas podem realizar a conferência. Na figura anterior, são destacadas algumas destas administradas pelo MCU.

Com exceção da mensagem *broadcast*, o processo de envio e recepção de pacotes do protótipo, é semelhante à comunicação com o NetMeeting e entre dois ou mais protótipos T.127 *lite* .

4.6 Considerações sobre a proposta da interface para o protótipo

O protótipo *T.127 lite* está baseado na troca e interpretação de diversos pacotes dentro de uma conferência. Interpretação porque o programa deve ser capaz de receber uma informação, analisá-la e a partir deste ponto, enviar um pacote que responda o que foi recebido, fazendo a parte de cliente e servidor.

As diversas trocas de pacotes que ocorrem, fazem com sejam estabelecidas as conexões nas camadas T.123, MCS, GCC e no T.127. As negociações no nível do T.127 têm o objetivo final de prover uma estação para que possa receber ou enviar um arquivo.

Para que o processo de estabelecimento de conexão e o envio de arquivos através da ferramenta ocorram, é necessária a interação do usuário. Portanto, é apresentada uma proposta de um modelo para a interface do protótipo T.127 *lite* . A figura 4.40, apresenta o *layout* da proposta da interface.

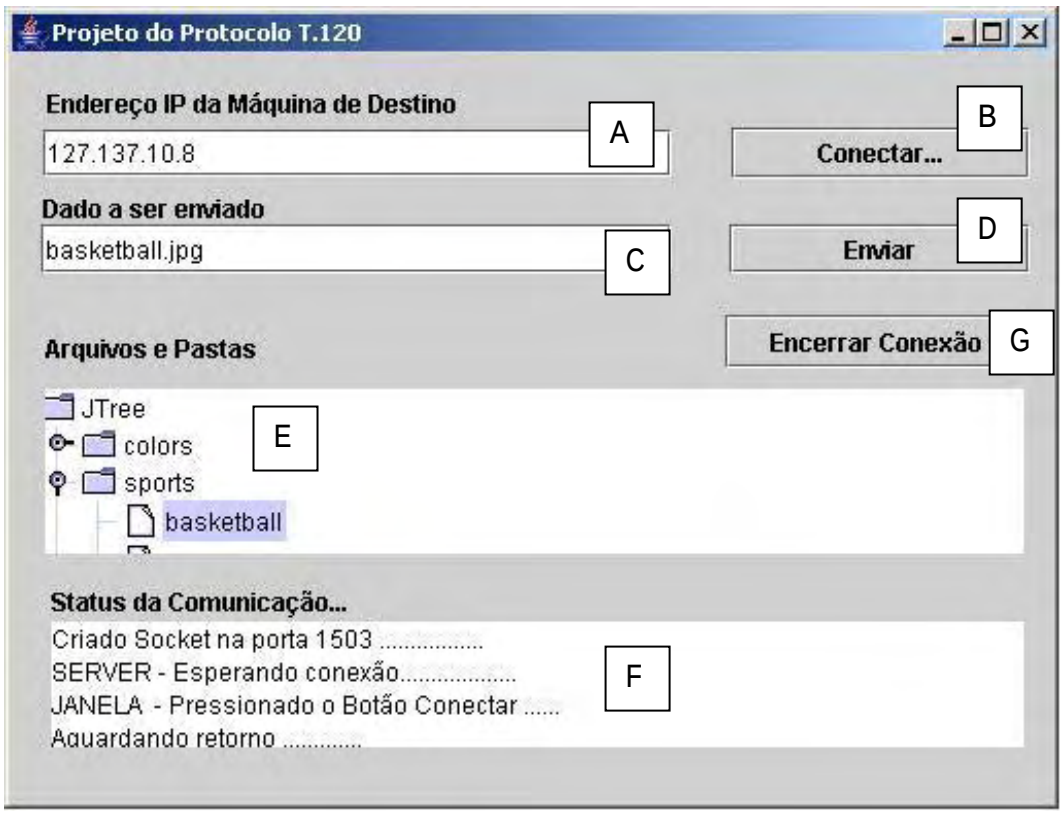

Figura 4.40: Layout do protótipo T.127 *lite*

A figura anterior mostrou a proposta de layout do T.127 onde o usuário deverá interagir. A letra “A” é o campo onde deve ser fornecido o IP da máquina em que deve ser realiza a troca de mensagens para estabelecimento da conexão nos diversos níveis da pilha T.120.

Na letra “B” é apresentada a opção do usuário iniciar a conexão com a máquina onde foi inserido o número IP. Ao acessar essa opção com o mouse ou teclado, são enviadas as estruturas e é ativada a interpretação dos pacotes recebidos pelo protótipo.

Onde aparece a letra “C”, é o campo que indica o nome do arquivo que deverá ser enviado pelo protótipo. O envio é ativado em “D”; neste ponto a primitiva FileStartPDU do T.127 é executada e o arquivo em “C” é transferido para o destino especificado em “A”.

Em “E”, o usuário identifica a origem do arquivo, e ao escolher, o mesmo é referenciado em “C”.

A opção “F” é uma forma de mostrar ao usuário o que está acontecendo em *background* na execução do programa. Neste campo podem ser exibidas diversas informações como: a criação do *socket* no momento em que a tela do protótipo torna-se visível ao usuário; a informação de que o protótipo está aguardando uma mensagem, neste caso, um pacote Connection Request.

Para a opção onde aparece a letra “G” é definida a função de encerrar a conexão, o que irá gerar as primitivas MCS-Disconect-Private-Ultimation e GCC-Conference-Terminate-Indication.

Quando o protótipo *T.127 lite* for receber um arquivo, o programa deve fornecer uma opção de informar ao usuário, se este deseja aceitar ou não o arquivo que a máquina remota deseja transferir. A figura 4.41, representa a mensagem que deve ser exibida ao usuário para informar se o mesmo deseja receber determinado arquivo

.

Figura 4.41: Mensagem referente ao recebimento de arquivo

Quando a máquina remota envia o arquivo para o protótipo, é exibida uma caixa de diálogo informando se o usuário aceita ou não o arquivo. Caso o usuário aceite; a primitiva FileStartPDU é acionada na máquina de origem do arquivo e enviado para o protótipo, este por sua vez deve apresentar uma tela com opções no estilo “salvar como”. As informações sobre taxa de transferência e tempo restante para o recebimento do arquivo podem ser apresentadas no campo “Status da Comunicação...”.

A seguir, são apresentadas as conclusões resultantes desta dissertação e os trabalhos futuros.

5 CONCLUSÕES E TRABALHOS FUTUROS

Atualmente, mesmo com uma maior popularização da videoconferência e da colaboração visual de dados, os usuários encontram problemas como o consumo dos recursos de rede, gerando problemas como a queda da conexão dos usuários que fazem parte da conferência e a dificuldade de utilizar recursos auxiliares como a transferência de arquivos.

A capacidade que os protocolos de videoconferência e colaboração visual de dados apresentam de permitir ajustes no decorrer de uma conferência, a troca de informações entre as camadas dos protocolos assim como informações de controle da conferência, influenciam diretamente no tráfego da rede, pois geram um número maior de pacotes.

A utilização de soluções proprietárias é um fator determinante na popularização da videoconferência e colaboração visual de dados, porque limitam muitas vezes a utilização de soluções e equipamentos de apenas um fabricante.

As empresas que acabam adotando uma solução proprietária têm a possibilidade de prover a diminuição da incompatibilidade entre os sistemas ou soluções de videoconferência e colaboração visual de dados e os hardwares existentes. Porém, de certa forma, também acabam influenciando na popularização da videoconferência e na colaboração visual de dados por determinar a utilização de licenças para o uso nos equipamentos.

Ao seguir a padronização T.120, a solução provou que pode ser utilizada tanto com soluções proprietárias como com código aberto, desde que estas sigam a norma T.120. O *T.127 lite* deve contribui de certa forma para que a colaboração visual de dados ganhe popularidade por disponibilizar o seu código, procurando contribuir para que novas linhas de pesquisa sobre o tema ganhem mais espaço.

O objetivo principal desta dissertação foi alcançado, que é a definição e apresentação de um modelo minimalista contendo o mínimo necessário de primitivas para o funcionamento de uma solução de transferência de arquivos nos moldes do T.127. São sugeridas simplificações no nível do MCS e do GCC, nas chamadas não pertinentes à transferência de arquivos e para a camada T.127. As simplificações influenciam na diminuição do tráfego da rede inerente à colaboração visual de dados, pois as camadas da arquitetura T.120 têm um número menor de primitivas ou pacotes a enviar.

O modelo também colabora para o melhor desempenho dos aplicativos que são executados localmente, a diminuição do número de primitivas nas camadas do T.120

ocasiona um ganho de processamento nas máquinas dos usuários e somente serão processadas as primitivas definidas no T.127 *lite* .

Para validar os estudos realizados nesta dissertação foi apresentado como o protótipo *T.127 lite* estabelece conexão nas camadas T.123, T.125 e T.124. A interoperabilidade do protótipo foi testada com o NetMeeting e o MCU, e foi comprovado através da análise do tráfego desses experimentos que a solução segue as especificações da norma T.120.

Para que ocorra o envio de primitivas para outras estações e a interpretação dos pacotes recebidos pelo protótipo T.127 *lite* , é necessária a interação do usuário com o programa. A dissertação sugeriu uma proposta de interface para o protótipo, possibilitando que o usuário interaja com outras ferramentas de colaboração visual de dados, como o NetMeeting.

Como trabalhos futuros, a solução pode ser expandida a fim de propiciar a implantação de soluções para compartilhamento de *desktop*, quadro branco e *chat*. Também pode fazer parte de outros projetos já existentes com o mesmo objetivo, como o projeto OpenH323, aliando sobre a proposta *lite* recursos de áudio e vídeo, sempre baseados em uma padronização, com código aberto e buscando resolver o problema do custo de licenças para as soluções de videoconferência e colaboração visual de dados.

REFERÊNCIAS

BALDI, M.; OFEK, Y. End-to-End Delay Analysis of Videoconferencing over Packet-Switched Networks**.** IEEE Transactions on Networking, [S.l.], v.8, n.4, Aug. 2000.

BOTIA, J.A.; RUIZ, P.; SKARMETA, A.F.G. User-Aware Videoconference Session Control using Software Agents. IEEE/WIC/ACM INTERNATIONAL CONFERENCE INTELLIGENT AGENT TECHNOLOGY, IAT, 2004, Beijing, China. **Proceedings…** Los Alamitos: IEEE Computer Society, 2004. p. 349-352.

CALYAM, P. et al. **H.323 Beacon**: An H.323 Application Related End-to-End Performance Troubleshooting Tool. Disponível em: < http://www.acm.org/sigs/sigcomm/sigcomm2004/workshop_papers/nts12-calyam1.pdf>. Acesso em: dez. 2004.

CARRION, S. **Um Modelo de Videoconferência para Computador Pessoal Orientado ao Perfil de Aplicação**. 2002. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

GAMMA, E.; HELM, R.; JOHNSON, R. **Design Patterns**. Reading: Addison-Wesley, 1995.

GIRGENSOHN, A. et al. Supporting Group-to-Group Collaboration in Videoconferences. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 35., 2002. **Proceedings…** [S.l.]: IEEE, 2002.

GLASMANN, J.; KELLERER, W.; MULLER, H. Service Development and Deployment in H.323 and SIP. In: SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 6., 2001. **Proceedings…** [S.l.]: IEEE, 2001. p. 378-385.

IMAI, S. et al. Design and Implementation of Knowledge - Based Videoconference System. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION

NETWORKING AND APPLICATIONS, AINA, 17., 2003, Xi'an, China. **Proceedings…**[S.l.:s.n.], 2003. p.105-115.

INTERNET2**.** Disponível em: <http://www.internet2.org>. Acesso em: nov. 2004.

ISO transport arrives on top of the TCP: RFC 983. Disponível em: <http://www.faqs.org/rfcs/rfc983.html>. Acesso em: jun. 2004.

ITU. **ITU-T Recommendation F.730**: Operation and Quality of Service - Videoconference Service – General. [S.1.], 1992.

ITU. **ITU-T Recommendation H.323**: Packet-based multimedia communications systems. [S.l], 1998.

ITU. **ITU-T Recommendation T.120**: Annex C Data protocols for multimedia conferencing - Lightweight profiles for the T.120 architecture. [S.l], 1995.

ITU. **ITU-T Recommendation T.120**: Data protocols for multimedia conferencing. [S.l], 1996.

ITU. **ITU-T Recommendation T.127**: multipoint binary file transfer protocol. [S.l.], 1995.

KARAHASHI, T. et al. Extension of Cooperation Protocol for a Flexible Videoconference System. In: INTERNATIONAL CONFERENCE ON INFORMATION NETWORKING, 12., 1998, Tokyo, Japan. **Proceedings…** Los Alamitos: IEEE, 1998. p34-37.

KORB, A. **Adicionando Qualidade de Serviço para um Ambiente de Colaboração Visual Baseado em H.323.** 2003. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

BAUER, M.; KORTUEM, G.; SEGALL, Z. Where are you point at? A Study of Remote Collaboration in a Wearable Videoconference System. In: INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, 3., 1999. **Digest of Papers.** [S.1.]: IEEE, 1999. p.151-158.

LEOPOLDINO, G.; MOREIRA, E. **Avaliação de Sistemas de Videoconferência**. Disponível em: <http://www.rnp.br/arquivos/AvaliacaoVideo.pdf>. Acesso em: out. 2004.

LEOPOLDINO, G. **Modelos de Comunicação para Videoconferência**. Disponível em: <http://www.rnp.br/newsgen/0105/video.shtml>.Acesso em: out. 2004.

LOGUINOV, D.; RADHA, H. **Large-scale Experimental Study of Internet Performance Using Video Traffic**. Disponível em: < http://www.acm.org/sigs/sigcomm/ccr/archive/2002/jan02/ccr-200201-loguinov.pdf>.

Acesso em: dez. 2004.

METROPOA - Rede Metropolitana de Porto Alegre. Disponível em: <http://www.metropoa.tche.br />. Acesso em: nov. 2004.

OBSERVER – Network Instruments em: <http://www.networkinstruments.com/>. Acesso em: dez. 2004.

OPENH323. Disponível em: <http://www.openh323.org />. Acesso em: nov. 2004.

ORTIZ, C. M.; GRANVILLE, L.Z.; TAROUCO, L. M. R. Aplicação que usa Protocolo de Perfil Leve para Transferência de Arquivos. In: SEMINCO, 12., 2003. **Anais...** Blumenau: FURB, 2003.

ORTIZ, C. M.; GRANVILLE, L.Z.; TAROUCO, L. M. R. Arquitetura para gerenciar e notificar a detecção de eventos em videoconferências e colaboração de dados**.** In: SIMPÓSIO DE INFORMÁTICA DO PLANALTO MÉDIO, 4., 2003, Passo Fundo. **Anais…** Passo Fundo: UPF, 2003. 1CD-ROM.

ORTIZ, C. M.; GRANVILLE, L.Z.; TAROUCO, L. M. R. Modelo de arquitetura para gerência de notificação e detecção de eventos em videoconferências e colaboração visual de dados. In: INTERNATIONAL CONFERENCE WWW/INTERNET, IADIS, 2003, 5., Lisboa. **Proceedings…** Lisboa: International Association for Development of the Information Society: 2003.

ORTIZ, C. M.; GRANVILLE, L.Z.; TAROUCO, L. M. R. Uma arquitetura para gerência de notificação e detecção de eventos em videoconferências e colaboração visual de dado. In: CONFERENCIA LATINOAMERICANA DE INFORMATICA, CLEI, 29., 2003, La Paz. **[Articulos]**. La Paz: Universidade Mayor de San Andrés, 2003.

ORTIZ, C. M.; GRANVILLE, L.Z.; TAROUCO, L. M. R. Uma solução de protocolo com perfil leve para colaboração visual de dados que utiliza transferência de arquivos com T.127. In: INTERNATIONAL CONFERENCE WWW/INTERNET, IADIS, 2003,

5., Lisboa. **Proceedings…** Lisboa: International Association for Development of the Information Society: 2003.

ORTIZ, C. M.; GRANVILLE, L.Z.; TAROUCO, L. M.,R. Um Protocolo *Lite* para Colaboração Visual de Dados que utiliza Transferência de Arquivos com T.127. In: SIMPÓSIO DE INFORMÁTICA DO PLANALTO MÉDIO, 9., 2003, Passo Fundo. **Anais…** Passo Fundo: UPF, 2003. 1CD-ROM.

PINHEIRO, C. **Tutorial – Especificação ITU H.323**. Disponível em: <http://penta2.ufrgs.br/h323/indroducao.html>. Acesso em: out. 2004.

POON, S.M. et al. Performance of buffer-based request-reply scheme for VoD streams over IP networks. **Computer Networks**, [S.l.], v.34, n.2, p.229-240, Aug. 2000.

RNP - Rede de Nacional Ensino e Pesquisa. Disponível em: <http://www.rnp.br/>.Acesso em: nov. 2004 .

SAJAL, K. et tal. Performance Optimization of VoIP Calls over Wireless Links Using H.323 Protocol. **IEEE Transactions on Computers**, [S.1.], v.8, n.08, June 2002.

VIDEOCONFERENCING - Videoconferencing Cookbook Version 3.0. Disponível em: <http://www.videnet.gatech.edu/cookbook>.Acesso em: dez. 2004.

VRVS - Virtual Rooms VideoConferencing System. Disponível em: <http://www.vrvs.org>. Acesso em: nov. 2004.

ZANIN, F.A. **Um Modelo para Videoconferência sobre Redes IP**. Erechim: EdiFAPES, 2001. 192 p. (Pensamento Acadêmico, 13).