UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Avaliação Experimental do Protocolo DCCP para Transmissão de Conteúdos Multimídia em Redes Sem Fio 802.11g e na Internet

Leandro Melo de Sales

Campina Grande, Paraíba, Brasil
Abril de 2008

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Avaliação Experimental do Protocolo DCCP para Transmissão de Conteúdos Multimídia em Redes Sem Fio 802.11g e na Internet

Leandro Melo de Sales

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Informática do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação (MSc).

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Redes de Computadores e Sistemas Distribuídos

Angelo Perkusich
Orientador
Hyggo Oliveira de Almeida
Orientador

Campina Grande, Paraíba, Brasil
Abril de 2008

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

S163a Sales, Leandro Melo
2008 Avaliação Experimental do Protocolo DCCP para Transmissão de Conteúdos Multimídia em Redes Sem Fio 802.11g e na Internet / Leandro Melo de Sales - Campina Grande, 2008
148f.: il. col.

Dissertação (Mestrado em Computação) - Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.
Orientadores: Prof. Dr. Angelo Perkusich e Prof. Dr. Hyggo Oliveira de Almeida

1– Datagram Congestion Control Protocol 2– Controle de Congestionamento
3– Protocolo de Rede e Experimentos 4– Redes de Computadores e Internet
I. Título.

CDU 004.728.731.75

"AVALIAÇÃO EXPERIMENTAL DO PROTOCOLO DCCP PARA TRANSMISSÃO DE CONTEÚDOS MULTIMÍDIA EM REDES SEM FIO 802.11G E INTERNET"

LEANDRO MELO DE SALES

DISSERTAÇÃO APROVADA EM 30.04.2008

PROF. ANGELO PERKUSICH, D.Sc
Orientador

PROF. HYGGO OLIVEIRA DE ALMEIDA, D.Sc
Orientador

PROF. MARCO AURÉLIO SPOHN, Ph.D
Examinador

PROF. ANTONIO MARCUS NOGUEIRA LIMA, Dr.
Examinador

CAMPINA GRANDE – PB

Resumo

Nesta dissertação são apresentados os resultados da avaliação experimental do protocolo DCCP (RFC 4340) quando utilizado para transmitir dados multimídia em redes 802.11g e na Internet. Durante as pesquisas, analisou-se o comportamento dos fluxos de dados transmitidos utilizando o protocolo DCCP em dispositivos com recursos limitados e em desktops. A abordagem adotada foi transmitir fluxos DCCP em conjunto com outros protocolos de transporte, o TCP e o UDP. Na execução dos experimentos, foram monitoradas diversas informações relacionadas ao estado da conexão, tais como a taxa de transmissão, a quantidade de dados transmitidos e perdidos e o *jitter*. Durante a execução dos experimentos, os dispositivos conectados à rede sem fio foram submetidos à execução de hand-off, operação bastante utilizada em redes móveis. Como resultados deste trabalho, constatou-se que os protocolos TCP e DCCP compartilham a largura de banda disponível na rede de forma justa. Em contrapartida, observou-se que os fluxos UDP degradam consideravelmente os fluxos transmitidos pelos protocolos TCP e DCCP, sobretudo quando utiliza-se os algoritmo de controle de congestionamento TCP Reno e DCCP CCID-2. Além dos resultados e discussões acerca dos experimentos realizados, outras contribuições podem ser destacadas: o desenvolvimento do Embedded Phone, uma aplicação de voz sobre IP baseada no protocolo DCCP; a implementação do algoritmo de controle de congestionamento DCCP CCID-4 no núcleo do Linux; e a disponibilização e manutenção do código do protocolo DCCP para a plataforma maemo, a qual é baseada no sistema operacional Linux.

Palavras chave: Datagram Congestion Control Protocol, Controle de Congestionamento, Protocolo de Rede e Experimentos, Redes de Computadores e Internet.

Abstract

In this dissertation it is presented the results for an experimental evaluation of the DCCP protocol (RFC 4340) when used to transmit multimedia data over 802.11g networks and over the Internet. During researches, it was analyzed the behavior of transmitted data flows using the DCCP protocol running on resource limited devices as well as on desktops. The approach adopted was to transmit DCCP data flows with other protocols flows, such as TCP and UDP. On these experiments, many different parameters related to the state of the connection were monitored such as throughput, the amount of data transmitted and lost, jitter. During some of the experiments' session, the mobile devices were subjected to hand-offs operations, which has been used in mobile networks, using three wireless access points. As a result, this work shows that both TCP and DCCP share the bandwidth available in the network fairly. Nonetheless, it was also observed that the UDP data flows substantially degenerate both TCP and DCCP data flows while being transmitted through the use of TCP Reno and DCCP CCID-2 congestion-control algorithms. Besides the results and discussions around the experiments performed, other contributions of this work are highlighted as follows: the development of "Embedded Phone", a VOIP-based application using the DCCP protocol; the implementation of the DCCP CCID-4, a congestion-control algorithm as integral part of the Linux operating system; and the availability and maintenance of the DCCP protocol code for the Nokia maemo platform, which is also based on the Linux operating system.

Keywords: Datagram Congestion Control Protocol, Congestion Control, Network Protocol and Experiments, Computer Network and Internet.

Agradecimentos

Em primeiro lugar a Deus por ter me proporcionado maravilhosos e inesquecíveis momentos ao lado das pessoas que amo. A toda a minha família que indiscutivelmente me ajudou e participou de modo efetivo de todas as etapas da minha vida.

Agradeço e dedico este trabalho aos meus pais, duas pessoas extraordinárias e dedicadas que já conheci. A meu adorável pai que em inumerosos momentos soube me mostrar os caminhos a seguir, ajudando-me incondicionalmente a alcançar os meus objetivos. A melhor mãe do mundo, que soube me dar forças sempre que eu precisei. Por terem tanto amor, carinho e dedicação comigo, muitíssimo obrigado por tudo "painho" e "mainha".

Aos meus maravilhosos irmãos por serem para mim uma grande prova que a união entre as pessoas, independentemente da distância ou diferencias, pode efetivamente existir. Júnior você é umas das pessoas que tenho como exemplo de coragem e persistência, que por mais que seja difícil de conseguir um objetivo, ele sempre é possível. Thiago tenho você como uma pessoa muito especial e importante para mim, estou torcendo muito por você, principalmente no mestrado, que é trivial, né?!

À minha namorada, Karine, pelo amor e carinho que sempre me concedeu. Por inúmeras vezes me mostrar que determinadas situações da vida podem ser resolvidas de maneira muito mais simples do que pensamos. Obrigado por compreender as minhas ausências durante o mestrado. Você faz muita diferença na minha vida, nunca esqueça disso. Agradeço também a seus adoráveis pais, que incondicionamente me apoiaram durante mais esta fase que está chegando ao fim.

A todos os outros membros da família, meus tios, primos e os "agregados", que embora não tenham sido mencionados (é muita gente!), sempre terão grande importância em minha vida.

Meus agradecimentos à CAPES pelo apoio financeiro, aos professores do Departamento de Computação da UFCG, em especial aos meus orientadores Angelo Perkusich e Hyggo Oliveira de Almeida pelas inúmeras horas de dedicação, conversas, conselhos e compreensão; a Arnaldo Carvalho de Melo, Gerrit Renker e Ian McDonald, que me ajudaram sobremaneira com eficientes discussões sobre o protocolo DCCP e o núcleo do Linux. Certamente sem todas essas pessoas este trabalho não teria acontecido.

Por fim, agradeço aos meus grande amigos do "33", Felipe e Mário, aos meus outros grande amigos do laboratório Embedded, em especial Ádrian, Danilo, Fred, Glauber, Gutemberg, Kyller, Leandro (cabelo), Loreno, Marcos Fábio, Mateus, Thiago Santos, Walter e Zé Luis; e o meu sincero agradecimento aos meus dedicados alunos dos projetos E-Phone e DCCP para maemo, Erivaldo, Felipe, Heverton e Vinícius.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Capítulo 1

Introdução

"A tecnologia sem fio provavelmente acelerará o crescimento da Internet. À medida que equipamentos sem fio ficarem mais baratos, as comunicações pela Internet estarão por toda a parte, em fones de ouvido, em jogos multi-usuário, em leitoras de bilhete de metrô etc. Toda essa tecnologia motivará novas aplicações, principalmente as baseadas em tecnologias multimídia." (Charlie Perkins, criador do IP Móvel)

Segundo resultados publicados pelo Ibope em Dezembro de 2007 [IN07], no Brasil o número de internautas aumentou $6,6$ milhões em um ano, dos quais $5,8$ milhões utilizam as chamadas conexões em banda larga, como as baseadas na tecnologia ADSL (*Asymmetric Digital Subscriber Line*) [KMS95]. Com esse aumento, o Brasil atingiu a marca de 39 milhões de internautas, um aumento de 21 % desde 2006. A expectativa para 2010 é um acréscimo de 15 milhões de internautas que acessam a Internet através de banda larga no Brasil, segundo estimativas da TeleBrasil (Associação Brasileira de Telecomunicações) [Tel07].

Neste cenário de crescimento da Internet, as redes sem fio também estão sendo largamente utilizadas no âmbito comercial, onde as empresas estão adotando esta tecnologia para disponibilizar acesso aos dispositivos sem fio e oferecer seus serviços. Dentre os diversos motivos desse crescimento pode-se destacar as melhorias alcançadas na qualidade de transmissão; o surgimento de vários mecanismos para prover segurança na transmissão de dados; a possibilidade de locomoção dos usuários enquanto eles estão conectados; a comodidade de se conectar a elas (não exige qualquer tipo de cabeamento); e a diminuição dos preços dos equipamentos voltados para esse tipo de rede.

Com base nesse crescimento visível tanto no Brasil quanto em outros países do mundo [EFE07], as aplicações multimídia recebem um destaque especial, principalmente devido a popularização do acesso a Internet residencial em alta velocidade. As aplicações

como as de VoIP[1] (ex. *Skype*), Rádios Online (ex. *SHOUTcast*), Jogos em Rede (ex. *World of Craft*) e Videoconferência (ex. *CuSeeMe*) passaram a oferecer recursos sofisticados que aproximam cada vez mais as pessoas de um diálogo face-a-face, muito embora elas possam estar localizadas a centenas ou milhares de quilômetros de distância umas das outras.

Basicamente três fatores impulsionam o crescimento do uso das aplicação multimídia, e especificamente as que têm foco em redes sem fio e mobilidade. Os principais fatores são:

1. o surgimento de bibliotecas com foco no desenvolvimento de aplicações multimídia e que requerem pouco poder de processamento (o que possibilita o funcionamento em dispositivos móveis, como celulares, *Palm Tops* e *Internet Tablets*);

2. a evolução da indústria de hardware, através da produção de equipamentos cada vez menores e com maior capacidade de processamento e de armazenamento de dados; e

3. a necessidade que as pessoas têm de se comunicar considerando a relação custo/benefício, como por exemplo as aplicações de voz sobre IP [KBS+98], as quais possibilitam diminuir os gastos em ligações interurbanas e, sobretudo, as internacionais.

Contudo, os requisitos para os serviços dessas aplicações diferem de modo significativo daquelas aplicações tradicionais orientadas a dados, como a *web* e o *e-mail*. Em particular, as aplicações multimídia possuem propriedades peculiares tais como o baixo tempo de resposta, a sensibilidade ao atraso fim a fim (latência) e a variação desse atraso (*jitter*) entre os pacotes de um mesmo fluxo (*stream*) [KR06]. Tais exigências sugerem que os atuais protocolos de transporte, como o TCP (*Transmission Control Protocol*) [Com04] e o UDP (*User Datagram Protocol*) [KR06], não sejam adequados às aplicação multimídia, embora eles sejam largamente utilizados neste tipo de aplicação.

1.1 Problemática

O aumento da demanda pelos serviços baseados na Internet atrelado a evolução das aplicações multimídia, tem preocupado a comunidade científica [FHK06] no que diz respeito ao modo como os dados dessas aplicações são transmitidos nas redes e, em particular, na Internet. A combinação desses dois fatores pode ocasionar problemas nas redes tradicionais IP e principalmente nas redes IP sem fio, focos neste trabalho.

Grande parte do tráfego multimídia é transmitido utilizando o protocolo UDP, que não fornece mecanismos para controlar a taxa de transmissão de dados na rede. A ausência de controle de congestionamento na Internet e nas redes locais IP (LAN, WLAN, PAN etc.) é o principal fator que contribui para o aumento do tráfego nessas redes. A conseqüência disso

[1] VoIP: *Voice over IP*, técnica para transmissão de conversa de voz (áudio) através de qualquer rede IP.

é que dependendo do grau de congestionamento na rede, até os serviços como *web* e *e-mail* podem se tornar inutilizáveis pelos usuários.

Na prática, o protocolo UDP utiliza o máximo de largura de banda disponível, ocupando a rede de forma desordenada por não implementar nenhum tipo de controle de congestionamento de rede. Por isto, apresenta-se com altas taxas de perda de pacotes, sobretudo quando há congestionamento na rede. Além de perder uma grande quantidade de pacotes se comparado com a quantidade de pacotes que é transmitida, o UDP ainda impede o pleno funcionamento do protocolo TCP. O TCP realiza controle de congestionamento e portanto quando ocorrem eventos de perda de pacotes a taxa de transmissão tende a diminuir, voltando a aumentar gradativamente à medida que ele consegue transmitir pacotes com sucesso.

O gráfico *vazão* $\times$ *tempo* apresentado na Figura 1.1 ilustra esse comportamento. Neste gráfico são ilustrados o comportamento de fluxos TCP e UDP durante a transmissão de dados através de 1 fluxo TCP e 3 fluxos UDP durante $400\,s$. Ambos os protocolos competiram pela utilização de um canal de transmissão de uma rede sem fio padrão 802.11g.

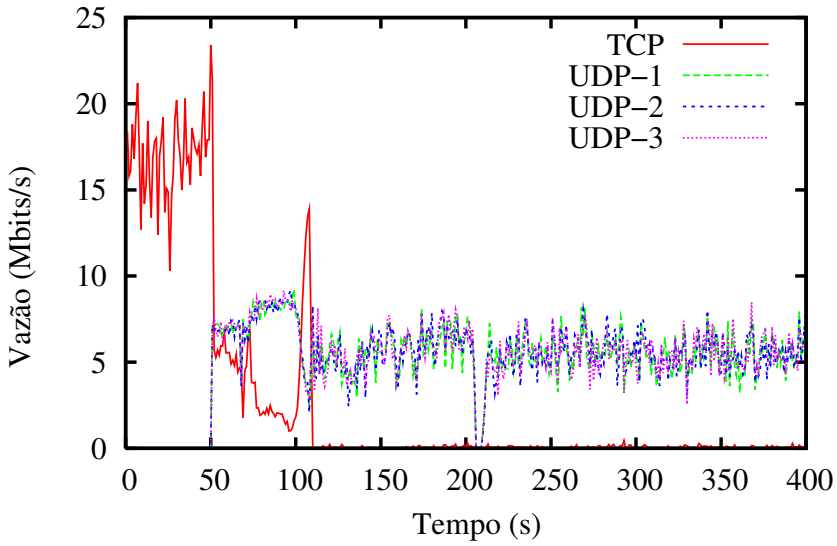


Figura 1.1: TCP $\times$ UDP. Após $50\,s$ do início do experimento, o fluxo UDP ocupa toda a largura de banda disponível na rede.

Para os primeiros $50\,s$ foi transmitido apenas o fluxo TCP, objetivando analisar o comportamento deste enquanto não competia com nenhum outro protocolo. Após os $50\,s$ iniciais, os 3 fluxos UDP foram transmitidos na rede. Nos primeiros $50\,s$ é possível constatar que a vazão do protocolo TCP foi consideravelmente melhor, variando entre $10\,Mbits/s$ e $23\,Mbits/s$. Quando os fluxos UDP foram iniciados, o protocolo TCP reduziu sua taxa de transmissão praticamente para 0. Isto ocorreu devido aos eventos de perda de pacotes provocados pelo UDP, permanecendo assim até o final do experimento.

Durante vários anos os desenvolvedores de aplicações para Internet em geral se deparavam com apenas essas duas opções (TCP ou UDP) para transportar os dados de suas aplica-

ções. O protocolo TCP é orientado à conexão e oferece confiabilidade na transferência de dados. Tais características fazem com que este seja utilizado por uma grande quantidade de aplicações de rede. Essas aplicações, denominadas também de elásticas, têm como principal característica a preferência pela confiabilidade se comparado ao tempo de resposta. Por exemplo, uma requisição HTTP (*Hyper Text Transfer Protocol*) [FGM+99] pode demorar 5 minutos e este atraso ser tolerável pelo usuário. Mas se as informações contidas na resposta forem incorretas, por exemplo, um arquivo corrompido, isto ocasiona problemas ao usuário.

No caminho praticamente inverso, o UDP vem sendo bastante utilizado em aplicações com restrições de tempo no envio e recebimento de dados, ou seja, quando o tempo de resposta é mais importante que a confiabilidade. Tais aplicações são tolerantes a perdas ocasionais de informação, mas necessitam de um fluxo contínuo de dados (*streaming*). Exceto o serviço de multiplexação e demultiplexação, o UDP não oferece nenhum serviço específico na camada de transporte, apenas utiliza o serviço de melhor esforço [Tan04] do protocolo IP (*Internet Protocol*) [Sem96] para transportar pacotes individuais de um sistema final para outro, sem garantir entrega de pacotes nem tampouco sua ordenação original.

Neste caso, existem basicamente dois cenários para as aplicações que utilizam o protocolo UDP:

1. o desenvolvedor ao utilizar o UDP, na maioria dos casos não oferece controle de congestionamento devido à complexidade de implementação; e

2. aqueles que precisam deste tipo de controle terão que implementá-lo diretamente na aplicação.

Para o segundo cenário, existe um aumento significativo da complexidade do ponto de vista de desenvolvimento, uma vez que o programador tem que implementar o serviço de controle de congestionamento na camada de aplicação. Um exemplo de tal estratégia é apresentada pelos autores do trabalho [CHM+05]. Segundo os modelos de rede, como o TCP/IP, o serviço de controle de congestionamento deve ser disponibilizado pela camada de transporte, e não pela camada de aplicação. Para estes casos ocorre uma quebra da filosofia do modelo de rede TCP/IP. Este modelo define os protocolos em camadas e determina que cada camada deve encapsular os serviços do seu nível e fornecê-los à camada superior, em vez de permitir que sejam implementados nas camadas superiores.

Com apenas essas duas opções para transporte de dados na Internet e objetivando promover melhorias nos serviços oferecidos pelas aplicações multimídia, a IETF (*Internet Engeneering Task Force*) [IET06] aprovou recentemente a especificação do protocolo DCCP (*Datagram Congestion Control Protocol*), RFC 4340 [KHF06a; KHF06b], como um dos padrões de transmissão de dados multimídia para Internet. Trata-se de um protocolo de rede localizado na camada de transporte da pilha TCP/IP, tal como o TCP e o UDP. É um protocolo orientado à conexão, não garante entrega e nem ordenação dos dados transmitidos,

porém implementa controle de congestionamento para transmissão não-confiável de fluxo de dados [SAPJ08]. Desta forma o DCCP herda do TCP as características de ser orientado à conexão e fornecer controle de congestionamento. Do UDP, o DCCP herda as características de não garantir entrega e nem ordenação dos dados transmitidos. Será visto no Capítulo 2 que além dessas características, o DCCP também utiliza dois conceitos novos: a escolha tardia de dados e um arcabouço para gerenciamento dos algoritmos de controle de congestionamento de forma modular, o que permite adicionar novos algoritmos de controle de congestionamento à aplicação e utilizá-los mesmo que uma conexão DCCP já tenha sido estabelecida.

Para entender as melhorias providas pelo protocolo DCCP, considere o gráfico *vazão* × *tempo* apresentado na Figura 1.2. Neste gráfico, ilustra-se os comportamentos dos protocolos TCP e DCCP quando utilizados para transmissão de conteúdo multimídia, similar ao caso anterior entre TCP e UDP. É possível notar que os protocolos TCP e DCCP compartilham entre si a largura de banda disponível na rede, onde cada fluxo consegue transmitir dados na rede.

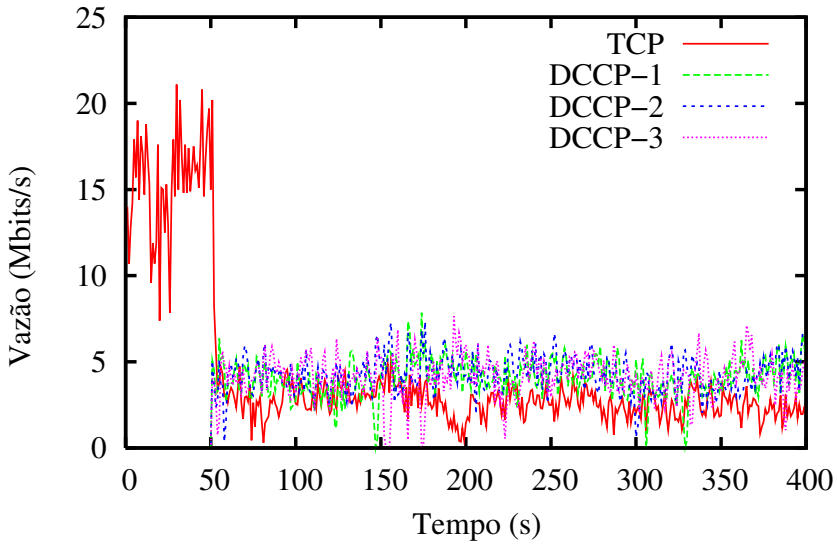


Figura 1.2: TCP × DCCP. Ambos os protocolos conseguem transmitir dados na rede.

Note que o comportamento do protocolo TCP para os 50 *s* iniciais foi parecido com o do apresentado na Figura 1.1 (TCP × UDP). Porém, diferentemente do que aconteceu naquele caso, para a transmissão TCP × DCCP a vazão do protocolo TCP foi satisfatória.

Embora serão discutidos alguns temas no Capítulo 2 que ajudarão a entender tecnicamente o conceito de escolha tardia de dados, é importante entender este mecanismo, o qual resolve outros problemas que podem ser causados pelo protocolo TCP e pelo protocolo UDP ao transportar dados multimídia em uma rede IP.

Considere a seguinte analogia: suponha o trânsito da Avenida Fernandes Lima (uma movimentada avenida da cidade de Maceió). Esta avenida é o eixo principal que dá acesso

a outras vias que ligam diversos bairros da cidade. O fluxo de veículos próximo às $8\,h$ ou próximo às $18\,h$ é muito intenso. Vários semáforos controlam o tráfego, de modo que o congestionamento de veículos seja o mínimo possível. Nenhum agente externo controla esses semáforos. Se uma ambulância precisar ter prioridade de ultrapassagem no trânsito ou precisar ultrapassar no sinal vermelho não há uma maneira para especificar qual veículo deve passar e quais devem dar passagem, a família do paciente precisará contar com a boa educação dos demais.

Para este exemplo, o trânsito controlado pelos semáforos equivale ao controle de congestionamento do protocolo TCP, onde os veículos trafegam baseando-se em regras pré-definidas: temporizadores dos semáforos e a interligação dos mesmos. A falta de um agente que permita especificar qual veículo tem maior prioridade, pode-se comparar a impossibilidade que o TCP impõe de não permitir, caso seja necessário, especificar qual pacote deve ser enviado (a urgência que a ambulância tem) e quais os que não precisam ser enviados, por exemplo, a mesma ambulância se o paciente for a óbito.

Para uma segunda situação, não seria difícil de imaginar o que aconteceria se nesses horários todos os semáforos não estivessem funcionando. Uma vez que o trânsito estaria em um total descontrole, todos os veículos iriam trafegar sem nenhum controle adequado, causando acidentes (a perda de pacotes). Este simples exemplo poderia comparar-se ao protocolo UDP, que não fornece o controle de congestionamento e como conseqüência pode gerar um aumento acentuado do congestionamento do trânsito, ou melhor, da rede.

Utilizando como base a analogia descrita anteriormente, o conceito de escolha tardia de dados funciona de modo a permitir que a ambulância ganhe passagem no trânsito considerando que ainda há esperança de salvar o paciente. Neste caso, a ambulância seria um pacote multimídia que está prestes a ser reproduzido pela aplicação de destino, e se não chegar a tempo o paciente vai a óbito, ou seja, não fará mais sentido reproduzir aquele pacote atrasado (o pacote será descartado na aplicação de destino, e portanto a rede foi utilizada para transportar um pacote inútil). Mas se existir uma outra ambulância que transporte um paciente ainda vivo (uma nova informação do conteúdo multimída sendo transmitido), é preferível transmitir este pacote a enviar o paciente em óbito, no caso, aquele pacote que não será mais útil àquela aplicação e que será portanto descartado no destino.

Se considerarmos as transmissões de dados multimídia em redes de computadores, onde satisfazer os requisitos de tempo pode definir o nível de qualidade da transmissão multimídia, o conceito de escolha tardia de dados pode melhorar a qualidade do fluxo multimídia e ainda resolver diversos problemas de congestionamento da rede, como os causados por retransmissões desnecessária de pacotes feitas pelo protocolo TCP ou por problemas de excessiva perda de pacotes quando utiliza-se o protocolo UDP.

Com base no que já foi discutido, a motivação do DCCP portanto está relacionada com as características intrínsecas das aplicações com restrição de tempo de resposta e no fato de que grande parte desse tipo de aplicação utilizar o UDP, o qual não realiza controle de con-

gestionamento. Essas aplicações, denominadas também de aplicações inelásticas [KR06], são aquelas que exigem um fluxo contínuo, em tempo real, como as que transmitem audio e vídeo. Por exemplo, nas aplicações de transmissão de voz na Internet, é preferível manter o fluxo de dados e reproduzir a voz do interlocutor, mesmo diante do fato de parte da informação ter sido perdida, a esperar que a informação perdida seja retransmitida pela aplicação que origina os dados.

Neste contexto, a retransmissão também influencia na qualidade do fluxo multimídia transmitido na rede. Este é o artifício adotado pelo TCP para garantir a entrega dos dados transmitidos na rede, o que causa um impacto direto no desempenho das aplicações multimídia. As informações tratadas por este tipo de aplicação são transientes, ou seja, elas devem ser reproduzidas no destino até um determinado instante. Passado este tempo, reenviar as informações da posição perdida pode não fazer mais sentido, uma vez que o fluxo é contínuo e sempre há novas informações para serem transmitidas. Retomando a analogia do conceito de escolha tardia de dados, é como se fosse dar passagem no transito a uma ambulância que transporta um paciente já em óbito. Além disso, caso a aplicação resolva esperar pela retransmissão da informação perdida, isto causará um atraso no fluxo contínuo de dados da aplicação, assim o usuário perceberá interrupções na reprodução daquele conteúdo sendo transmitido, muitas vezes em tempo real, como as aplicações de voz sobre IP.

Considerando os problemas e limitações dos protocolos TCP e UDP discutidos nesta seção, neste trabalho o interesse portanto é estudar uma das possíveis soluções para o seguinte problema-chave: de uma lado está o protocolo UDP, adotado por diversas aplicações com restrição de tempo e que não realizam controle de congestionamento. Do outro lado está o protocolo TCP, cujas aplicações podem se tornar inutilizáveis devido ao congestionamento causado pelo UDP, além de realizarem retransmissões para garantir a entrega de dados.

Como possíveis soluções para este dilema na escolha do melhor protocolo de transporte para conteúdos multimídia, duas opções podem ser exploradas: a primeira é substituir os equipamentos e os meios físicos de rede por outros cada vez mais eficientes, sempre que se perceber que o desempenho da rede não é favorável. Essa opção pode levar a gastos desnecessários, como apontou um estudo publicado recentemente [Ser07], o qual mencionou que a Internet pode parar de funcionar se não forem realizados investimentos em infra-estrutura de rede que giram em torno de 137 bilhões de dólares. A segunda é utilizar os recursos de rede disponíveis atualmente de forma mais apropriada e eficiente.

Uma opção para solucionar os problemas apresentados até aqui é o protocolo DCCP, o qual foi escolhido como ponto de partida para buscar uma integração dos recursos necessários exclusivamente às aplicações multimídia, principalmente àquelas executadas sobre redes IP sem fio. Além disso, permitir que as aplicações baseadas em TCP consigam transmitir seus dados na rede em paralelo aos fluxos de dados multimídia, como discutimos nesta seção através do gráfico ilustrado na Figura 1.2.

1.2 Objetivos do Trabalho

Com base nas discussões apresentadas anteriormente, o principal objetivo nesta dissertação é avaliar o desempenho do protocolo DCCP em redes IP sem fio 802.11g e na Internet. A avaliação consiste em um processo comparativo entre o DCCP e os protocolos TCP e UDP quando utilizados para transmitir fluxos multimídia nessas rede.

Como objetivo secundário figura o interesse em estudar o comportamento dos protocolos TCP, UDP e DCCP quando utilizados em dispositivos com recursos limitados (por exemplo, em celulares). Objetivamos também contribuir com soluções de implementação do protocolo DCCP no núcleo do Linux e desenvolver ferramentas que ajudem a disseminar o uso desse novo protocolo na Internet.

1.3 Relevância

A demanda por serviços multimídia em redes de computadores tem aumentado dia após dia. Desta forma, o estudo desenvolvido nesta dissertação ajudará em tomadas de decisões sobre futuros desenvolvimentos desse tipo de aplicação, pois são apresentados resultados e discussões sobre o desempenho acerca de três protocolos de transporte disponíveis para uso nas redes de computadores, sendo o DCCP o foco deste trabalho. Nesse sentido, dependendo do objetivo da aplicação a ser desenvolvida, a equipe de projeto pode fazer uso dos resultados aqui apresentados e decidir qual, ou quais protocolos dentre os estudados deverão ser utilizados na aplicação a ser desenvolvida.

Há também contribuições na linha de disseminar o uso do protocolo DCCP e disponibilizar à comunidade científica os pontos fortes e fracos para transmissão de dados multimídia em redes sem fio, principalmente do ponto de vista do protocolo DCCP. Nesse escopo, a pesquisa tem o papel fundamental de investigar os conceitos e protocolos de rede. É com esse pensamento que muitos avanços tem sido realizados nas diferentes linhas de pesquisa associadas aos vastos conceitos das redes de computadores, como tecnologias de rede sem fio, protocolos de comunicação e controle de congestionamento. Estes são conceitos chaves estudados neste trabalho. As investigações neste sentido tem duas abordagens, a baseada em simulações e a baseada em experimentos. Esse trabalho aparece, portanto, como uma contribuição que apresenta resultados experimentais envolvendo diversos conceitos importantes na área de redes de computadores, como um tema mais geral.

Outras contribuições promovidas por este trabalho podem ser destacadas, como a implementação de um algoritmo de controle de congestionamento para o protocolo DCCP no núcleo do Linux, correções de erros da implementação deste protocolo em Linux e melhorias no nível da aplicação para a API (*Application Programming Interface*) de *socket*[2] DCCP,

[2] O conceito de socket será discutido no Capítulo 2.

sendo essas melhorias já submetidas ao grupo de desenvolvimento de DCCP em Linux durante todo o tempo de desenvolvimento deste trabalho. Para essa última contribução, ela ocorreu em duas vertentes, uma no âmbito de desenvolvimento do protocolo no nível de sistema operacional e a outra oferecendo melhorias na biblioteca de *socket* DCCP considerando o ponto de vista do desenvolvedor de aplicações baseadas em DCCP.

Por fim, uma aplicação direta dos resultados e contribuições desta dissertação está acontecendo no contexto de desenvolvimento de aplicações multimídia para sistemas embarcados (Projeto Percomp) [Emb08]. Este projeto está sendo desenvolvido nas dependências do Laboratório de Sistemas Embarcados e Computação Pervasiva, localizado na Universidade Federal de Campina Grande (UFCG).

1.4 Estrutura da Dissertação

Nesta dissertação, os capítulos foram organizados de modo a minimizar a dependência entre eles. Assim, o leitor pode optar em ler apenas os pontos que lhe interessa de acordo com a seguinte estrutura:

- No Capítulo 2 são apresentados os principais conceitos relacionados à camada de transporte do modelo TCP/IP, dando ênfase nos três protocolos estudados no contexto deste trabalho, o TCP, o UDP e o DCCP, com destaque para as funcionalidades do protocolo DCCP e ao tema controle de congestionamento implementados tanto pelo TCP quanto pelo DCCP;

- No Capítulo 3 são abordados os conceitos envolvendo as aplicações multimídia executadas sobre redes sem fio, com destaque para as características desse tipo de rede, as quais podem gerar um impacto significativo na qualidade dos serviços de rede oferecidos pelas aplicações multimídia;

- No Capítulo 4 são discutidos alguns procedimentos adotados para a construção das redes sobre as quais os experimentos foram executados. Além disso, são apresentadas algumas contribuições em nível de implementação, sendo essas utilizadas para a realização dos experimentos;

- No Capítulo 5 são apresentados os métodos adotados nesse trabalho para a realização dos experimentos e os cálculos para obtenção dos valores finais de cada métrica analisada. Serão apresentados também os algoritmos de controle de congestionamento que foram utilizados, as decisões tomadas para a realização dos experimentos e os parâmetros de configuração de cada um dos protocolos de transporte;

- No Capítulo 6 são discutidos os resultados acerca dos experimentos realizados nas redes descrita no capítulo 5;

- No Capítulo 7 são apresentados os trabalhos relacionados;
- Por fim, no Capítulo 8 são apresentadas as considerações finais, discutindo resumidamente os principais tópicos elencados neste trabalho, bem como os trabalhos em andamento e futuros.

Além dos capítulos descritos anteriormente, são disponibilizados 2 apêndices. No Apêndice A, apresenta-se um guia de como habilitar e utilizar o protocolo DCCP no Linux, oferecendo detalhes de como o DCCP pode ser utilizado para desenvolver aplicações de rede. E no Apêndice B, apresenta-se um guia de como utilizar duas ferramentas usadas nos experimentos realizados nesta dissertação. Uma é a IPerf, utilizada nos experimentos para gerar e transmitir dados na rede. A outra é o GStreamer, um arcabouço multimídia baseado em componentes para processamento de dados multimídia.

Nos capítulos seguintes, quando o termo "*aplicação multimídia*" for citado, considere que são aquelas aplicações executadas sobre as redes de computadores baseadas no protocolo IP. Esta consideração se estende para o termo "*rede sem fio*", o qual refere-se às redes sem fio padrão $802.11g$, salvo quando mencionado explicitamente.

Capítulo 2

Camada de Transporte TCP/IP

"Controle de congestionamento está relacionado a como usar a rede da forma mais efetiva quanto possível. Atualmente as redes estão super equipadas e a pergunta mudou de *como eliminar o congestionamento da rede?* para *como utilizar de forma eficiente toda a capacidade disponível da rede?*" (Michael Welzl)

Como já discutido no Capítulo 1, os recentes avanços nas tecnologias de hadware para dispositivos móveis e redes sem fio têm alavancado a criação de novas aplicações multimídia com foco nessas redes. Essas aplicações podem gerar congestionamentos na rede uma vez que a grande maioria delas utilizam o protocolo UDP para transportar dados. A conseqüência disso é perceptível: o mau funcionamento de importantes aplicações executadas sobre essas redes.

Os recursos de redes utilizados para prover os serviços das aplicações multimídia nas redes de computadores, trás à tona diversos pontos conceituais, dentre eles os relacionados aos protocolos de rede de computadores, às técnicas para controle de congestionamento e de fluxo e os conceitos das redes de computadores sem fio, as quais possuem características que agravam ainda mais o desempenho das aplicações multimídia.

Como a maioria desses conceitos fazem parte da camada de transporte do modelo de referência TCP/IP, neste capítulo são apresentados os fundamentos relacionados a estes temas, principalmente no tocante aos protocolos disponíveis na camada de transporte estudados nesta dissertação: o TCP, o UDP e o DCCP. Antes, porém, é definido o que são os protocolos de rede, os quais realizam um papel fundamental para o funcionamento das aplicações que transmitem e recebem dados através das redes de computadores.

2.1 Protocolos de Redes de Computadores

De forma geral, um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de mensagens ou eventos [KR06].

Um protocolo pode ser explicado através de uma analogia do mundo real. Uma pessoa ao perguntar as horas para outra poderia abordá-la falando a palavra "Oi" e a outra pessoa também poderia responder, "Oi!". Para o protocolo humano o segundo "Oi" é uma indicação cordial de que o diálogo pode continuar. Então, neste momento, a pessoa interessada em saber as horas poderia fazer a pergunta "que horas são?". Após receber a resposta esta pessoa agradece com um "Obrigado.", obtendo do interlocutor uma resposta como "De nada.". As semelhanças são claras, a única diferença com relação aos protocolos de redes é que as entidades que trocam mensagens e realizam ações são componentes de *hardware* e *software*, em vez de pessoas. Este exemplo foi extraído da referência [KR06].

Todas as transmissões de dados na Internet que envolvem duas ou mais entidades remotas, portanto, são governadas por um protocolo. O exemplo citado, pode ser dividido em três partes:

1. o primeiro "Oi" pode ser comparado a um pedido de conexão, o segundo à aceitação ou rejeição de conexão, caso a pessoa que falou o primeiro "Oi" recebesse como resposta "Estou ocupado.";

2. após o estabelecimento da conexão, ocorre a troca de dados, uma vez que o interessado em saber as horas perguntou "Que horas são?", recebendo uma resposta compatível;

3. Após a troca de dados, acontece o encerramento da conexão através de um sinal de agradecimento ("Obrigado") e uma confirmação de aceitação de encerramento da conexão através da palavra "De nada.".

As requisições e respostas devem ser compatíveis com a especificação do protocolo. Por exemplo, quando uma pessoa perguntar "Que horas são?" ela não pode receber como resposta "Segunda-Feira.".

Diferentes tipos de protocolos são usados para realizar diferentes tarefas de comunicação. Estes protocolos são simples e diretos (conversa entre duas pessoas pra saber a hora), enquanto outros podem ser mais complexos (por exemplo, as regras e ações de um jogo de futebol).

Portanto um protocolo é dividido em cinco partes:

1. o serviço que é oferecido;

2. as hipóteses sobre o ambiente onde ele é executado, incluindo os serviços utilizados por ele;

3. o vocabulário de mensagens utilizado para implementá-lo;

4. o formato de cada mensagem no vocabulário; e

5. as regras definidas em algoritmos, as quais governam a consistência das mensagens trocadas e a integridade do serviço provido.

Os protocolos TCP, UDP e DCCP são exemplos de protocolos de rede, responsáveis especificamente por definir regras de como os dados são transportados entre dois computadores inter-conectados em rede. Eles oferecem serviços que especificam como os dados da aplicação serão encapsulados e entregues às aplicações remotas; como realizar controle de congestionamento e de fluxo (se aplicável) e se haverá ou não garantia de entrega dos dados transmitidos na rede e se esses dados serão ordenados ao chegar no destino.

2.2 Visão geral do modelo de referência TCP/IP

As grandes redes de computadores, e a Internet principalmente, são sistemas complexos e que possuem muitos componentes: inúmeras aplicações e protocolos, vários tipos de sistemas e conexões entre eles, comutadores de pacotes (por exemplo, os roteadores), além de vários tipos de meios físicos que conectam esses sistemas. Para diminuir o nível de complexidade resultante da união de diferentes tipos de arquitetura tanto de protocolos quanto de equipamentos de rede, os pesquisadores propuseram uma arquitetura de rede chamada de modelo de referência TCP/IP.

Com o objetivo de fornecer uma estrutura para projetos de protocolos de rede, estes pesquisadores dividiram a arquitetura em camadas funcionais, onde cada protocolo de rede pertence a uma dessas camadas. Essa separação tem uma importância considerável, pois provê modularização dos serviços através de várias camadas, o que torna a estrutura mais flexível para receber modificações funcionais. Por exemplo, é possível modificar a implementação de um serviço de uma camada sem afetar os serviços oferecidos pelas camadas adjacentes. A estratégia é que uma camada forneça seus serviços para a camada acima dela e utilize os serviços da camada que está abaixo dela, mantendo o restante do sistema inalterado. Este conceito é denominado de modelo de serviço de uma camada. Na Figura 2.1, ilustra-se alguns dos protocolos para cada uma das camadas do modelo TCP/IP.

O sistema de camadas de protocolos tem vantagens conceituais e estruturais. A divisão de camadas proporciona um modo estruturado de discutir componentes de sistemas, uma vez que a modularidade facilita a atualização destes componentes. Porém uma desvantagem potencial desse modelo é que uma camada pode duplicar a funcionalidade de uma camada inferior e, ainda, a funcionalidade de uma camada pode necessitar de informações que estão presentes somente em uma outra camada. Por exemplo, isso pode ocorrer se um desenvolvedor utilizar o protocolo UDP e necessite de controle de congestionamento durante as

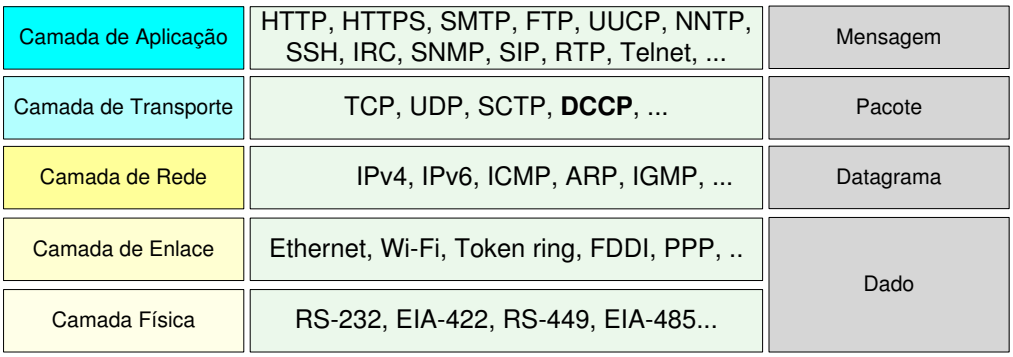


Figura 2.1: Modelo de Referência TCP/IP e Protocolos de Internet

transmissões da sua aplicação: ele terá que implementar este serviço na sua própria aplicação, o que infringe o objetivo de separação das camadas e também aumenta a complexidade da aplicação. Os problemas gerados nesses casos foram discutidos na Seção 1.1.

No caso do protocolo DCCP, isto é resolvido implementando controle de congestionamento para transmissão não-confiável de datagrama direto na camada de transporte. Isto evita que o desenvolvedor se preocupe com a complexa tarefa de implementar controle de congestionamento na camada de aplicação.

Retornando para a ilustração da Figura 2.1, o interesse neste trabalho está nos serviços providos pela camada de transporte, mas é importante mencionar os conceitos a respeito das camadas de aplicação e de rede, uma vez que a camada de transporte fornece serviços à camada de aplicação e utiliza os serviços da camada de rede.

1. **Camada de aplicações:** na camada mais alta, os usuários executam aplicações que acessam serviços de rede em um sistema remoto. Uma aplicação através de um protocolo específico (por exemplo, HTTP e FTP) interage com um dos protocolos do nível de transporte para transmitir e receber mensagens. Cada aplicação escolhe o protocolo de transporte necessário, que tanto pode ser uma seqüência de pacotes individuais (UDP, DCCP) ou um fluxo contínuo de bytes (TCP). Ao escolher o tipo de transporte, a camada de aplicação está utilizando o serviço da camada abaixo, como especifica o modelo de protocolos em camadas discutido no parágrafo anterior.

2. **Camada de rede:** responsável pela movimentação de datagramas de um sistema para outro. No sistema de origem, os protocolos da camada de transporte da Internet (TCP, UDP, DCCP etc.) passam um segmento e um endereço de destino à camada de rede. A camada de rede fragmenta cada segmento e o envia até o sistema remoto através do meio físico da rede, que pode passar por diversos comutadores existentes entre os dois sistemas. Nesta camada está definido o protocolo IP, que, dentre outros propósitos, determina os campos do cabeçalho dos datagramas transmitidos, bem como o modo com que os sistemas finais e os roteadores interpretam esses campos. Além do protocolo IP, um outro componente importante é o protocolo de roteamento, o qual determina as

rotas que os datagramas devem seguir entre a origem e o destino.

2.3 A Camada de Transporte

A camada de transporte está posicionada entre as camadas de aplicação e de rede. Esta camada desempenha um papel fundamental na arquitetura de redes em camadas. Nesta seção serão abordadas as funcionalidades providas por esta camada e uma visão geral sobre entrega confiável, controle de controle de congestionamento e controle de fluxo.

O modelo de referência TCP/IP disponibiliza pelo menos três protocolos de transporte. Um deles é o UDP, que provê à aplicação um serviço não confiável e não orientado à conexão. O segundo e o terceiro são os protocolos TCP e DCCP, onde apenas o primeiro provê à aplicação um serviço confiável, e ambos são orientados à conexão e oferecem controles de congestionamento e de fluxo.

Serviços da camada de transporte

Um protocolo de camada de transporte oferece comunicação lógica entre as aplicações (processos, no nível de sistema operacional) que estão em execução nos computadores. O conceito de comunicação lógica significa que, do ponto de vista da aplicação, tudo se passa como se os computadores comunicantes e que executam as aplicações estivessem conectados diretamente. Na prática, sabe-se que esses sistemas poderão estar em lados opostos do planeta, conectados por diversos roteadores e uma ampla variedade de tipos de meios físicos.

Devido a essa diversidade, aparece o primeiro conceito importante, o conceito de *socket* [SFR05]. Considere a seguinte analogia: uma aplicação está para uma casa e o *socket* dessa aplicação, está para a combinação do endereço da casa e a porta de entrada dessa casa. Quando um processo deseja enviar uma mensagem a um outro processo ele envia a mensagem para o endereço da casa e informa a porta de entrada. Este processo assume que existe uma infra-estrutura de transporte do outro lado da porta que transportará a mensagem pela rede (ou pelas ruas da cidade) até a porta do processo destinatário. Finalmente, ao chegar no destino, a mensagem passa através da porta do processo receptor através do *socket* de recepção.

Na Figura 2.2, ilustra-se a idéia em que os processos, representados pela aplicação na visão do usuário, comunicam-se logicamente através da camada de transporte para enviar mensagens entre si, considerando-se livres da preocupação dos detalhes da infra-estrutura física utilizada para transportar as mensagens.

Através da Figura 2.2, é possível entender que os protocolos da camada de transporte são implementados nos sistemas finais, mas não nos comutadores da rede. No sistema final de origem, a camada de transporte ao receber uma mensagem da camada de aplicação, adiciona algumas informações de controle (cabeçalho da camada de transporte) e então repassa o

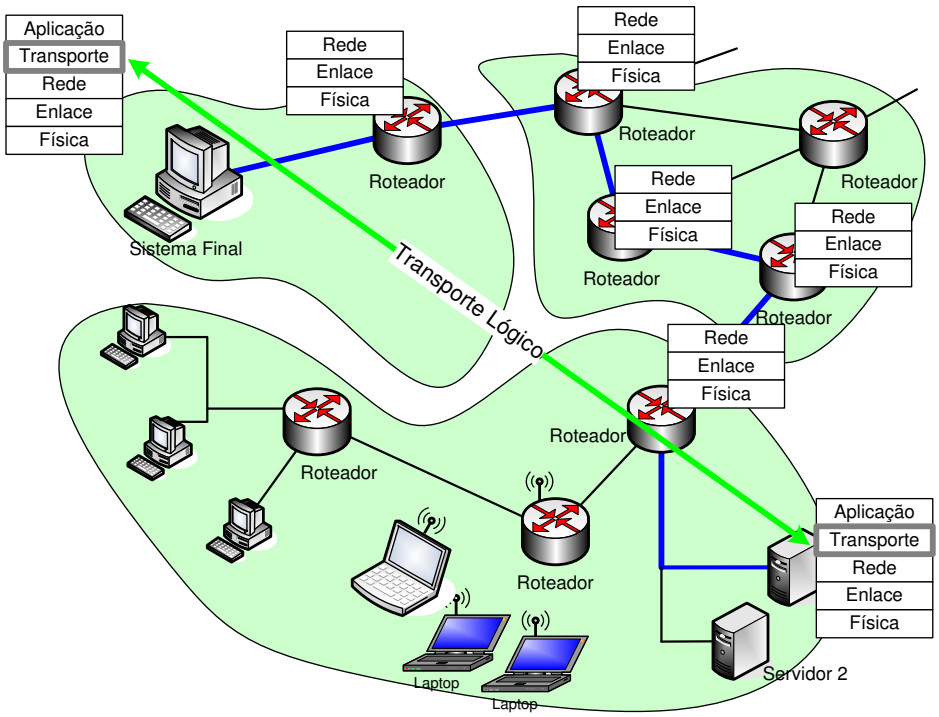


Figura 2.2: Comunicação lógica entre as camadas de transporte dos sistemas finais. Adaptado de [KR06].

segmento para a camada de rede local. Este processo marca a conversão lógica (conceitual) das mensagens em segmentos, os quais serão convertidos em datagramas quando chegarem na camada de rede, como ilustrado na Figura 2.3. Esta conversão consiste na adição de cabeçalhos, os quais possuem informações que são necessárias para prover os serviços de cada camada.

Os comutadores de rede acessam somente informações dos campos adicionados pela camada de rede, isto é, os campos de cabeçalho do segmento IP adicionados pela camada de transporte não são examinados pelos comutadores. Apenas quando o pacote chega no sistema receptor, a camada de rede extrai do datagrama o segmento e repassa-o para a camada de transporte. Finalmente a camada de transporte processa o segmento recebido, disponibilizando os dados para o processo da aplicação via a biblioteca de *socket* disponibilizada pelo sistema operacional.

Multiplexação/Demultiplexação

No sistema final de destino, a camada de transporte recebe segmentos da camada de rede e tem a responsabilidade de entregar os dados desses segmentos ao processo de aplicação apropriado. Por exemplo, se o usuário estiver executando um programa navegador que acessa páginas *web* e baixando um arquivo via FTP existem dois processos de aplicação sendo executados.

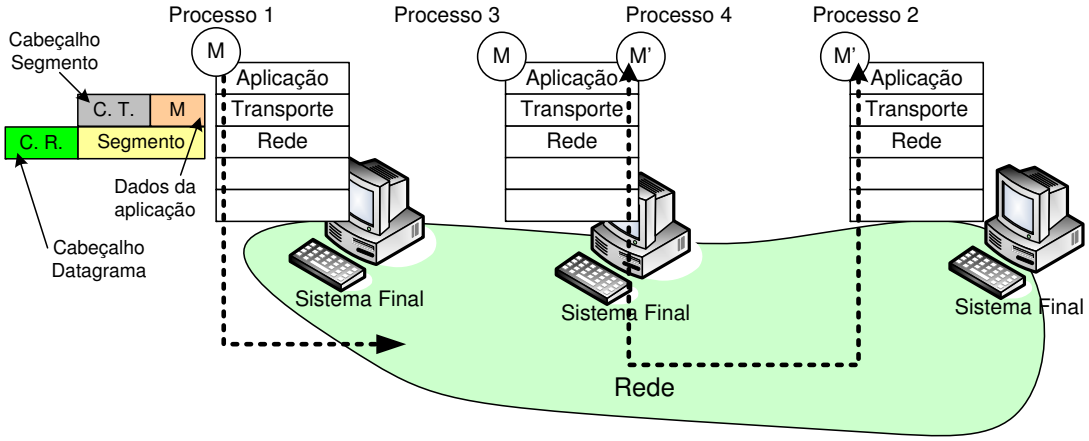


Figura 2.3: Processo de adição de cabeçalhos aos pacotes à medida que estes passam pelas diversas camadas da rede. Adaptado de [KR06]

Quando a camada de transporte desse sistema receber esses dados da camada de rede, precisará direcionar os dados recebidos a um desses dois processos. Considerando a analogia citada anteriormente da **casa e o socket**, então se 5 pessoas residem naquela casa e o carteiro entrega uma carta endereçada a uma das 5, o endereço descrito no envelope seria o endereço IP da aplicação em execução e que detém aquele *socket*; o nome da pessoa seria a porta de entrada daquele *socket*, a qual identifica unicamente o processo correspondente a aplicação – tal como a pessoa destinatária da carta (supondo que na casa não residem duas pessoas com o mesmo nome, o que de fato ocorre para os endereços IP em uma rede de computadores); e a pessoa é a aplicação, que interpretará o conteúdo da carta e realizará uma ação.

Como ilustrado na Figura 2.4, a camada de transporte utiliza o conceito de *socket* para se comunicar com a aplicação remota. O *socket* é identificado por um número único (a porta do *socket*) naquele sistema final.

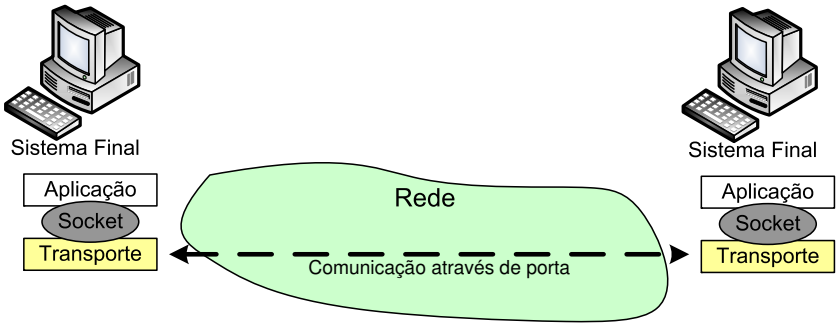


Figura 2.4: omunicação entre a camada de transporte e a camada de aplicação através de um *socket*. Adaptado de [KR06]

O protocolo da camada de transporte de destino ao receber um pacote examina o número da porta de destino que está presente no cabeçalho de cada segmento recebido e identifica para qual processo os dados do segmento deverão ser entregues. Para esse processo dá-se o

nome de **demultiplexação**.

A **multiplexação** ocorre de forma similar a demultiplexação. Este conceito consiste em reunir no sistema de origem os dados proveninentes de diferentes aplicações (portas), adicionar informações de cabeçalho a cada conjunto de dados e repassá-los para a camada de rede, que se responsabiliza de redirecionar cada pacote para o sistema remoto correspondente. Quando o segmento chega no destino, ele passa pelo processo de demultiplexação explicado no parágrafo anterior.

2.3.1 Transporte de dados não orientado à conexão

A principal característica desse serviço da camada de transporte é que não há apresentação mútua entre os sistemas que desejam se comunicar, ou seja, quando uma aplicação envia pacotes a um outro sistema final ela simplesmente o envia, em vez de solicitar um estabelecimento de conexão. O UDP é um exemplo de protocolo que funciona desta forma.

Como não há procedimento de apresentação mútua antes da transmissão de pacotes de dados, os dados podem ser entregues mais rapidamente, o que torna o serviço não orientado à conexão ideal para aplicações multimídia. Uma desvantagem dessa abordagem é que um sistema que transmite não sabe quais pacotes chegaram ao destino e não tem informação alguma, por exemplo, tamanho de *buffer* de recepção etc., sobre o sistema final de destino. Se este for o caso, mais uma vez a camada de aplicação estará infringindo o conceito de protocolos em camadas explicado previamente.

2.3.2 Transporte de dados orientado à conexão

Quando uma aplicação utiliza um serviço orientado à conexão, ela e a aplicação remota enviam informações de controle entre si antes de enviar os dados da aplicação. Esse procedimento que antecede o envio de dados da aplicação é chamado de **estabelecimento de conexão** (*tree-way handshake*). É análogo ao que acontece no diálogo entre duas pessoas para saber informações da hora, citado na Seção 2.1. Na prática, uma conexão consiste em variáveis de estado alocadas nos sistemas finais que informam quais os processos locais estão se comunicando com determinados processos executados nos sistemas remotos.

2.4 Princípios de Controle de Congestionamento

O tema *Controle de Congestionamento* é um assunto vasto e tem sido bastante discutido entre os pesquisadores ao redor do mundo. Atualmente esse tema faz parte do dia-a-dia das pessoas, principalmente para aqueles usuários assíduos da Internet.

A maioria das pessoas conhece os efeitos de um congestionamento: baixar uma música na Internet pode levar 5 minutos, e no outro dia, 10 minutos. Quando leva 10 minutos

considera-se que a rede está congestionada. Tenta-se até deduzir o porquê dessa lentidão: "*deve haver vários usuários baixando músicas e vídeos ou transmitindo algum conteúdo multimídia na rede*".

Porém, muitas vezes as justificativas para um congestionamento da rede pode ser complicada de se descobrir. Um congestionamento pode ser causado por um ataque de negação de serviço [Bap02] (consideravelmente complicado de se descobrir) ou porque a largura de banda disponível da rede já não é mais suficiente para a demanda crescente. Independente de qual seja a justificativa para o congestionamento da rede, nesta seção discute-se o que vem a ser um congestionamento de rede e alguns mecanismos para prover controle de congestionamento adotados pelos atuais protocolos de transporte de rede.

2.4.1 O que é congestionamento de rede?

O serviço de controle de congestionamento de um protocolo da camada de transporte evita que a rede entre em calapso quando um comutador de pacotes fica congestionado [Flo00]. Na prática isso ocorre quando os intervalos de tempo de transmissão e resposta aumentam e os comutadores começam a enfileirar os datagramas nos *buffers* de recepção até que possam distribuí-los. Cada roteador possui uma capacidade limitada de armazenamento, ou seja, *a priori* não há pré-alocação de recursos para conexões TCP, UDP ou DCCP. No pior caso, o número total de datagramas que chega ao roteador congestionado cresce até que o roteador alcance sua capacidade e comece a descartar pacotes.

Em uma conexão TCP por exemplo, as aplicações em execução nos sistemas finais geralmente não tem conhecimento sobre o ponto que ocorreu o congestionamento ou o porquê dele ter acontecido. Para esses sistemas, o congestionamento significa um aumento no tempo de resposta ou a não confirmação de recepção de um segmento, o que presume-se descarte de pacotes em um dos comutadores presente na rota que um determinado pacote foi transmitido.

O TCP reage ao congestionamento implementando a estratégia de retransmissão. Quando um pacote é perdido ele é retransmitido. Porém as retransmissões podem agravar o congestionamento, pois como mais segmentos são transmitidos, ocorre um aumento no tráfego da rede, o que aumenta o tempo de resposta ou resulta na não confirmação de recepção do segmento transmitido. Isto pode levar a um efeito "bola de neve", pois com mais retransmissões, aumenta o tempo de resposta, mais perdas de pacotes, mais retransmissões e assim por diante, até que a rede atinge seu limite e não comporta a quantidade de dados sendo transmitido. Para este fenômeno dá-se o nome de **colapso de congestionamento**.

2.4.2 Duas Abordagens para Controle de Congestionamento

Existem basicamente dois métodos para controlar a taxa de transmissão de dados na rede: o controle baseado em janela e o controle baseado em relatórios enviados pelo sistema recep-

tor [Wel05]. Cada um desses métodos possuem vantagens e desvantagens e funcionam como descrito a seguir:

- *Baseado em Janela*: o transmissor mantém uma janela imaginária e seu tamanho representa uma certa quantidade de pacotes (ou bytes) que o sistema transmissor pode enviar antes que pacotes de confirmação de recepção enviados pelo sistema receptor cheguem no transmissor. Para cada pacote transmitido, o tamanho da janela diminui até ser igual a 0. Por exemplo, suponha que o tamanho da janela de um transmissor é 10, onde a unidade de medida é pacotes e nenhuma confirmação de recepção é recebida pelo transmissor enquanto ele transmite. Neste caso, o transmissor pode transmitir exatamente 10 pacotes, em seguida ele deve parar de transmitir. Se o transmissor receber pacotes de confirmação, ele aumenta o tamanho de sua janela para um determinado valor.

- *Baseado em Relatórios do Receptor*: o valor da taxa de transmissão para um determinado instante é baseado em relatórios transmitidos pelo receptor, que determina qual a taxa máxima que o sistema transmissor deve enviar seus dados (geralmente em bits por segundo). É uma abordagem mais simples e bastante utilizada para transmissão de dados multimídia porque o transmissor não pára de transmitir mesmo que não chegue nenhuma confirmação de recepção.

2.4.3 ECN - *Explicit Congestion Notification*

Existem diversos mecanismos utilizados para que um sistema final tenha conhecimento do estado da rede em termos de congestionamento. Alguns mecanismos estão implícitos através da própria transmissão, como o aumento no tempo de resposta e a não confirmação de recepção de pacotes, o que pode-se inferir que o sistema receptor não está recebendo os dados transmitidos.

Porém, existe um mecanismo explícito utilizado para notificar que a rede está congestionada, também conhecido por ECN (*Explicit Congestion Notification*) [RFB01; RFC03]. O ECN considera o descarte de pacotes pelos roteadores, o que pode acontecer de forma aleatória e depende da capacidade de processamento do roteador. Em vez de fazer descarte de pacotes, o roteador utiliza determinados pacotes para marcá-los com uma sinalização de que a rede está congestionada, ou também conhecido por sinalização CE (*Congestion Experencied*). O objetivo dessa sinalização é notificar o transmissor que a rede está congestionada (ou na iminência de congestionar) e que ele reduza sua taxa de transmissão.

Com o uso de ECN é possível diminuir as perdas de pacotes quando o congestionamento é incipiente, reduzindo as retransmissões e o tráfego na rede. Como o ECN evita perdas desnecessárias de pacotes, as aplicações com pouca troca de dados ou que sejam sensíveis ao

atraso podem se beneficiar com isso [dF04]. O mecanismo de ECN para IP está especificado no RFC 3168 [RFB01] e tanto o TCP quanto o DCCP suportam esse mecanismo.

O ECN é um dos pontos discutidos em [FHK06], o RFC que apresenta diversas justificativas para criação do protocolo DCCP. Se uma aplicação UDP precisar de controle de congestionamento, este mecanismo deverá ser implementado na camada de aplicação, o que aumentará sua complexidade. Neste caso, existem duas opções:

1. ignorar os pacotes marcados com sinal de ECN; ou
2. permitir que a aplicação tenha acesso direto ao campo ECN do cabeçalho IP, sendo possível atribuir ou ler valores deste campo.

Simplesmente ignorar a sinalização ECN não faz muito sentido quando se implementa controle de congestionamento, além de ser útil às aplicações que não garantem entrega de dados. Permitir acesso das aplicações às informações contidas no cabeçalho IP, exigiria alteração das APIs de *sockets* existentes, e mesmo que isto fosse simples de se conseguir, teria que garantir que a camada de rede repassaria a sinalização ECN para a camada de aplicação, o que requer alterações também na camada de rede.

2.5 Protocolo DCCP

Após uma revisão sobre os principais conceitos relacionados à camada de transporte TCP/IP e discussões acerca dos princípios de controle de congestionamento, nesta seção é apresentada uma introdução ao protocolo DCCP.

O DCCP é um protocolo da camada de transporte e realiza transporte não-confiável de datagrama IP. Ele oferece diversas características, sendo as principais o estabelecimento de conexão, não garante entrega e nem ordenação dos dados transmitidos e implementa controle de congestionamento para transmissão não-confiável de fluxo de dados. Assim, o DCCP herda do TCP as características de ser orientado a conexão e fornecer controle de congestionamento. Já do UDP, o protocolo DCCP herda as características de não garantir entrega e nem ordenação dos dados transmitidos.

As principais justificativas para especificar um protocolo orientado à conexão é facilitar a implementação do controle de congestionamento e permitir que as aplicações funcionem mesmo quando estejam conectadas via NAT (Network Address Translation) [RFC94]. O protocolo UDP não apresenta suporte a essas características e por isso a IETF publicou a RFC 3489 [RWHM03], conhecida por STUN (*Simple Traversal of UDP over NAT*). O STUN é uma solução paleativa para suprir a ausência dessa característica do UDP, pois permite que uma aplicação descubra qual seu endereço público de rede, o tipo de NAT utilizado e qual porta NAT está associada a porta do endereço local, para os casos em que a aplicação conecta-se a um sistema fora da rede local via NAT. As informações providas pelo STUN

são usadas para permitir a comunicação UDP entre o cliente e um servidor externo à rede local, e então, poder transmitir dados entre um sistema com endereço local e acessando a rede externa via NAT e um sistema com endereço válido na Internet. Antes do surgimento do STUN, alguns serviços que faziam uso do UDP para transmitir conteúdo multimídia simplesmente não funcionavam quando executados através de NAT. Esse foi o caso do *MSN Messenger*, que até a versão 6.0 o serviço de transmissão de vídeos não funcionava quando conectado através de uma rede com NAT.

Além de procurar resolver problemas já conhecidos, como o mencionado anteriormente, o protocolo DCCP oferece dois mecanismos peculiares e bastante importantes. O primeiro é denominado **Escolha Tardia de Dados** [KL05]. Ele consiste em permitir que a aplicação altere as informações de um pacote imediatamente antes da sua transmissão na rede. O outro é um arcabouço modularizado que permite desenvolver, adicionar e remover **algoritmos de controle de congestionamento**, os quais podem ser selecionados por um determinado tipo de aplicação de acordo com o tipo de conteúdo multimídia sendo transmitido. Mais detalhes sobre esses dois mecanismos são apresentados nas Seções 2.5.2 e 2.5.3, respectivamente.

O protocolo DCCP é especificado pela IETF e sua especificação é divida basicamente em 4 principais RFCs, além de diversos *Internet Drafts*[1] não mencionados neste trabalho. A primeira especificação é a RFC 4336 [FHK06], que apresenta diversos problemas e motivações para a criação de um novo protocolo, em vez de estender o protocolo UDP, por exemplo. A RFC 4340 [KHF06a] trata da especificação do protocolo DCCP propriamente dita, contendo explicações detalhadas do funcionamento interno do DCCP e como funciona o gerenciamento de algoritmos de controle de congestionamento. Já nas RFCs 4341 [FK06] e 4342 [FK06], os autores do protocolo DCCP definem dois mecanismos padrões para controle de congestionamento no DCCP. O primeiro baseado em janelas, similar ao TCP e o segundo baseado em relatórios enviados pelo receptor. Na Seção 2.5.3, são apresentados mais detalhes sobre esses dois algoritmos, onde também são apresentadas justificativas para modularização dos algoritmos de controle de congestionamento.

Devido a essas características que serão detalhadas nas próximas seções, a proposta da IETF para o DCCP é que ele seja utilizado em larga escala na Internet em transmissões de dados multimídia, em muitos casos substituindo o protocolo UDP.

2.5.1 Principais Características do DCCP

A lista a seguir apresenta um resumo das principais características do protocolo DCCP:

- processo de conexão em três-vias similar ao TCP, onde ocorre o estabelecimento e finalização da conexão;

[1] Internet Draft: são documentos de trabalho da IETF antes de se tornarem um padrão da IETF, as chamadas RFCs. Geralmente são válidos por seis meses e podem ser atualizados, substituídos ou descartados por outros documentos em qualquer tempo.

- fluxo de dados não-confiáveis, com confirmação seletiva de recebimento de pacotes;
- opções de negociação com confirmação (por isso o termo confirmação seletiva no item anterior), incluindo negociação do mecanismo de controle de congestionamento a ser utilizado (decisão da aplicação) em tempo de conexão;
- controle de congestionamento com suporte a ECN (*Explicit Congestion Notification*);
- estatísticas da conexão contendo informações sobre quais pacotes de dados chegaram no receptor ou se aqueles pacotes foram marcados com uma sinalização ECN, corrompido, ou deletado por falta de espaço no *buffer* de recepção;
- descoberta de PMTU (*Path Maximum Transmission Unit*) [RFC90], o que ajuda a evitar fragmentação na camada IP.

2.5.2 Estrutura do protocolo DCCP

Nesta seção são discutidos detalhes de funcionamento de algumas das características do protocolo DCCP mencionadas na seção anterior.

Ciclo de vida de uma conexão DCCP

Uma conexão DCCP é estabelecida entre dois sistemas finais, *DCCP A* e *DCCP B*. O *DCCP A* inicia a conexão (cliente) e o outro recebe o pedido de conexão (servidor). Existe também o *DCCP processor*, que se refere a qualquer sistema que processa e repassa um cabeçalho DCCP. Um *DCCP processor* pode ser os próprios sistemas finais ou qualquer outro sistema presente no caminho da conexão, como *firewalls*, roteadores e tradutores de endereços de rede (*NAT*) [RFC94].

O estabelecimento de uma conexão DCCP é ilustrado na Figura 2.5. O processo inicia quando o cliente envia para o servidor um pacote do tipo *DCCP-Request* e o servidor responde com um pacote *DCCP-Response*. Ao receber o *DCCP-Response*, o cliente envia para o servidor um pacote que confirma o recebimento do *DCCP-Response* e a partir desse momento a conexão é efetivamente estabelecida. Após o estabelecimento da conexão, os dois sistemas trocam dados entre si através dos pacotes *DCCP-Data* ou *DCCP-DataAck* enquanto ocorrer a conexão. A conexão é finalizada quando o servidor envia um pacote do tipo *DCCP-Closereq* ou o cliente envia um pacote *DCCP-Close*. Ao receber do cliente a confirmação de recebimento do *DCCP-Closereq* transmitido, o servidor envia para o cliente um *DCCP-Reset* para informar que a conexão está finalizada. Nesse ponto, o cliente permanece no estado de *TIMEWAIT*, que serve para receber eventuais pacotes da conexão que ainda estão em trânsito na rede.

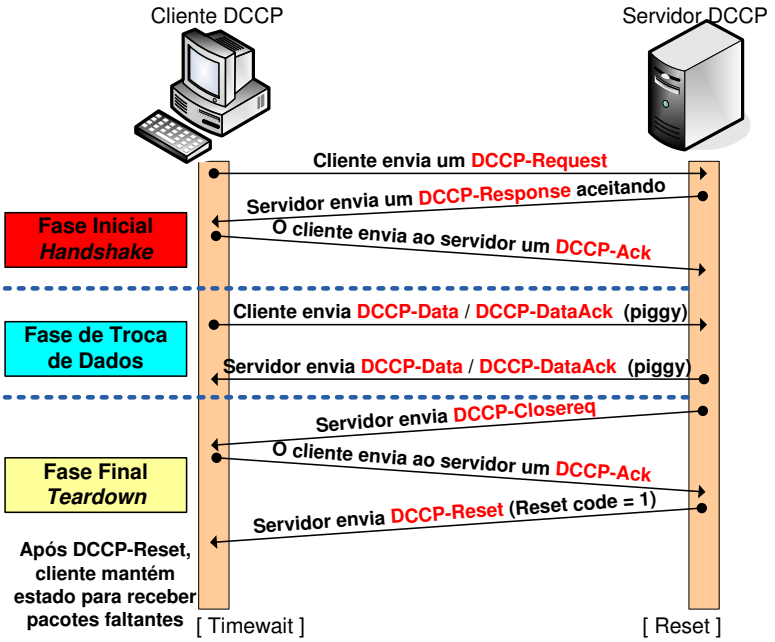


Figura 2.5: Ciclo de vida de uma conexão DCCP.

Conexão bi-direcional

A conexão bi-direcional entre o *DCCP A* e o *DCCP B* indica que pode ocorrer transmissão de informações de A para B ou de B para A, simultaneamente, como ilustrado na Figura 2.6. Na realidade esta conexão consiste em duas conexões unidirecionais chamadas de sub-conexões ou *half-connection*.

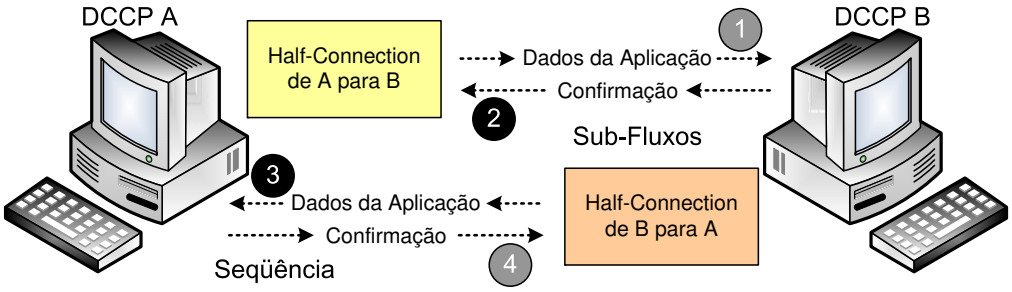


Figura 2.6: Conexão bi-direcional do protocolo DCCP.

Embora estas duas sub-conexões sejam logicamente distintas, elas se sobrepõem. Por exemplo, em uma transmissão de A para B um pacote *DCCP-DataAck* contém dados da aplicação gerados por A e informações que confirmam a recepção de dados transmitidos previamente de B para A.

Os itens enumerados na Figura 2.6 representam os dados de uma conexão DCCP. Estes dados são transmitidos entre os sistemas finais A e B. Em cada conjunto de dados (1, 2, 3, 4) existem tipos de pacotes específicos que trafegam na rede. Na Seção 2.5.2 são discutidos mais detalhes a respeito desses pacotes. Os itens a seguir definem os conceitos relacionados

a estes conjuntos de dados:

Sub-fluxo: consiste de pacotes de dados ou de confirmação transmitidos em uma direção. Cada um dos subconjuntos de pacotes na Figura 2.6 são subfluxos, os quais podem se sobrepor, uma vez que um pacote de confirmação pode "pegar carona" (*piggyback*) em um pacote de dados;

Seqüências: é determinada por todos os pacotes transmitidos em uma direção, podendo ser pacotes de dados ou de confirmação. Nesse caso, os conjuntos $1, 4$ e $2, 3$ são seqüências, onde cada pacote em uma seqüência tem um número de seqüência diferente;

Sub-conexões: consiste de pacotes de dados enviados em uma direção mais as confirmações correspondentes. Os conjuntos $1, 2$ e $3, 4$ são sub-conexões. Na sub-conexão $1, 2$, de A para B, são transmitidos pacotes de dados e de B para A, pacotes de confirmação;

HC-transmissor e HC-receptor: no contexto de uma sub-conexão, o HC-transmissor é o sistema que transmite os dados enquanto que o HC-receptor é o sistema que envia informações de confirmação. Por exemplo, na sub-conexão de A para B, o sistema DCCP A é o HC-transmissor e o DCCP B é o HC-receptor.

Ainda no contexto das sub-conexões, o protocolo DCCP não permite apenas uma delas. Ou seja, o protocolo ao finalizar uma conexão as duas sub-conexões são finalizadas, portanto sendo tratadas como uma única entidade.

Negociação de características da conexão

As características de uma conexão DCCP são atributos da conexão cujos valores são negociados pelos sistemas envolvidos. Com este mecanismo genérico é possível negociar algumas propriedades, tais como o algoritmo de controle de congestionamento que deve ser utilizado em cada uma das sub-conexões. Esta negociação acontece através do uso de opções sinalizadas no cabeçalho de um pacote DCCP.

Uma característica é identificada por um número e um sistema final. Esta característica é denotada por "F/X", onde F representa o número da característica localizada no sistema final X. Cada característica é negociada para uma sub-conexão, sendo possível ocorrer valores diferentes para uma determinada característica em cada direção. Portanto é desta forma que o DCCP possibilita ter um algoritmo de controle de congestionamento sendo executado de A para B e um outro de B para A.

Escolha Tardia de Dados

As aplicações multimídia que utilizam protocolos que implementam controle de congestionamento podem apresentar problemas de desempenho na entrega dos dados. Um dos problemas é que uma informação ao ser transmitida na rede pode chegar a um sistema remoto

depois que esta já não é mais relevante. Isto pode ocorrer devido ao atraso provocado pelos algoritmos de controle de congestionamento implementados na camada de transporte dos sistemas ou por qualquer outro tipo de atraso.

Um mecanismo do protocolo DCCP para tentar solucionar esse problema é chamado de Escolha Tardia de Dados [KL05]. Este mecanismo permite que as aplicações mudem os dados a serem transmitidos imediatamente antes da transmissão, mesmo que a aplicação já tenha liberado os dados para a camada de transporte. Uma aplicação libera os dados para a camada de transporte através das funções de transmissão das APIs de *socket*, como *write* e *send*. Em aplicações de redes multimídia esse serviço pode ser utilizado quando uma aplicação libera os dados para serem transmitidos via DCCP, porém, antes que eles sejam efetivamente transmitidos na rede, a aplicação pode detectar que as informações a serem transmitidas serão descartadas pela aplicação de destino e, então, altera o conteúdo do pacote que será supostamente descartado adicionando informações mais recentes.

Uma aplicação direta desse recurso é na adaptação da qualidade do fluxo multimídia transmitido na rede. À medida que forem ocorrendo mudanças no nível de congestionamento da rede, o sistema altera o conteúdo dos pacotes multimídia com uma qualidade menor do que tinha sido criado previamente (quando a rede não estava congestionada). Como visto na Seção 2.4, quando uma rede está congestionada significa dizer que a quantidade de pacotes em trânsito na rede é maior que a quantidade de pacotes que os roteadores conseguem armazenar em suas filas de roteamento, o que faz eles descartarem pacotes, e quando o protocolo TCP detecta esta perda realiza retransmissões dos pacotes perdidos, o que contribui para o aumento no nível de congestionamento da rede.

Neste sentido, utilizando Escolha Tardia de Dados, diminui-se a quantidade de pacotes na rede que serão apenas descartados no destino. Isto significa que existirão menos pacotes sendo transmitidos na rede, liberando mais espaços nas filas dos comutadores de pacotes. Considerando que toda aplicação DCCP pode fazer uso desse recurso, cada uma delas estará contribuindo para que a rede se recupere do congestionamento, e ainda utiliza melhor os recursos disponíveis da rede. Ou seja, aumentam as chances de transmitir apenas pacotes úteis às aplicações remotas, ao passo que diminuem os que serão descartados no destino.

A escolha tardia de dados pode ser utilizada nas aplicações de voz sobre IP. Por exemplo, dependendo do atraso de um pacote durante a transmissão, o mesmo pode se tornar inútil à aplicação que o receberá. Com base em experimentos relatados pelos autores dos trabalhos referenciados em [dCLF06; GDW05], uma aplicação de voz sobre IP suporta atrasos que variam entre 150 ms e 400 ms. Acima desse valor ou a qualidade do áudio diminuirá ou a aplicação terá que descartar o pacote recebido. Para este cenário, se um pacote não for transmitido em menos de 400 ms, o DCCP pode sinalizar a aplicação que o pacote está atrasado e a aplicação por sua vez, fazendo uso do mecanismo de Escolha Tardia de Dados, interfere na transmissão daquele pacote atrasado e alterado seu conteúdo, adicionando informações do trecho de áudio mais recentes. Se assim não o fizer, este pacote provavelmente

será descartado na aplicação de destino.

Já no contexto de videoconferência, onde em geral o áudio é mais importante que o vídeo, pode-se desenvolver um mecanismo reativo ao congestionamento da rede: quando perceber que a rede está congestionada ou na iminência de congestionamento, este mecanismo pode dar preferência a transmissão de pacotes de áudio a pacotes de vídeo, como apresentado em [SW99].

Cabeçalho DCCP

Na Figura 2.7 é ilustrado o cabeçalho genérico do protocolo DCCP. O nome genérico é justificado porque o cabeçalho assume um formato diferente dependendo do valor de X, conhecido como bit de número de seqüência extendido (*Extended Sequence Numbers bit*). Se o valor de X for 1, o campo Número de Seqüência tem o tamanho de 48 bits e o cabeçalho genérico fica com o tamanho de 16 bytes.

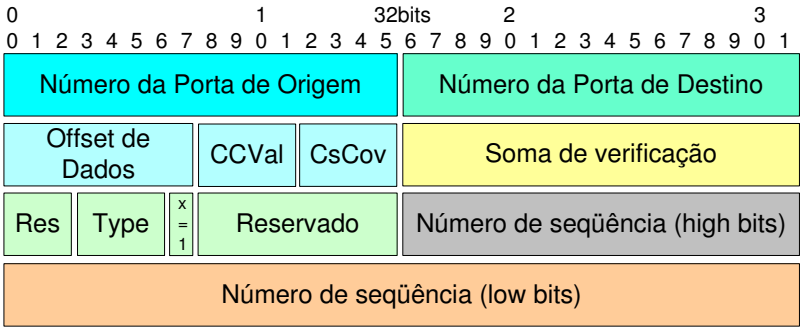


Figura 2.7: Cabeçalho do protocolo DCCP (extendido).

Se o valor de X for 0, apenas os 24 bits do Número de Seqüência são transmitidos e o cabeçalho do DCCP fica com o tamanho de 12 bytes, como ilustrado na Figura 2.8.

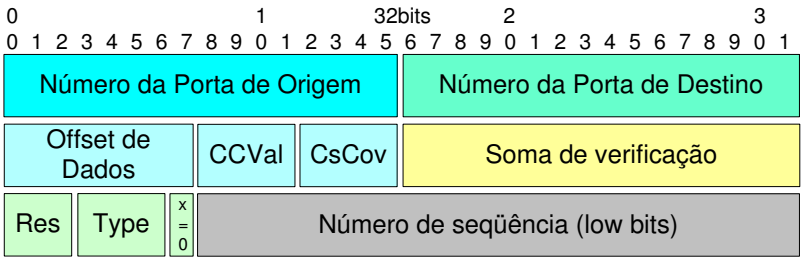


Figura 2.8: Cabeçalho do protocolo DCCP (simplificado).

Os campos do cabeçalho são definidos como segue:

Porta de origem e destino: cada porta possui um tamanho de 16 bits. Estes campos identificam a conexão, como acontece com os protocolos TCP e UDP;

***Data offset*:** ou simplesmente *offset*, determina o tamanho do cabeçalho DCCP, contando do início do cabeçalho até o início de onde estão os dados da aplicação. Este campo tem o tamanho de 8 bits;

CCVal: é utilizado pelo controle de congestionamento do sistema transmissor. O tamanho desse campo é de 4 bits. Em uma *half-connection* de A para B o CCID de A pode enviar 4 bits de informação para B e estes 4 bits são armazenados em CCVal;

***Checksum Coverage* (CsCov):** tamanho de 4 bits. Este campo determina quais partes são protegidos pelo campo de *Checksum*;

***Checksum*:** tamanho de 16 bits. Este campo é utilizado para checagem de erro. Dependendo do valor do campo *Checksum Coverage*, todo, parte ou nenhum dado da aplicação presente no pacote será verificado;

Reservado: tamanho de 3 bits. Campo reservado para utilizações futuras;

Tipo do pacote: tamanho de 4 bits. Este campo determina o tipo de pacote que está sendo transmitido/recebido. Os possíveis valores desse campo são apresentados na Seção 2.5.2;

Número de seqüência extendido (X): se valor de X for 1, o pacote terá o tamanho do campo de número de seqüência com 48 bits, se 0, com 24 bits;

Número de seqüência (bits altos, bits baixos): pode ter o tamanho de 48 ou 24 bits. Identifica unicamente um pacote transmitido na rede por um sistema final. Este número aumenta em 1 a cada pacote transmitido.

Todos os tipos de pacotes, com excessão do *DCCP-Packet* e *DCCP-Data* transportam um sub-cabeçalho para o campo do número de confirmação. Este sub-cabeçalho aparece logo após o cabeçalho genérico, e varia de acordo com o valor de X. Para mais informações a respeito desses campos e dos sub-cabeçalhos, consulte a referência [KHF06a].

Tipo de pacotes

O cabeçalho do protocolo DCCP apresenta um campo denominado *tipo do pacote*. Este campo determina que informação está contida em um determinado pacote DCCP. Na Tabela 2.1 são apresentados os possíveis valores desse campo, nome do pacote e descrição.

2.5.3 Algoritmos de Controle de Congestionamento (CCIDs)

Os CCIDs (*Congestion Control IDentifier*) são módulos independentes do restante do protocolo e responsáveis por realizar o controle de congestionamento durante o ciclo de vida de

Tabela 2.1: Tipos de Pacotes do protocolo DCCP.

#	Tipo	Descrição
0	Request	Pedido de estabelecimento de conexão
1	Response	Resposta ao pedido de estabelecimento de conexão
2	Data	Contém dados da aplicação
3	ACK	Confirmação de recebimento de pacote
4	DataACK	Dados da aplicação e confirmação de recepção
5	CloseReq	Servidor solicita término de conexão sem TIMEWAIT.
6	Close	Servidor/Cliente solicita término da conexão.
7	Reset[2]	Determina, incondicionalmente o final da conexão
8	Sync	Sincronia após perda de pacote ou de uma das sub-conexões
9	SyncACK	Sincronia mais confirmação de recepção
10-15	Reservado	Uso futuro e ignorado pelo receptor

[2] O tipo de pacote **DCCP-Reset** é utilizado com este propósito, porém utiliza-se também para outros motivos: para sinalizar número de porta incorreto; comportamento inapropriado de opções etc.

uma conexão DCCP. Eles descrevem como um sistema que utiliza DCCP limita a taxa de transmissão de pacotes na rede e os valores iniciais de parâmetros da conexão, por exemplo, o tamanho inicial da janela de transmissão (controle de congestionamento baseado em janela) ou como e com qual freqüência o receptor envia informações de congestionamento para o transmissor (controle de congestionamento limitado pelo receptor).

Um determinado CCID pode ser utilizado em qualquer momento da conexão, sendo permitido uma aplicação selecionar um outro algoritmo de controle de congestionamento a qualquer momento da conexão através do mecanismo de negociação de características discutido na Seção 2.5.2. Além de ser negociado no estabelecimento da conexão DCCP, os CCIDs podem ser negociados durante o ciclo de vida da conexão, sendo possível a execução de um CCID em uma direção e um outro CCID na direção contrária. Esta flexibilidade na utilização dos algoritmos de controle de congestionamento em uma conexão DCCP é importante, uma vez a característica de tráfego em uma direção de uma conexão pode ser totalmente diferente se comparada ao tráfego na direção contrária.

A principal justificativa para prover um arcabouço modular para gerenciar os CCIDs é que um determinado algoritmo pode ser mais apropriado para um tipo de aplicação, sendo possível adicionar novos CCIDs ou removê-los de forma independente do núcleo do protocolo. Por exemplo, as aplicações de jogos na Internet podem fazer uso de qualquer largura de banda disponível na rede, pois muitas delas utilizam técnicas de diferença de quadros, onde são enviadas apenas as diferenças entre uma cena do jogo e a outra. Por outro lado, as aplicações de voz sobre IP transmitem rajadas de pequenos pacotes em um curto espaço

de tempo (quando um dos interlocutores fala), sendo as rajadas separadas por períodos de silêncio (quando a pessoa para de falar para dar a vez a outra).

Esta flexibilidade no uso dos algoritmos de controle de congestionamento permite deixar a cargo dos desenvolvedores qual mecanismo de congestionamento é mais adequado à sua aplicação. Desta forma, para um sistema final que envia e recebe dados multimídia é possível ter um algoritmo de controle de congestionamento sendo executado em uma direção (transmissão, por exemplo) e outro algoritmo na direção contrária. Além disso, permite que novos algoritmos de controle de congestionamento sejam desenvolvidos independente da implementação do núcleo do protocolo.

Oficialmente a IETF provê dois CCIDs para o protocolo DCCP: O *TCP-Like Congestion Control* (ou CCID-2) [FK06] e o *TCP Friendly Rate Control* (ou CCID-3) [FKP06]. Além desses dois algoritmos padrões da IETF, existe o *TCP Friendly Rate Control For Small Packets* (ou CCID-4), que encontra-se em estado experimental. Com relação ao restante dos identificadores, entre 0-1 e entre 5-255, eles são reservados para usos posteriores.

TCP-like Congestion Control - CCID-2

O CCID-2 [FK06] é apropriado para as aplicações que utilizam o máximo da largura de banda disponível da rede, mas que se adaptam a mudanças repentinas na largura de banda disponível para transmissão na rede.

O CCID-2 é baseado em controle de congestionamento por janela, similar ao controle de congestionamento do TCP. Quando um pacote é recebido pelo DCCP, ele envia um ACK para o transmissor. O transmissor ao receber o ACK ajusta o tamanho da janela de transmissão e o tempo de expiração para os pacotes ainda não confirmados. Essa estratégia é baseada no conceito de Aumento Aditivo com Redução Multiplicativa (*AIMD - Additive Increase, Multiplicative Decrease*) [Wel05] para controle de congestionamento baseado em janela. Similar ao TCP, o CCID-2 utiliza uma janela de transmissão com tamanho *wsize* números de pacotes, sendo cada pacote de tamanho *psize* bytes. O sistema transmissor ajusta o tamanho da janela à medida que recebe pacotes de confirmação. Desta forma, o valor de *wsize* aumenta em uma unidade nos seguintes casos: (1) a cada confirmação de pacote recebida e (2) quando toda uma janela de pacotes for confirmada na fase de prevenção de congestionamento (*Congestion Avoidance*). Por outro lado, o valor de *wsize* diminui pela metade quando o transmissor detecta perda de pacotes, de maneira equivalente ao TCP. Caso o transmissor não receba um pacote de confirmação de recepção antes do tempo de expiração, o valor de *wsize* é atribuído para 1. O funcionamento é bastante similar ao TCP Reno, discutido em mais detalhes na Seção 2.7.3.

Este algoritmo apresenta basicamente duas diferenças com relação ao algoritmo de controle de congestionamento do protocolo TCP:

- a unidade de medida para o tamanho da janela de congestionamento do protocolo TCP

é em *bytes* (*byte-stream*). Como o protocolo DCCP transmite mensagens datagrama em vez de fluxos de bytes, a unidade considerada para o tamanho da janela de congestionamento é a quantidade de pacotes transmitidos ou recebidos na rede. Assim, a escolha do tamanho de pacote a ser transmitido pode influenciar na qualidade do fluxo transmitido e no desempenho da aplicação. Por exemplo, uma má escolha pode resultar em fragmentação de pacotes na camada de rede;

- o controle de congestionamento padrão do TCP precisa distinguir entre um pacote novo e pacotes retransmitidos. O protocolo DCCP não necessita realizar tal distinção, pois não retransmite pacotes perdidos.

TCP-Friendly Rate Control (TFRC) - CCID-3

O CCID-3 [FKP06] especifica um algoritmo de controle de congestionamento baseado no sistema receptor. A estratégia é que o receptor limite a taxa de envio de pacotes do transmissor através de relatórios contendo informações do estado da conexão, como a taxa de recepção, intervalos de perda e o tempo em que um pacote permanece na fila de recepção (*buffers de recepção*) até que seja confirmado como recebido ao transmissor. Em posse dessas informações fornecidas pelo receptor, o transmissor determina a sua taxa de transmissão para um determinado instante através da equação de vazão 2.1. A equação é denominada *TFRC Throughput Equation*.

$$X = \frac{s}{R\sqrt{\frac{2p}{3}} + 4R(3\sqrt{\frac{3p}{8}}p(1 + 32p^2))} \tag{2.1}$$

onde,

- X é a taxa de transmissão em bytes por segundo;
- s é o tamanho do pacote em bytes;
- R é o RTT (*Round Trip Time*), especificado em segundos;
- $p \in [0, 1]$ é a taxa do evento de perda, que representa a fração de pacotes perdidos;

O CCID-3 é apropriado para aplicações que se adaptam melhor a mudanças mais suaves na largura de banda disponivel para transmissão.

TCP-Friendly Rate Control For Small Packets - CCID-4

Além dos dois CCIDs já padronizados, a IETF está trabalhando na especificação do CCID-4 [KHF07], um novo algoritmo de controle de congestionamento baseado no CCID-3. O CCID-4 está sendo desenvolvido baseando-se em requisitos das aplicações multimídia que

transmitem rajadas de pacotes pequenos (entre 512 bytes e 1024 bytes) em um curto espaço de tempo, como as de voz sobre IP. O CCID-4 permitirá que as aplicações adaptem o fluxo multimídia de acordo com nível de congestionamento da rede (baseado na taxa atual de transmissão) [Cam05]. A idéia é que a qualidade do fluxo sendo transmitido seja adaptado variando o tamanho do pacote através do uso de codificadores que implementam o recurso de VBR (*Variable Bit Rate*). Caso a rede esteja congestionada, diminui-se o tamanho dos pacotes sendo transmitidos, o que diminui a qualidade do conteúdo multimídia sendo transmitido. Porém, se a rede não apresentar congestionamento a qualidade do fluxo multimídia pode ser melhorada, o que altera o tamanho do pacote.

2.6 DCCP e o Protocolo IP

No cabeçalho IP existe um campo chamado *Protocolo*. Este campo tem um papel análogo ao do campo *número de porta* no segmento da camada de transporte. Ou seja, o valor deste campo identifica o protocolo de camada de transporte que será responsável por receber e manipular o datagrama recebido, ao passo que o valor do campo *número de porta* identifica a aplicação que a camada de transporte deve repassar o respectivo segmento.

O IANA (*Internet Assigned Numbers Authority*) é o órgão que define o valor deste campo, que para o protocolo DCCP o valor correspondente é 33 [Aut08]. Este valor[3] deve ser informado na criação de uma conexão DCCP em qualquer API de *socket* disponível atualmente, bastante similar ao estabelecimento de uma conexão TCP. O Código 2.1 ilustra um exemplo utilizando a linguagem C para estabelecer uma conexão DCCP em um servidor remoto utilizando a API de *socket* BSD e considerando as versões mais recentes da *glibc* [4].

Codigo 2.1: Conexão DCCP Através da API de Socket Berkeley

```
#include <arpa/inet.h>
#include <errno.h>
#define IPPROTO_DCCP 33

(...)
int result = 0;
if ((sock_fd = socket (AF_INET, SOCK_DCCP, IPPROTO_DCCP)) > 0) {
    int result = connect(sock_fd, (struct sockaddr *)&serverSockIn,
                         sizeof(serverSockIn));
    return result;
} else {
        printf("Conexão Recusada.");
}
```

[3] Outros valores para o campo *Protocolo* do cabeçalho IP: TCP igual a 6, UDP igual a 17, entre outros.

[4] http://www.gnu.org/software/libc

(. . .)

Um ponto importante nesse contexto é com relação ao funcionamento do protocolo DCCP na Internet. No caso dos experimentos realizados com DCCP, a configuração de bloqueio de datagrama IP teve que ser alterada. A rede utilizada foi a rede da Universidade Federal de Campina Grande, Brasil e a Rede da Universidade Aberdeen, Escócia. Em ambas as redes existiam bloqueios de pacotes IP desconhecidos (diferente de $protocolo = 6$ (TCP) e $protocolo = 17$ (UDP)). Neste caso, foi solicitado ao administrador da rede para que fosse desbloqueado datagrama IP com campo do cabeçalho IP contendo $protocolo = 33$.

Para mais informações de como habilitar e utilizar o protocolo DCCP no núcleo do Linux, consulte o Apêndice A.

2.7 Protocolo TCP

O protocolo TCP (*Transmission Control Protocol*) é um dos protocolos mais importantes da família TCP/IP. A principal característica deste protocolo é que ele fornece um serviço de entrega de pacotes confiável e é orientado à conexão. O protocolo TCP está definido nas RFCs 793 [RFC81], 1122 [Bra89], 1323 [JBB92], 2018 [MMFR96], 2581 [APS99] e 3390 [AFP02].

Como já foi dito, o foco deste trabalho está no protocolo DCCP, e portanto, diferentemente da nossa discussão a respeito do DCCP na Seção 2.5, não serão apresentados detalhes a respeito das funcionalidades do protocolo TCP e seus mecanismos internos, uma vez que existem diversas referências consolidadas disponíveis sobre o tema. Para mais informações a respeito deste protocolo, as referências [KR06] e [Com04] apresentam excelentes leituras como pontos de partida. Ainda assim, existem alguns pontos sobre o protocolo TCP que estão ligados mais diretamente a este trabalho e que é importante mencioná-los. Nas próximas seções estes pontos serão discutidos.

2.7.1 Transporte Confiável, sem Duplicação, com Ordenação e Verificação de Erro

O TCP implementa algumas técnicas para prover transmissão confiável de dados na rede. Dentre elas estão o mecanismo de detecção de erro, a confirmação de recepção e a retransmissão de pacotes.

O protocolo TCP garante a entrega de dados através de retransmissão de pacotes perdidos, sejam detectados através da recepção de pacotes duplicados de confirmação ou por tempo de expiração de confirmação, que é quando o transmissor, após um determinado tempo, não recebe confirmação de recebimento de um pacote transmitido ao sistema remoto. O TCP também evita duplicação e desordenação de pacotes, entregando apenas uma vez à

aplicação um determinado pacote, mantendo a ordenação original mesmo que o transmissor envie mais de uma vez o mesmo pacote.

Para prover todas essas funcionalidades, o protocolo TCP utiliza o campo *número de seqüência* presente no cabeçalho de qualquer pacote TCP. O valor desse campo é negociado no momento do estabelecimento da conexão e nas primeiras versões do TCP sempre iniciava com valor 0, porém devido a alguns problemas de segurança, esse número é gerado em tempo de estabelecimento de conexão e apenas os sistemas comunicantes conhecem esse número. Além disso, o TCP realiza verificação de erro no pacote utilizando o campo de cabeçalho *checksum*, também presente no cabeçalho de um pacote TCP.

Para esse conjunto de características do TCP apresentadas nesta seção, apenas a verificação de erro é feito pelo DCCP.

2.7.2 Controle de Fluxo do TCP

Uma característica do protocolo TCP é o controle de fluxo, também presente no protocolo DCCP. Esta funcionalidade impede que os sistemas comunicantes sobrecarreguem um ao outro com uma quantidade excessiva de informação. O protocolo TCP reserva um *buffer* de recepção para a conexão. Quando o TCP recebe os dados através da rede, o protocolo armazena-os neste *buffer* de recepção. O processo associado a conexão fica responsável por carregar esses dados que estão no *buffer* e disponibilizá-los à aplicação.

Desta forma, como tudo que chega pela rede é armazenado nesse *buffer* de recepção, a aplicação pode não ser capaz de entregar os dados recebidos à aplicação no momento em que eles chegam pela rede (ela pode estar ocupada com alguma outra tarefa). Assim, para esse caso, o sistema transmissor pode saturar o *buffer* de recepção da conexão transmitindo uma quantidade excessiva para o sistema remoto antes que ele entregue efetivamente os dados já recebidos à aplicação de destino. Para evitar este problema, o TCP fornece um serviço de controle de fluxo, responsável por evitar que o transmissor sature o *buffer* de recepção do sistema remoto. No processo de estabelecimento da conexão, os dois sistemas informam o tamanho do *buffer* de recepção e desta forma ambos podem transmitir dados até no máximo o valor do tamanho do *buffer* de recepção. Mas, este não é o único critério que determina quanto o TCP pode transmitir na rede em uma conexão. Na próxima Seção é apresentado um conceito chamado de Janela de Congestionamento, que é utilizado pelo transmissor para determinar a quantidade de dados que ele pode transmitir para o sistema receptor.

Com relação ao protocolo DCCP, ele também implementa controle de fluxo, considerando os mesmos princípios utilizados pelo TCP e apresentados nesta seção.

2.7.3 Controle de Congestionamento do TCP

Como discutido na Seção 2.4, o controle de congestionamento é uma condição do aumento no tempo de resposta e/ou perda de pacotes causados por uma sobrecarga de datagramas em um ou mais pontos de comutação.

Na prática, o protocolo TCP reage aos congestionamentos de interligação das redes controlando a taxa de transmissão de pacotes para uma determinada conexão em um certo instante.

Com base no que foi apresentado na Seção 2.4.1, o colapso de congestionamento ocorre quando há um congestionamento na rede e os protocolos – ou até mesmo as aplicações, dependendo de sua implementação – reagem ao congestionamento retransmitido os segmentos perdidos.

Para evitar este colapso de congestionamento o protocolo TCP reduz a taxa de transmissão quando o congestionamento é detectado, mas continua retransmitido os pacotes perdidos. Para realizar controle de congestionamento através da redução da taxa de transmissão de uma conexão, o TCP utiliza o valor máximo entre o *tamanho da janela do receptor* e o *tamanho da janela de congestionamento* e utiliza as seguintes abordagens:

- Partida lenta (*slow start*);
- Aumento Aditivo com Redução Multiplicativa (*AIMD - Additive Increase, Multiplicative Decrease*); e
- Reação a eventos de perda baseado no tempo limite de espera de confirmação.

As condições são as seguintes: no estado normal, em uma transmissão não-congestionada, a janela de congestionamento é do mesmo tamanho da janela do receptor. Isso significa que reduzir a janela de congestionamento reduz a quantidade de dados que o TCP pode transmitir na conexão. No início de uma conexão TCP o tamanho da janela de congestionamento é igual a 1 segmento. Como a transmissão do protocolo TCP é baseada em fluxos de bytes e não em transmissões de pacotes individuais, como o UDP e o DCCP, o tamanho da janela na verdade é igual ao tamanho de um segmento (tipicamente *536* ou *512* bytes).

À medida que o sistema transmissor recebe confirmação do receptor que os pacotes transmitidos foram recebidos, o valor da janela de congestionamento é incrementado dependendo do estágio em que se encontra a conexão TCP. Ela pode estar no estágio de *partida lenta* (inicio da conexão), no estágio de *prevenção de congestionamento* ou no estágio de *recuperação rápida*.

Quando os pacotes de confirmação são recebidos pelo sistema transmissor, a janela de congestionamento é incrementada de um para dois, e dois segmentos podem ser transmitidos. Quando cada um destes dois segmentos é confirmado, a janela de congestionamento

é incrementada para quatro, de quatro para oito e assim por diante. Isto caracteriza um aumento exponencial. Este processo acontece apenas no estágio de partida lenta, quando em geral o tamanho da janela de congestionamento é igual ao tamanho de um segmento [KR06]. Isto significa que a taxa de transmissão inicial do TCP é de 1 segmento a cada RTT (*Round Trip Time*) [XW01]. Como a largura de banda disponível para a conexão pode ser muito maior que 1 segmento por RTT, seria impraticável esperar por um tempo suficientemente longo até que a taxa de transmissão aumentasse a um valor aceitável. Por isso justifica-se o aumento exponencial na fase de partida lenta.

Ainda na fase de partida lenta, o valor da janela de congestionamento continua aumentando até que ocorra um evento de perda. Neste ponto o valor da janela de congestionamento passa a ser igual a metade do valor atual. Um evento de perda é determinado quando o transmissor recebe três confirmações duplicadas. Neste ponto o valor do tamanho da janela de congestionamento passa a aumentar linearmente. Essa fase de aumento linear é conhecida como prevenção de congestionamento.

Uma situação onde pode ocorrer duplicação de pacote é a seguinte: se um sistema receber um pacote fora de ordem, ele reenvia o último ACK para o último pacote válido recebido. Por exemplo, se o transmissor enviou 5 pacotes e o receptor recebeu os 3 primeiros e o quinto, mas não recebeu o quarto pacote, o receptor enviará um ACK igual a 4. Para o transmissor isso é um pacote duplicado de confirmação, pois quando o terceiro pacote foi recebido, o receptor já havia confirmado com o ACK igual a 4. Se o sexto pacote chegar no receptor e o quarto ainda não estiver chegado, o receptor continuará enviado ACK igual a 4.

O controle de congestionamento do TCP reage de maneira diferente quando um evento de perda é detectado por esgotar o tempo limite de espera por confirmação de um segmento transmitido. Nesse caso, após esgotar o tempo limite de espera, o transmissor entra na fase de partida lenta, isto é, ajusta a janela de congestionamento para o tamanho igual ao tamanho de 1 segmento e então aumenta a janela exponencialmente. O valor da janela de congestionamento continua a aumentar nessa proporção até que este valor alcance a metade do valor que tinha antes de ter esgotado o limite de espera de confirmação de um pacote transmitido e que não foi confirmado. Nesse ponto, o valor da janela de congestionamento volta a aumenta linearmente, da mesma forma como explicado anteriormente.

Um dos primeiros algoritmos de controle de congestionamento do TCP (conhecido também por Tahoe), diminui incondicionamente o tamanho da janela de congestionamento para o tamanho de 1 segmento e entra no estágio de partida lenta após qualquer um dos tipos de evento de perda citados (ou por três duplicações de confirmação de recepção ou por esgotar o tempo de espera por confirmação).

A versão seguinte ao TCP Tahoe é o TCP Reno, que cancela o processo de partida lenta após detectar um evento de perda por receber três pacotes de confirmação duplicados. O motivo de não entrar na fase de partida lenta é baseado na constatação de que mesmo se um pacote tenha sido perdido, a chegada de três confirmações iguais indica que alguns segmentos

foram recebidos no remetente e, portanto, a rede ainda é capaz de entregar alguns pacotes, mesmo perdendo outros pacotes devido ao congestionamento. Essa nova fase adicionada ao TCP Reno é chamada de recuperação rápida.

No entanto, quando ocorre eventos de perda de pacote por esgotamento do tempo de espera por confirmação, o TCP Reno não entra na fase de recuperação rápida, e sim na de fase de partida lenta. Essa decisão está relacionada com a idéia de que a rede não tem capacidade de entregar nenhum pacote e que provavelmente todos estão sendo descartados em algum ponto da rede. Para que o transmissor receba um pacote de confirmação, o pacote transmitido deve primeiro ser recebido pelo receptor. Como os eventos de perda de pacotes ocorrem por esgotamento do tempo de espera por confirmação, isto significa que nenhum pacote transmitido chegou no destino e portanto a rede está descartando todos os pacotes.

Outros detalhes de como funciona o esquema de controle de congestionamento do protocolo TCP são apresentados na referência [Jac98].

2.7.4 Outros Algoritmos de Controle de Congestionamento do TCP

Além do TCP Reno, existem outros algoritmos para controle de congestionamento utilizados no protocolo TCP. Nos experimentos realizados nesta dissertação, foram utilizados outros dois algoritmos de controle de congestionamento, o Cubic e o Veno.

TCP Cubic

O algoritmo de controle de congestionamento TCP Cubic [RX05; HLRX07] é baseado no algoritmo TCP BIC [GS07]. O Cubic simplifica o controle da janela de congestionamento do TCP BIC e melhora a relação de equidade entre os fluxos TCP. Há também melhorias relacionadas à estabilidade do algoritmo quando novos fluxos TCP são transmitidos na rede. O algoritmo definido pelo TCP Cubic controla o tamanho da janela de congestionamento através da Equação 2.2, em termos do tempo decorrido desde do último evento de perda de pacotes. Esta equação foi extraída extraída da referência [RX05].

$$W_{cubic} = C(T - \sqrt[3]{\frac{W_{max}\beta}{C}})^3 + W_{max} \tag{2.2}$$

Onde,

- C é uma constante escalar;
- W_{max} é o tamanho da janela de congestionamento antes da sua última redução;
- T é o tempo decorrido desde a última redução da janela;
- β é o fator de decréscimo multiplicativo depois do evento de perda.

Considerando os conceitos do TCP Reno, o TCP Cubic, em geral, funciona da seguinte forma:

1. o valor da constante escalar C determinar quanto tempo a janela de congestionamento permanece com um valor constante, sem alteraração. Atualmente este valor para a implementação no núcleo do Linux é igual a 0.4;

2. quando ocorre perdas de pacotes detectadas por três ACKs duplicados, o TCP Cubic realiza o decréscimo multiplicativo de acordo com a função 2.2, alterando o valor de β multiplicando-o por um determinado fator;

3. quando acontece perda de pacote por limite do tempo de confirmação de recepção, o algoritmo reinicia todos as variáveis e todo o processo recomeça (partida lenta e mecanismo de prevenção de congestionamento);

O TCP Cubic é o algoritmo de controle de congestionamento padrão do sistema operacional Linux. Na referência [RX05], discute-se em mais detalhes o funcionamento do controle de congestionamento TCP Cubic.

TCP Veno

O TCP Veno [FL03] é um algoritmo para controle de congestionamento baseado no TCP Vegas [BOP94] e no TCP Reno, apresentado anteriormente. A proposta do TCP Veno é obter melhor vazão quando utilizado em redes sem fio considerando a seguinte motivação: diferentemente das redes cabeadas, onde as perdas de pacotes devido a erros de verificação de bit são insignificantes e raramente acontecem [WF94; XD05], nas redes sem fio as perdas de pacotes por este tipo de erro ocorrem com mais freqüência. Para esse tipo de perda o motivo não é o congestionamento da rede. Esses erros são de fato causados por ruídos no canal, problemas no meio físico (causados por obstáculos) ou qualquer outro motivo diferente do congestionamento da rede. Essas perdas de pacotes degradam significativamente o desempenho do algoritmo de controle de congestionamento TCP Reno, por exemplo.

A questão nesse ponto é que o TCP Reno não distingue as perdas de pacotes por erros no conteúdo dos pacotes ou devido ao congestionamento da rede [BPSK97; Bal98]. A estratégia do TCP Veno é monitorar o nível de congestionamento da rede e usar essa informação para decidir se as perdas de pacotes são devido ao congestionamento ou trata-se de uma perda aleatória causada por qualquer motivo diferente do congestionamento na rede. A relação Veno e Reno é a seguinte:

1. o algoritmo de partida lenta continua sendo o mesmo que o Reno, aumento exponencial até que ocorra uma perda de pacote;

2. o Veno altera o algoritmo de aumento aditivo do TCP Reno: além do tamanho da janela de controle de congestionamento (*wsize*), no Reno existe um limiar para a partida lenta *sshresh*. Quando *wsize* é menor que *sshresh* o algoritmo de partida lenta é utilizado para ajustar o valor de *wsize*. Porém quando *wsize* é maior que *sshresh*, a taxa de aumento da janela diminue para evitar congestionamento. No Reno, o valor de *sshresh* é igual a 85.3 *KB*[5]. No Veno esse valor é ajustado dinamicamente, dependendo do mecanismo que determina se as perdas de pacotes estão sendo causadas pelo congestionamento na rede ou por algum outro motivo. Portanto, para os casos em que for determinada perdas de pacotes por erros de verificação de bit, na fase de partida lenta o Veno continua aumentando a taxa de transmissão por um período mais longo que o TCP Reno;

3. o Veno altera o algoritmo de decréscimo multiplicativo do tamanho da janela de congestionamento. No TCP Reno existem duas maneiras de detectar perda de pacotes. A primeira é quando um pacote não é confirmado pelo receptor até esgotar o tempo limite de espera por confirmação. Nesse caso, o algoritmo é reiniciado com o valor de *wsize* e *sshresh* igual a 1. Ou seja, para perda de pacote causada por esgotamento do tempo limite de espera por confirmação, pode-se considerar um congestionamento severo na rede. O TCP Veno não altera essa parte do algoritmo. A outra maneira de detectar perda de pacotes é através do recebimento de três confirmações repetidas, como discutido anteriormente. No Reno, quando a duplicação ocorrer por três vezes, considera-se que o pacote foi perdido, mesmo não ocorrendo o esgotamento do tempo limite de espera por confirmação. Nesse ponto o algoritmo entra na fase de recuperação rápida. Esse algoritmo funciona da seguinte forma:

   (a) Retransmite o pacote perdido, o valor de *wsize* passa a ser igual a *sshresh* e o valor de *sshresh* passa a ser igual a metade do valor antigo de *wsize*;

   (b) Cada vez que um ACK repetido chegar, incrementa o valor de *wsize* em uma unidade;

   (c) Quando um ACK chegar confirmando um pacote, o valor de *wsize* passa a ser igual ao valor de *sshresh*.

   O que o TCP Veno faz é modificar o passo "a" descrito acima. Ele reduz o valor de *sshresh* e consequentemente diminui o limite para *wsize*, porém quando é determinado que a perda de pacote foi causada por erro de verificação de bit, o valor de *sshresh* passa a ser igual a *wsize* * 4/5. Se a perda for causada pelo congestionamento da rede, o TCP Veno funciona como o TCP Reno, o valor de *sshresh* é igual a metade de *wsize*.

[5] Valor referente a implementação do TCP Reno no Linux

Para determinar se a rede está congestionada, o TCP Veno utiliza a estratégia definida pelo TCP Vegas. O processo consiste em continuamente medir o RTT e guardar um histórico dessas medições. Se o valor de RTT aumentar à medida que novas medições de RTT são feitas, a rede está congestionada, caso contrário, os eventos de perda são considerados eventos aleatórios devido à interferências no meio físico. Na referência [BOP94], esse mecanismo é explicado em mais detalhes.

2.8 Protocolo UDP

O UDP é um protocolo de transporte simplificado, em termos de serviços oferecidos, se comparado ao TCP e ao DCCP. É um protocolo não orientado à conexão, portanto não há estabelecimento de conexão antes que os dois processos transmitam dados de um para o outro. Tal como o TCP e o DCCP, o principal objetivo do UDP é fornecer um mecanismo para enviar datagrama a um outro processo executando remotamente. Tal como acontece com o TCP, o protocolo UDP fornece número de portas para estabelecer a distinção entre os diversos programas executados em um sistema final, sendo possível realizar assim a multiplexação e demultiplexação discutidas na Seção 2.3. O protocolo UDP está definido no RFC 768 [RFC80].

O protocolo UDP utiliza o serviço de entrega de datagramas do protocolo IP para transmitir uma mensagem da aplicação para uma aplicação remota na rede. Como já foi mencionado, o protocolo provê um serviço não-confiável de transferência de dados, isto é, quando um processo envia uma mensagem através de um *socket* UDP, o protocolo não oferece nenhuma garantia de que a mensagem chegará ao processo receptor.

Portanto, o protocolo UDP não oferece confiabilidade na entrega de dados, não realiza ordenação, não implementa controle de congestionamento e nem evita duplicação de recebimento de informação. A ausência dessas características faz com que muitos desenvolvedores o utilizem em suas aplicações multimídia. Isto ocorre pelo fato que as aplicações de tempo real podem tolerar perda de informação, embora exijam um taxa mínima de dados entregues no destino.

Além da aplicabilidade do protocolo UDP na área de aplicações multimídia, o UDP é utilizado pelos servidores de nomes da Internet [Moc87], pois este serviço troca pouca informação e deve funcionar de forma rápida. Neste caso estabelecer uma conexão TCP todas as vezes que um sistema precisar resolver um nome para um endereço IP demandaria tempo. Nessa linha, o UDP também tem sido utilizado por usuários Internet mal intencionados, que o utiliza para realizar ataques de negação de serviço distribuído, também chamado de DDoS (*Distributed Denial of Service*). Um ataque bastante conhecido dessa natureza ocorreu em outubro de 2002 [Bap02], onde 9 dos 13 servidores DNS (*Domain Name Server*) raíz da Internet entraram em colapso por uma hora, devido ao excessivo número de requisições rea-

lizadas a estes servidores, requisições estas feitas de forma intencional e com fins de fazer a Internet literalmente parar de funcionar.

Para mais informações acerca do protocolo UDP, consulte as referências [KR06] e [Com04].

2.9 Comparação do DCCP com o TCP e UDP

Na Tabela 2.2 é apresentada uma comparação das principais características discutidas ao longo deste capítulo com relação aos protocolos TCP, UDP e DCCP. Através desta tabela é possível observar que o protocolo DCCP é diferente do protocolo TCP em apenas 4 pontos, destacados em negrito na tabela.

Tabela 2.2: Tabela comparativa das características do TCP, UDP e DCCP.

Característica	UDP	TCP	DCCP
Tamanho do Cabeçalho	8 bytes	20 bytes	**12 ou 16 bytes**
Entidade da camada de transporte	Datagrama	Segmento	**Datagrama**
Numeração de porta	Sim	Sim	Sim
Detecção de Erro	Opcional	Sim	Sim
Garantia de entrega de dados	Não	Sim	**Não**
Número de seqüência e ordenação	Não	Sim	Sim/**Não**
Controle de fluxo	Não	Sim	Sim
Controle de congestionamento	Não	Sim	Sim
Suporte a ECN	Não	Sim	Sim

O primeiro deles o tamanho do cabeçalho de cada pacote, onde como foi explicado na Seção 2.5.2 o tamanho do cabeçalho varia de acordo com o valor do campo X do cabeçalho, podendo assumir tamanho de 12 *bytes* ou 16 *bytes*.

O segundo ponto que muda é mais conceitual, enquanto que o TCP transmite um segmento, o DCCP transmite um datagrama. O terceiro ponto é que o protocolo DCCP não garante entrega dos dados transmitidos, exceto quando existe negociação das características da conexão, como apresentado na Seção 2.5.2. E o quarto ponto é que embora o DCCP utilize número de seqüência, ele não garante entrega ordenada dos pacotes transmitidos, utilizando este campo para realizar implementar os mecanismos de confirmação de pacotes.

Neste capítulo foram discutivos diversos assuntos relacionados à camada de transporte TCP/IP. Foi apresentada uma visão geral desta camada e seus principais serviços. Em seguida foram abordadas as principais características e mecanismos de funcionamento interno

do protocolo DCCP. Por último foi apresentada uma visão geral dos protocolos TCP e UDP e uma tabela comparativa entre esses dois protocolos e o DCCP.

Capítulo 3

Redes Sem Fio e Aplicações Multimídia

> As vezes me pergunto como pode ter acontecido de eu ser o único a desenvolver a Teoria da Relatividade. Acredito que a razão é que um adulto normal nunca pára para pensar sobre problemas de *espaço* e *tempo*. (Albert Einstein)

Após as discussões acerca da camada de transporte do modelo TCP/IP, neste capítulo é apresentada uma visão geral sobre o padrão 802.11 e os atuais desafios que a computação móvel impõe às aplicações multimídia executadas nesse tipo de rede. Em seguida são apresentados os requisitos necessários para transmissão de dados em redes sem fio, iniciando com uma breve introdução ao padrão 802.11 e depois apresentando-se as discussões sobre a utilização de aplicações multimídia neste tipo de rede.

3.1 Redes Sem Fio 802.11

O padrão 802.11 compõe o conjunto de padrões da família 802.x e oferece um conjunto de especificações descritas em diversos documentos disponibilizados pela IEEE (*Institute of Electrical and Electronics Engineers*) [TIoEEE05a]. Este padrão é sub-dividido em quatro especificações, são elas a 802.11a [TIoEEE05b], 802.11b [TIoEEE05c], 802.11g [TIoEEE05e] e a mais recente, a 802.11e [TIoEEE05d].

A família de padrões 802.x refere-se às camadas físicas e de *enlace* do modelo ISO/OSI. O padrão 802.11 foi aceito pela IEEE em 1997 e revisado em 1999. Ele especifica o mecanismo de controle de acesso ao meio (*Medium Access Control* - MAC) para redes sem fio e a camada física (PHY) para prover conectividade entre dispositivos fixos, portáteis e móveis dentro de uma rede local (LAN).

As especificações do padrão 802.11 podem ser descritas da seguinte forma:

- **802.11** é o primeiro padrão para redes sem fio da IEEE. Permite conexões a uma taxa

de até 2 *Mbits/s* na freqüência de 2.4 *GHz*. Atualmente não são mais fabricados produtos que seguem esta especificação e a mesma foi estendida para o padrão 802.11b;

- **802.11a** é também uma extensão do padrão 802.11. Foi criada na mesma época do 802.11*b* e provê transmissão a 54 *Mbits/s* na freqüência de 5 *Ghz*. É melhor, se comparado ao padrão 802.11b, para aplicações multimídia em ambientes com alta densidade populacional;

- **802.11b** também chamado de 802.11 *High Rate* ou *Wi-Fi*, é uma extensão do 802.11 e oferece transmissão a uma taxa de até 11 *Mbits/s* na freqüência de 2.4 *GHz*. Não funciona com equipamentos que seguem o padrão 802.11a, pois cada um dos padrões funciona em uma freqüência diferente;

- **802.11g** oferece transmissão a uma taxa de até 54 *Mbits/s* na freqüência de 2.4 *GHz*. Foi criado para substituir e ser compatível com o padrão 802.11b. Este foi o padrão adotado para a execução dos experimentos realizados neste trabalho;

- **802.11e** é o padrão mais recente da série 802.11 e provê um mecanismo para transmissão de dados baseado na qualidade de serviço [LAS03] da aplicação. É uma adaptação voltado para as aplicações que transmitem dados multimídia em redes sem fio, como as aplicações de VoIP.

3.1.1 Problemas inerentes às WLANs e que afetam as aplicações multimídia

As redes WLAN podem gerar problemas às aplicações multimídia devido à natureza do meio de transmissão – utiliza-se ondas de rádio freqüência para transmitir os dados – e devido aos protocolos utilizados, que podem afetar a qualidade dos serviços oferecidos neste tipo de rede.

Os canais sem fio são imprevisíveis e, dependendo do tráfego na rede, podem ocorrer erros devido aos diferentes caminhos que os pacotes podem seguir para alcançar o destino. Estes erros no canal de transmissão podem corromper ou gerar perda de pacotes.

A largura de banda das redes sem fio é menor se comparada às redes cabeadas. Assim, este tipo de rede está mais suscetível a congestionamento de transmissão, especialmente nos pontos de acesso, devido à centralização do tráfego da rede.

A interferência de Rádio Freqüência (RF) também é um dos problemas. Uma fonte de RF pode bloquear uma rede sem fio durante todo o tempo que o sinal de interferência estiver presente. Se o modo DCF estiver em uso, as estações verificam se o meio está ocupado antes de transmitir, e apenas transmitirão quando o meio esteja livre. Porém, para o caso de uma interferência de RF, o meio não está sendo utilizado, e sim ocupado por informações

inúteis, porém isso já é suficiente para não permitir envio de dados por parte dos dispositivos conectados.

Um outro problema é a mobilidade dos usuários. O alcance dos sinais transmitidos pelos pontos de acesso sem fio 802.11g alcançam entre 70 e 100 metros, porém os usuários podem se locomover para um lugar onde não tem cobertura de sinais desses pontos de acesso. Uma técnica bastante utilizada pelas aplicações de rede é a técnica de *hand-off*.

3.1.2 Hand-off

O *hand-off* é o processo pelo qual um dispositivo móvel consegue manter sua conexão ativa mesmo quando ele sai da cobertura de uma rede sem fio e entra na cobertura de uma nova rede [SZ06; OSL+06].

Esta funcionalidade é bastante requerida em aplicações móveis e pode ser exemplificada em três passos, como ilustrado na Figura 3.1. Nesta figura, são apresentados três domínios administrativos[1] de rede: o laboratório Embedded, o laboratório LSD e o ponto de acesso sem fio (*WLAN Access Point*) chamado AP de Apoio, hipotético, pertencente à rede local da UFCG.

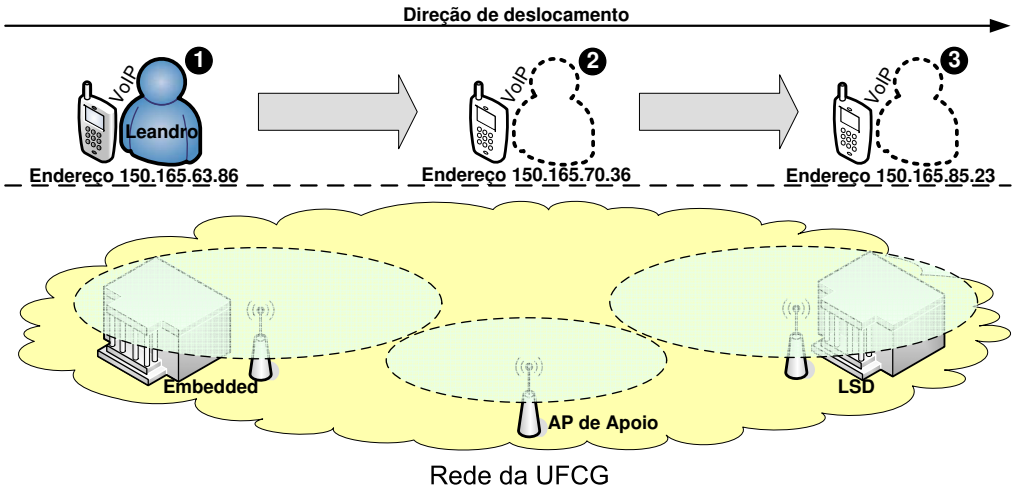


Figura 3.1: *Hand-off* de uma aplicação VoIP entre o Embedded e o LSD.

Suponha que o usuário *Leandro* (passo 1), portando um dispositivo móvel, esteja em uma chamada telefônica de voz sobre IP através da Internet. Neste momento (Passo 1) ele se desloca partindo do laboratório Embedded com destino ao laboratório LSD, ambos localizados na UFCG. Ao longo desse trajeto existe um ponto de acesso utilizado para dar cobertura de conexão no caminho a fim de evitar perda de conexão do usuário com a Internet.

O dispositivo do usuário no Passo 1 está conectado à Internet através do ponto de acesso presente na rede do laboratório Embedded. Assim o endereço IP da conexão do dispositivo

[1]Domínio Administrativo: Local ou rede de computadores sob administração de uma pessoa ou grupo de pessoas de um departamento.

nesse instante é 150.165.63.86, alocado pela infra-estrutura de rede do laboratório Embedded. À medida que *Leandro* se desloca com destino ao LSD, o mecanismo de *hand-off* presente no trajeto permite que sua conexão não seja perdida, mesmo que o dispositivo do usuário mude de ponto de acesso. Isto é possível porque quando a intensidade do sinal do ponto de acesso do Embedded diminui em relação ao dispositivo do usuário, a intensidade do sinal entre o dispositivo do usuário e o ponto de acesso de apoio aumenta. Assim, o dispositivo do usuário desconecta do ponto de acesso atual (ponto de acesso do Embedded) e se conecta ao ponto de acesso de apoio (Passo 2).

Este procedimento faz com que o dispositivo obtenha novas configurações de rede como, por exemplo, o endereço IP 150.165.70.36. O usuário continua conectado à rede e possivelmente com sua chamada telefônica VoIP mantida. Quando *Leandro* se afasta do "AP de Apoio" (Passo 3), a intensidade do sinal da conexão *sem fio* diminui ao passo que aumenta entre o dispositivo dele e o ponto de acesso localizado nas proximidades do laboratório LSD. Neste ponto ocorre um processo similar ao Passo 2, o que resulta na obtenção de um novo endereço para o dispositivo do usuário, o endereço IP 150.165.85.23.

Com a realização de *hand-off*, o algoritmo de controle de congestionamento TCP tende a interpretar as perdas de pacotes como se fossem causados por congestionamento, o que fará reduzir desnecessariamente a janela de congestionamento e portanto diminuir a vazão.

3.1.3 IP Móvel

Existem diversas soluções que suportam o mecanismo de *hand-off*. A solução mais conhecida para redes IP é o padrão IP Móvel [Per02; Per98], que permite roteamento transparente de pacotes IP para dispositivos móveis conectados à Internet. Segundo [KR06], o padrão IP Móvel oferece basicamente três partes principais:

- *Descoberta de agente*: define os protocolos utilizados por um agente nativo ou por um agente externo para anunciar seus serviços a dispositivos móveis e protocolos para que esses dispositivos solicitem os serviços de um agente externo ou nativo;
- *Registro junto ao agente nativo*: define os protocolos usados pelo dispositivo móvel e/ou agente externo para registrar e anular os registros de COAs (Care-Of-Address) junto ao agente local de um dispositivo móvel; e
- *Roteamento indireto de datagramas*: define a maneira pela qual os datagramas são repassados para os dispositivos móveis por um agente nativo, incluindo as regras para repassar os datagramas, regras para manipular as condições de erro e diversas formas de encapsulamento.

3.1.4 WANs Heterogêneas

Na maioria dos casos, em redes locais, a comunicação é estabelecida entre dois sistemas finais. Os dados são roteados através de uma WAN (*Wide Area Network*), sendo em sua maior parte cabeada. Para um sistema final pertencente a uma rede sem fio, os dados trafegam em ambas as redes, nas cabeadas e nas redes sem fio. Estas redes são chamadas WANs heterogêneas.

O gargalo nesse tipo de rede é geralmente no canal sem fio, uma vez que os canais cabeados são considerados mais confiáveis e suportam um tráfego maior de dados. Um dos cenários configurado para a realização foi exatamente esse. A topologia da rede configurada para este cenário será discutida, a seguir, no Capítulo 5.

3.2 Aplicações Multimídia em Redes IP

Como já foi discutido anteriormente, as aplicações multimídia tem sido bastante utilizadas na Internet. A demanda por essas aplicações tem aumentado à medida que elas tornam-se cada mais sofisticadas. Essas aplicações possuem restrições de tempo e podem tolerar eventuais perdas de dados.

Devido as melhorias alcançadas nas tecnologias de transmissão de dados, a qualidade na transmissão dos dados desse tipo de aplicação melhorou substancialmente ao longo dos anos e cada tipo de aplicação possui diferentes requisitos de rede. Os requisitos para algumas dessas aplicações são apresentados na Tabela 3.1.

Tabela 3.1: Requisitos das aplicações de rede. Tabela extraída de [KR06].

Aplicação	Perda	Larg. de Banda	Sensível ao Atraso
Áudio em Tempo Real	Tolerante	*Kbits/s* – 1 *Mbits/s*	Sim: décimos de segs.
Vídeo em Tempo Real	Tolerante	10 *Kbits/s* – 1 *Mbits/s*	Sim: décimos de segs.
Jogos em Rede	Tolerante	Alguns Kbps – 10 *Mbps*	Sim: décimos de segs.
A/V Armazenado	Tolerante	Igual acima	Sim: alguns segs.
Trans. de Arquivo	Sem Perda	Elástica	Não
E-Mail	Sem Perda	Não	Não
Documentos Web	Sem Perda	Elástica/Alguns kbps	Não
Msgs Instantâneas	Sem Perda	Elástica	Sim e Não

As quatro primeiras aplicações listadas na tabela são aplicações que transmitem conteúdo multimídia. Todas elas toleram perdas ocasionais de dados e algumas mais sofisticadas possuem mecanismos de recuperação de dados perdidos durante a transmissão. O objetivo de

tais mecanismos é manter o nível de qualidade do conteúdo multimídia mesmo quando algumas informações são perdidas, evitar a retransmissão e consequentemente evitar o uso desnecessário dos recursos de rede ao passo que contribui para manter o nível de tráfego de dados na rede relativamente baixo.

Para este trabalho existe um interesse particular nas aplicações de voz sobre IP. Portanto nas próximas seções deste capítulo são abordados apenas os tópicos relacionados a este tipo de aplicação.

3.2.1 Desafios das Aplicações VoIP no Contexto das WLANs

Em aplicações de VoIP os dados são transmitidos através de uma rede comutada por pacote [CY88], ao passo que a telefonia tradicional utiliza as redes comutadas por circuito [CY88]. Para a tecnologia de VoIP, os protocolos de sinalização, como o SIP [WAR04] e o H.323 [BA03] são utilizados para estabelecer, controlar e finalizar as chamadas telefônicas.

Requisitos Específicos para Aplicações de VoIP

As aplicações de VoIP possuem requisitos específicos devido as suas características específicas. O tráfego de telefonia é intenso, a transmissão se caracteriza por rajadas de pacotes em um curto espaço de tempo. O fluxo de voz consiste em períodos de conversação intensa separados por silêncio. A qualidade da voz percebida pelo usuário é sensível ao atraso e à variação do atraso. Além disso, os pacotes precisam ser reproduzidos em ordem para que o interlocutor seja entendido pela pessoa que o escuta.

Nas aplicações de voz sobre IP, como qualquer outra aplicação multimídia em tempo real, o tempo de resposta é mais importante que à qualidade da informação. Se a informação chegar com um nível alto de atraso, esta não será utilizada. Para estas aplicações é mais importante que a maioria dos pacotes alcancem o seu destino do que todos os pacotes eventualmente alcancem, em algum momento indefinido. Atualmente, é possível reproduzir o áudio mesmo faltando alguns pacotes. Para um certo nível de perda, alguns *Codecs* de áudio e vídeo são robustos a perda de pacotes, pois implementam alguns algoritmos para recuperação ou preenchimento automático da parte perdida [DYP07].

O atraso depende da duração da transmissão de um pacote. Quanto menor for um pacote, mais rapidamente é codificado e transmitido na rede. Esta é a razão pela qual as aplicações de VoIP utilizam pacotes geralmente pequenos. Uma relação deve ser encontrada a fim de reduzir o tempo de transmissão e codificação de cada pacote e ainda manter a transmissão de carga útil satisfatória. Não é tão simples encontrar essa relação, pois quanto mais pacotes for transmitido, maior será a quantidade de informações de controle transmitida na rede.

Efeitos do Comportamento das Redes Sobre as Aplicações VoIP

Como já foi mencionado na introdução deste documento, as redes IP oferecem o serviço de melhor esforço para transmitir um datagrama IP até o destino. A qualidade do fluxo de uma aplicação VoIP pode sofrer degradações devido a este tipo de serviço. Devido à latência da rede, as aplicações VoIP devem reestruturar o fluxo recebido antes de reproduzí-lo, pois alguns pacotes podem ser perdidos ou chegarem fora de ordem.

Para um nível de latência considerado baixo, em torno de 30 *ms* [Cam05], será maior a qualidade da voz transmitida. Além disso, se a rede prover mecanismos para garantir a qualidade de serviço, a aplicação pode certificar-se de que existe largura de banda fim-a-fim suficiente para realizar a chamada.

As aplicações VoIP são sensíveis aos seguintes comportamentos de rede: atraso (*delay*), variação do atraso (*jitter*) e a perda de pacotes (*packet loss*). Com relação ao atraso e perda de pacotes, esses temas foram discutidos no Capítulo 2, porém é importe discutir sobre o *jitter*.

O *jitter* é a variação do atraso entre pacotes. Se o valor do atraso fosse constante, por exemplo, *todo pacote atrasa 5* ms, bastaria que a aplicação alocasse um *buffer* capaz de comportar os 5 *ms* de atraso e reproduzir os pacotes com 5 *ms* de atraso. Porém esse atraso não é constante, um pacote pode atrasar 50 *ms* e um outro em seguida atrasar 80 *ms*. A variação do atraso nesse caso é de 30 *ms*. Existem pesquisas, como em [LCLZ06; Hua06; BV05; HDS05; JF96; MPSWR96], que tentam alocar o tamanho do *buffer* dinamicamente, de acordo com a variação do atraso. A idéia é compensar a variação do atraso e ainda assim continuar reproduzindo o fluxo sem interrupções.

Telefonia Internet Sem Fio

A telefonia de Internet sem fio tem como principal objetivo prover aplicações móveis utilizando as atuais técnicas presentes na tecnologia VoIP tradicional. A mVoIP, como também é chamada, é baseada nas tecnologias de transmissão sem fio, como a 802.11 e o *blueto-oth* [GDL$^+$04].

A idéia é realizar transmissões de voz em uma rede comutada por pacotes sobre um meio de transmissão sem fio. Um usuário ao utilizar um dispositivo móvel, por exemplo um celular ou um PDA, é capaz de realizar chamadas telefônicas através de uma aplicação sendo executada neste tipo de dispositivo. Essa chamada pode ser feita para um outro cliente mVoIP ou para qualquer outro dispositivo compatível com a tecnologia VoIP.

Os problemas que acontecem nas aplicações tradicionais VoIP também aparecem nas aplicações mVoIP, porém com dois agravante: o meio de transmissão das redes sem fio é mais sensível a interferências, e possui uma menor largura de banda para suportar transmissões de dados multimídia. Estas redes são mais suscetíveis ao congestionamento, principalmente nos pontos de acesso. A tendência é que nas redes sem fio o congestionamento aconteça em

níveis mais elevados e de forma mais rápida, pois a largura de banda disponível é menor se comparada com a largura de banda das redes cabeadas.

A telefonia Internet sobre as redes sem fio gera ou acentua os problemas que acontecem nas redes locais. Um dos principais problemas que acontecem nas redes sem fio é causado pela mobilidade. Os usuários que utilizam a mVoIP tendem a caminhar enquanto conversam ao telefone. Neste caso, fornecer recursos que permitem esta mobilidade é crucial, o que não é necessário nas redes cabeadas. Para atender a este requisito, a cobertura do sinal de uma rede sem fio precisa ser estendida para lugares onde antes não fazia muito sentido, como corredores, escadas etc.

O quesito de segurança é um outro fator importante que precisa ser mencionado. Alguns ataques de rede, como o de escuta (*man-in-the-middle*), são implementados de forma mais fácil sobre as redes sem fio, uma vez que não há como prover segurança física no meio de transmissão das redes sem fio. Neste caso, as aplicações mVoIP devem ser capazes de manter a transmissão de dados confidencial, utilizando técnicas de criptografia, como as que já estão sendo empregadas com uma extensão do protocolo RTP, o protocolo SRTP [BMN$^+$04]. A criptografia em tempo real adiciona mais processamento às aplicações e, como conseqüência, pode ocorrer aumento no tempo de resposta, principalmente se a aplicação VoIP é executada em um dispositivo que tem recursos de memória e de processamento limitados, como os celulares. No entanto, o desafio nesse ponto é manter a qualidade do serviço e ao mesmo tempo transmitir tais informações de forma segura.

Os requisitos de hardware para as aplicações multimídia também é outro fator que pode causar problemas. Os dispositivos portáteis possuem certas limitações quanto ao poder de processamento e de memória. O desafio neste caso está em implementar aplicações capazes de consumir o mínimo possível do processamento do dispositivo, além de tentar diminuir ao máximo o atraso na geração de dados que devem ser transmitidos na rede.

Nos capítulos anteriores foram discutidos os principais assuntos relacionados a este trabalho. Procurou-se oferecer uma visão geral de três grandes temas: **Camada de Transporte TCP/IP**, **Redes Sem Fio** e algumas peculiaridades quando elas são utilizadas para executar as **Aplicações Multimídia**, em particular nas redes padrão 802.11. É impossível discutir todos os tópicos relacionados a estes temas, sendo todos eles objetos de estudos em um vasto número de documentos. Para mais discussões a respeito desses e de outros assuntos relacionados, sugere-se as seguintes referências: [KR06], [Com04] e [Fil98], [Wel05] e [Tan03]. Estas referências foram consultadas e utilizadas como base para compor os capítulos iniciais deste trabalho.

Capítulo 4

Construção da Rede para Execução dos Experimentos

"O que queremos mesmo é gerar um grande tráfego de rede que
incremente ainda mais nossas vendas de roteadores e switches."
(Luis Carlos Rego, Gerente Cisco Brasil)

Para executar os experimentos e coletar os dados para posterior análise a fim de compor os resultados deste trabalho, foi necessário configurar uma série de aplicações e equipamentos de rede, além de desenvolver aplicações para dar suporte aos referidos experimentos e coleta de dados. Por exemplo, para transmitir conteúdos multimídia utilizando o protocolo DCCP – onde focamos os nossos esforços – foi necessário implementar uma séria de funcionalidades relacionadas tanto com o núcleo do Linux quanto com o nível de espaço de usuário (uma aplicação multimídia para transmitir áudio na rede). Neste capítulo são discutidos os pontos centrais neste contexto.

A execução dos experimentos foi constituida em três fases, as quais são descritas a seguir:

1. experimentos utilizando dispositivos com recursos limitados. Foram utilizados 4 dispositivos Nokia *Internet Tablet* N800, mais 2 computadores `desktop`;

2. experimentos utilizando computadores com maior capacidade de processamento e memória, se comparados aos dispositivos utilizados na primeira fase. Foram utilizados 4 *notebooks*, 2 modelos Toshiba Portége e 2 modelos Sony Vaio; e

3. experimentos utilizando 4 computadores conectados à Internet. Foram utilizados 2 *notebooks*, sendo 1 Sony Vaio e 1 HP ZD8000 e 2 computadores *desktop*.

Os detalhes de cada um destes dispositivos são apresentados na Seção 4.2. Para as duas primeiras fases, os fluxos foram transmitidos através de uma rede sem fio composta de três pontos de acesso 802.11g e em uma LAN heterogênea (ver Seção 3.1.4).

4.1 Fases dos Experimentos

Primeira Fase

O objetivo na primeira fase foi analisar o desempenho dos protocolos TCP, UDP e DCCP quando executados em dispositivos com pouco poder de processamento e capacidade de memória. Uma peculiaridade desta fase foi a análise do comportamento dos protocolos quando a aplicação realiza *hand-off* entre os pontos de acesso sem fio.

Os resultados desta primeira fase estão sendo aplicados na implementação do projeto E-Phone (Embedded-Phone), que será apresentado no Capítulo 8. A topologia de rede definida para esta fase é ilustrada na Figura 4.1. Neste caso, foram utilizados quatro N800 para transmitir três fluxos UDP ou DCCP (um em cada N800) e o quarto N800 para transmitir um fluxo TCP.

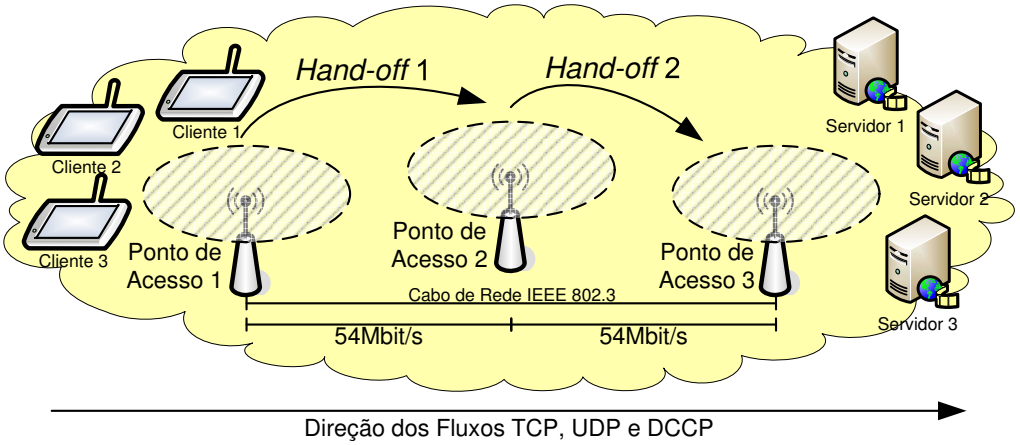


Figura 4.1: Topologia de rede da Fase 1. Dois N800 transmitindo fluxos UDP ou DCCP e um terceiro transmitindo um fluxo TCP sobre uma rede sem fio 802.11g.

Segunda Fase

Na segunda fase o objetivo foi investigar se a capacidade de processamento de um dispositivo poderia acentuar o congestionamento da rede, uma vez que o volume de dados transmitidos é consideravelmente maior se comparado a fase 1. Como os dispositivos possuem maior capacidade de processamento, a idéia nesta fase foi gerar um congestionamento na rede com os fluxos UDP e com um ponto de saturação, reproduzindo um cenário comum atualmente. Na topologia de rede definida para esta fase, dois *notebooks* foram conectados ao barramento 802.3 (*ethernet* 100BASE-TX) do ponto de acesso e os outros dois conectados através do canal sem fio 802.11g, do outro ponto de acesso, como ilustrado na Figura 4.2.

Note que para os dois computadores clientes, a capacidade de transmissão para o barramento *ethernet* é de no máximo 100 *Mbits/s*, enquanto que a capacidade de transmissão para o canal sem fio é de apenas 54 *Mbits/s*. Portanto, quando a direção da transmissão for

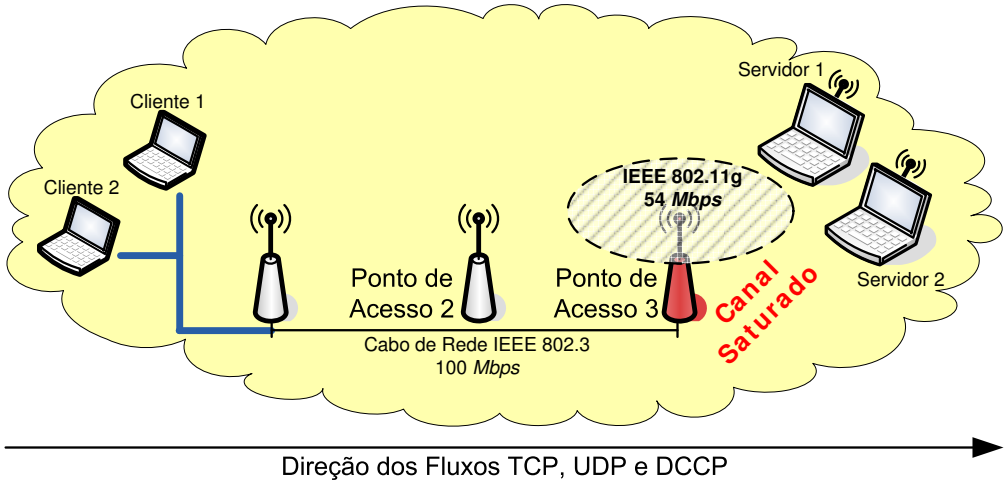


Figura 4.2: Topologia da rede da Fase 2. Um notebooks transmitindo três fluxos UDP/DCCP e o outro, um fluxo TCP.

no sentido da rede cabeada para a rede sem fio, ocorrerá um congestionamento no ponto de acesso, uma vez que este recebe a uma taxa de 100 *Mbits/s* e repassa apenas a uma taxa de 54 *Mbps*. Esta foi a peculiaridade desta fase, construir um cenário onde sempre ocorrerá perdas de pacotes por descarte no roteador, uma vez que o canal fica saturado devido ao limite da fila de roteamento no roteador, fazendo com que ele passe a descartar pacotes.

Este cenário foi discutido na Seção 2.4. É um exemplo típico de uma LAN heterogênea, como discutimos na Seção 3.1.4. A topologia da rede utilizada nesta fase é ilustrada na Figura 4.2.

Terceira Fase

Por fim, na última fase dos experimentos foram transmitidos fluxos TCP, UDP e DCCP na Internet, entre a Universidade Federal de Campina Grande (Brasil) e a Universidade Estadual de São Francisco (Estados Unidos). O principalmente objetivo nesta fase foi analisar o desempenho do protocolo DCCP na Internet, uma vez que a proposta é que o DCCP venha substituir o protocolo UDP em transmissões multimídia na Internet. Uma peculiaridade desta fase é que o conteúdo multimídia transmitido foi armazenado no destino. A estratégia é comparar a qualidade do conteúdo multimídia transmitido com o recebido por cada um dos protocolos analisados. A topologia da rede utilizada nesta fase é ilustrada nas Figuras 4.3 e 4.4.

Figura 4.3: Parte 1 da topologia da rede da Fase 3. Dois *notebooks* no Brasil.

4.2 Configuração dos Dispositivos

A configuração dos dispositivos utilizados em cada fase é apresentada a Tabela 4.1. Todos eles executaram o sistema operacional Linux versão 2.6.25, exceto os dispositivos Nokia N800, cuja versão do Linux é 2.6.21.

Tabela 4.1: Configuração dos dispositivos utilizados nos experimentos.

Modelo	Processador	Memória	Placa de Rede
Nokia N800	ARM 330 *MHz*	DDR 128 *MB*	Texas Instruments
Sony Vaio VGN-FE690G	Pentium 4 2.2 *GHz*	DDR 1 *GB*	Intel PRO 3945ABG
Toshiba Portége M405	Duo Core 2.0 *GHz*	DDR 2 *GB*	Intel PRO 3945ABG
HP ZD8220	Pentium 4 3.2 *GHz* HT	DDR 512 *MB*	Intel PRO 3945ABG
Desktops	Dual Core 2 2.2 *GHz*	DDR 2 *GB*	Intel 82573*L*

Com relação aos pontos de acesso utilizados nos experimentos, os três têm as seguintes configurações:

Modelo: Asus-500g Premium

Processador: 266 *MHz*

Memória: 32 *MB*

Figura 4.4: Parte 2 da topologia da rede da Fase 3. Dois *desktops* nos Estados Unidos.

Portas LAN: 802.3 10/100BASE-T

Modulação: OFDM, CCK

Antena: 2, uma interna (Inverted-F PCB) e a outra externa, Dipolo Reverse-SMA

Alcance: Ambientes fechados: 25 *m*, a 54 *Mbps*

4.3 Ferramentas Desenvolvidas para Execução dos Experimentos

Para transmitir os fluxos de dados na rede durante os experimentos, foram utilizadas duas aplicações: a gstreamer[1] e a iperf[2].

O GStreamer é um arcabouço para desenvolvimento de aplicações multimídia e utilizado pelo projeto Gnome[3] para prover todas as funcionalidades de áudio e vídeo. O projeto é escrito em linguagem C e é baseado em GObject[4]. Do ponto de vista de desenvolvimento, o GStreamer é um arcabouço multiplataforma que suporta o desenvolvimento de componentes (*GStreamer Plugins*). Ele utiliza uma arquitetura de componentes que permite a integração com diversas bibliotecas de mídia disponíveis, como *MP3*, *FFMPeg* e outras. Além disso, o

[1] http://gstreamer.freedesktop.org/
[2] http://dast.nlanr.net/Projects/Iperf/
[3] http://www.gnome.org/
[4] http://library.gnome.org/devel/gobject/stable

GStreamer está disponível para várias linguagens de programação, como Python, C++, Perl, GNU Guile e Ruby.

Neste contexto foi desenvolvido um componente **DCCP para GStreamer**, o qual foi utilizado durante as três fases dos experimentos realizados neste trabalho. A combinação deste componente com outros já disponíveis no GStreamer possibilitou transmitir conteúdos multimídia através da rede. Por exemplo, para transmitir um arquivo *MP3* e reproduzí-lo no servidor, basta executar os dois comandos apresentados no trecho de código 4.1.

Codigo 4.1: Comando para Executar o componente de DCCP para GStreamer

```
Servidor:
  # gst-launch dccpserversrc port=9011 ccid=2 ! mad ! alsasink
Cliente:
  # gst-launch filesrc location=/audio/sample.mp3 ! mp3parse \
      ! dccpclientsink host=192.168.1.1 port=9011
```

Neste caso, o computador cliente transmitirá o arquivo mp3 para o servidor com endereço IP 192.168.1.1 e *dccpserversrc* e *dccpclientsink* são os elementos providos pelo componente DCCP para o GStreamer. Existem outros dois elementos disponíveis, o *dccpserversink* e *dccpclientsrc*, o que permite comunicação em duas vias, para o caso de uma aplicação de voz sobre IP, por exemplo.

Esta implementação tornou-se uma contribuição, sendo atualmente o componente padrão para transmissão de dados utilizando o protocolo DCCP em GStreamer. Este componente pode ser utilizado em qualquer aplicação baseada em GStreamer, uma delas é a aplicação E-Phone. O GStreamer fornece uma API de desenvolvimento que permite acessar as funcionalidades de um componente. Para mais informações sobre o GStreamer e como desenvolver aplicações utilizando o componente de DCCP, consulte o Apêndice B e a referência [SAP08], a ser publicado em Julho de 2008 na revista *The Linux Magazine*[5], EUA.

Uma outra ferramenta utilizada é chamada de IPerf. O IPerf é uma aplicação para análise de desempenho de rede, sendo possível obter relatórios de vazão, perda de pacotes e latência durante uma transmissão de dados. É um projeto mantido pelo Laboratório Nacional de Pesquisas Aplicadas em Redes de Computadores (*The National Laboratory for Applied Network Research*) da Universidade de Illinois, EUA e disponibilizado sob licença GPL.

A versão original do IPerf não suportava o protocolo DCCP, apenas os protocolos TCP e UDP. Devido a esta limitação e para ser possível utilizá-lo nos experimentos, foi disponibilizado o suporte para o protocolo DCCP para o IPerf. Para mais informações sobre o IPerf e o como utilizá-lo com suporte a DCCP, consulte o Apêndice B. Como para as duas primeiras fases de experimentos não existia a preocupação de enviar um tipo de dado específico, utilizou-se nestas duas fases o recurso do IPerf que gera e transmite dados aleatórios na rede com tamanho fixo do pacote, nestes casos pacotes de 512 *bytes* e 1424 *bytes*.

[5] http://www.linux-magazine.com/

Já quanto a coleta de dados e filtragem de informação, foi desenvolvida uma aplicação que otimizasse esse processo. Como será visto no Capítulo 5, foram executados quase 300 experimentos e, portanto, seria inviável realizar um processamento manual do grande volume de relatórios gerados pelo IPerf e pelo componente de DCCP para GStreamer.

Para processar todas as informações e obter os valores de cada métrica de interesse, foi necessário desenvolver uma aplicação capaz de filtrar as informações geradas, possibilitando uma análise posterior dos dados. A aplicação calcula os valores das métricas conforme metodologia apresentada no Capítulo 5 e foi utilizada também para gerar todos os gráficos ilustrados no Capítulo 6.

Com relação a Fase 3, onde foram transmitidos fluxos de áudio na rede, a avaliação da qualidade dos fluxos transmitidos foi objetiva e se baseou na comparação do conteúdo do áudio recebido pelo computador de destino. Na Seção 5.5 é apresentada a metodologia usada para realizar as comparações de áudio. Nesse contexto, foi desenvolvida uma aplicação utilizando o arcabouço GStreamer que é capaz de gerar informações estatísticas baseadas no conteúdo do áudio.

4.4 Núcleo do Linux

A implementação do protocolo DCCP utilizada para realizar os experimentos, foi a da versão do Linux 2.6.25 do núcleo do Linux [Mel05; Cor05]. Como foi especificado na Seção 4.2, o sistema operacional utilizado nos dispositivos Nokia N800 foi o Linux 2.6.21. Este núcleo não suporta as últimas funcionalidades implementadas no protocolo DCCP e, portanto, foi necessário adaptar (*backporting*) este núcleo para suportar as novas funcionalidades disponíveis apenas no núcleo versão 2.6.25.

O resultado desta adaptação também está sendo utilizado pelo projeto E-Phone, através do projeto **DCCP para Maemo** – apresentado mais adiante, no Capítulo 8 – que tem como principal objetivo manter atualizado o núcleo do Linux para a plataforma maemo[6], uma vez que não foi possível utilizar as versões mais recentes do núcleo do Linux diretamente no dispositivo Nokia N800.

[6]http://www.maemo.org/

Capítulo 5

Métodos e Experimentos

"Sabemos que uma conclusão definitiva não pode ser feita baseando-se nas características de todos os sistemas, mas é possível obter conclusões probabilísticas baseando-se num intervalo onde estarão as características da maioria dos sistemas." (Raj Jain, em *A Arte da Análise de Desempenho em Sistemas de Computação*)

Neste capítulo são descritos os parâmetros utilizados em cada cenário dos experimentos, cujos valores foram variados à medida que os experimentos foram evoluindo. Além disso, são apresentadas as metodologias adotadas para obtenção dos valores finais para cada uma das métricas de interesse durante a execução dos experimentos. Por último, apresenta-se um método estatístico baseado na teoria da probabilidade [Jan91], o qual possibilitou calcular a quantidade de repetições necessárias para um experimento e assim obter um nível de confiança aceitável. Com este mecanismo, foi possível realizar comparações quanto ao desempenho de cada um dos protocolos analisados. Tais discussões comparativas são apresentadas no Capítulo 6.

Para entender melhor os conceitos apresentados a seguir, antes é necessário entender o que são os **cenário de experimentos**. Um cenário de experimento engloba os parâmetros configurados, as configurações dos dispositivos e o conjunto dos experimentos executados considerando os valores de cada parâmetro em um determinado momento.

Portanto, um exemplo para um cenário de experimentos pode ser: as definições da fase 1 (topologia de rede, dispositivo utilizado, N800, Linux, memória RAM 128 *MB* etc.), os confrontos dos protocolos (TCP vs UDP, TCP $\times$ DCCP ou DCCP $\times$ UDP), a variação do tamanho dos pacotes transmitidos (512 *bytes* ou 1424 *bytes*), execução ou não de *hand-off* e os algoritmos de controle de congestionamento utilizados em cada protocolo (TCP Cubic, DCCP CCID-2 etc.).

5.1 Parâmetros para os Experimentos

Alguns parâmetros foram definidos para as execuções dos experimentos, cujos valores foram variados em cada cenário de experimentos. Para alguns cenários alguns dos parâmetros definidos não foram variados. Por exemplo, no cenário dos experimentos das Fases 2 e 3 não houve execução de *hand-off* e no cenário dos experimentos da Fase 3 não houve alteração para o parâmetro *tamanho do pacote*, fixado em 1424 *bytes*. Nesta seção são discutidos os valores usados para cada um dos parâmetro e como esses valores foram variados.

Confrontos dos Protocolos

Para transmitir os dados em qualquer cenário de experimentos, os protocolos foram confrontados dois-a-dois, ou seja, TCP × UDP, TCP × DCCP e UDP × DCCP. A quantidade dos fluxos transmitidos durante os experimentos são apresentadados na Tabela 5.1.

#	Confrontos	Fluxo TCP	Fluxos UDP	Fluxos DCCP
1	**TCP × UDP**	1	2	0
2	**TCP × DCCP**	1	0	2
3	**UDP × DCCP**	0	2	1

Tabela 5.1: Quantidade de fluxos utilizados por confronto de protocolos.

O objetivo na definição destes confrontos foi analisar a eqüidade em termos de utilização da largura de banda disponível de cada protocolo quando colocados em confronto entre si.

Tamanho do Pacote

Para cada cenário de experimento o tamanho dos pacotes foi variado. Nas Fases 1 e 2, para cada confronto de protocolo, os experimentos foram realizados com diferente tamanhos de pacotes em cada um dos experimentos. Os valores utilizados foram 512 *bytes* e 1424 *bytes*.

O objetivo para este caso é analisar se a variação do tamanho do pacote transmitido gera algum impacto no desempenho dos protocolos analisados. Como discutido nas Seções 2.5.3 e 3.2.1, algumas aplicações realizam adaptação da qualidade do fluxo multimídia em resposta ao congestionamento da rede. Para realizar este tipo de adaptação, os algoritmos executados pelos codificadores de áudio e vídeo do tipo VBR (veja Seção 2.5.3) variam a taxa de bit para cada pacote gerado de acordo com o conteúdo multimídia sendo transmitido, o que altera dinamicamente o tamanho do pacote ao longo da conexão.

Algoritmos de Controle de Congestionamento

Na Tabela 5.2, apresenta-se os algoritmos de controle de congestionamento utilizados por cada protocolo de transporte durante a execução dos experimentos.

Protocolos	Controles de Congestionamento
TCP	Cubic, Reno, Vegas
DCCP	CCID-2, CCID-3
UDP	–

Tabela 5.2: Algoritmos de controle de congestionamento utilizados nos experimentos.

O objetivo em variar esse parâmetro é de analisar o desempenho de cada algoritmo de controle de congestionamento durante a transmissão dos fluxos de dados. O segundo objetivo é compará-los em termos de eqüidade entre um protocolo e outro quanto ao compartilhamento da largura de banda disponível na rede. Além disso, o terceiro objetivo é analisar o comportamento desses protocolos diante de um congestionamento das topologias de rede definidas, uma vez que o protocolo UDP congestiona a rede.

Execução de *Hand-off*

Nos experimentos realizados na Fase 1, dois dos quatro dispositivos executaram *hand-off*, de acordo como descrito na Seção 5.3. É sabido que durante a execução de um *hand-off* ocorrem perdas de pacotes (Seção 3.1.2) e também que diversos algoritmos de controle de congestionamento, dentre eles o TCP Reno e o DCCP CCID-2, utilizam os eventos de perda de pacotes para inferirem que a rede está congestionada, veja Seções 2.4, 2.5.3, 2.7.3 e 2.7.4.

Portanto, o objetivo em realizar *hand-off* está vinculado ao interesse em avaliar o desempenho dos protocolos analisados diante da execução de *hand-offs*, o que levará os algoritmos de controle de congestionamento a inferirem erroneamente que a rede está congestionada devido aos eventos de perda de pacotes durante o *hand-off*.

5.2 Experimentos

Para todos os cenários de experimentos, primeiramente foi transmitido apenas um fluxo do protocolo TCP, ao passo que sempre[1] foram transmitidos ou 2 fluxos UDP ou 2 DCCP, dependendo do confronto dos protocolos considerado. Executando os experimentos desta forma, é possível avaliar o quanto os protocolos TCP e DCCP são estáveis quando novos fluxos de dados são transmitidos na rede.

[1]Exceto para os confrontos DCCP $\times$ UDP, onde foi transmitido apenas 1 fluxo DCCP e 2 fluxos UDP.

Quanto ao tempo de duração de cada experimento, os experimentos sem *hand-off* tiveram uma duração de 100 *s*. Entretanto, o tempo de duração dos experimentos com *hand-off* foi de 300 *s*. Para este caso, os *hand-offs* foram executados em dois momentos pelo sistema transmissor: o primeiro em 100 *s* e o segundo em 200 *s*.

Um outro ponto considerado foi com relação ao momento de iniciar cada fluxo. Em se tratando dos fluxos TCP, em qualquer cenário de experimentos eles foram iniciados primeiro e permaneceram sempre transmitindo dados durante os 20 *s* iniciais do experimento. Após esse tempo, ou os fluxos UDP ou os fluxos DCCP foram iniciados. O objetivo para tal procedimento foi analisar o *antes e depois*, ou seja, analisar o desempenho do protocolo TCP quando utiliza a rede de forma exclusiva e qual é o impacto no seu comportamento quando novos fluxos UDP ou DCCP são transmitidos na rede.

5.3 Métricas Selecionadas e Métricas Derivadas

Foram analisadas diversas métricas para os fluxos transmitidos nos experimentos. As métricas foram a vazão, a perda de pacote e a latência. Considerando essas métricas, é possível obter outras duas, o *jitter* e a relação quantidade de pacotes perdidos por quantidade de pacotes transmitidos. Através da latência, calcula-se o *jitter* médio para uma determinada transmissão; e atráves da vazão e da quantidade de pacotes perdidos, obtem-se a carga efetiva de dados transmitidos.

Para cada métrica selecionada, foram coletados seus valores instantâneos (por segundo), de acordo com o tempo de duração do respectivo experimento, tal como descrito na Seção 5.2. Contudo, para cada métrica definida, é necessário fazer algumas considerações, as quais são feitas a seguir.

5.3.1 Vazão Média, Carga Efetiva Média, Latência Média e Jitter

Para **um determinado cenário de experimento**, a média final da vazão e da carga efetiva transmitida pelo TCP foi obtida através da média aritmética das médias de cada repetição ***r***, ou seja, através das Equação 5.1 e 5.2, onde ***n*** é o total de repetições. Assim, temos:

$$\mu_{vazaotcp} = \frac{\sum_{r=1}^{n} vazão\_média_r}{n} \tag{5.1}$$

$$\mu_{cargatcp} = \frac{\sum_{r=1}^{n} carga\_média_r}{n} \tag{5.2}$$

No entanto, para obter as médias da vazão e carga efetiva dos protocolos UDP e DCCP, o procedimento foi um pouco diferente. Considerando que foram transmitidos dois fluxos UDP/DCCP – tomados de forma independente – contra apenas um fluxo TCP, e ambos foram sempre iniciados 20 *s* após o fluxo TCP, é preciso definir um mecanismo que não penalize

os dois protocolos, já que eles deixaram de transmitir por $20\,s$, enquanto que o TCP já tinha sido transmitido.

Para não penalizar o protocolo TCP em termos da vazão e carga efetivamente transmitida – já que a disputa foi entre 1 fluxo TCP contra 2 fluxos UDP/DCCP – o cálculo para as médias não poderia ser simplesmente a média aritmética da soma das vazões e das cargas efetivas dos 2 fluxos UDP/DCCP, mas sim a média aritmética das médias das vazões e das cargas efetivas de cada um dos fluxos UDP/DCCP. Assim, tem-se que:

$$\mu_{vazao-parcial(udp/dccp)} = \frac{\sum_{r=1}^{n} \textit{vazão\_média}_r}{n}$$

sendo que,

$$\textit{vazão\_media}_r = \frac{\sum_{k=1}^{F} \textit{vazão\_média\_fluxo}_k}{F}$$

e a $\textit{vazão\_media\_fluxo}_k$ é obtida através da média aritmédia das vazões em cada segundo do experimento. Portanto, o valor final para a vazão dos protocolos UDP e DCCP é obtida através da Equação 5.3.

$$\mu_{vazao-final(udp/dccp)} = \mu_{vazao-parcial(udp/dccp)} + S \times (\frac{\mu_{vazao-parcial(udp/dccp)}}{T}) \tag{5.3}$$

Onde,

- F, número de fluxos utilizados nos experimentos, sendo $F_{UDP} = F_{DCCP} = 2$ para os confrontos TCP $\times$ UDP/DCCP e $F_{DCCP} = 1$ para os confrontos UDP $\times$ DCCP;
- S, o tempo de atraso para iniciar os fluxos UDP ou DCCP ($S = 20\,s$); e
- T, o tempo total do experimento ($T = 100\,s$ ou $T = 300\,s$).

Assim, através da Equação 5.3 as médias são normalizadas para não penalizar nenhum dos protocolos, nos termos discutidos no parágrafo anterior.

De forma equivalente, pode-se obter a carga média efetivamente transmitida e da latência média. Note que para o confronto UDP $\times$ DCCP, temos $F_{DCCP} = 1$, assim a vazão média e carga média são obtidas através das Equações 5.1 e 5.2, respectivamente.

Jitter

O cálculo para obter o valor médio do *jitter* para um fluxo transmitido é bastante similar ao cálculo da vazão média. Este valor pode ser obtido através da Equação 5.5. Esta equação foi obtida da seguinte forma:

$$\mu_{\textit{jitter-parcial(udp/dccp)}} = \frac{\sum_{r=1}^{n} \textit{jitter\_médio}_r}{n} \tag{5.4}$$

onde,

$$\textit{jitter\_médio}_r = \frac{\sum_{k=1}^{F}(\frac{\sum_{k=1}^{QI} VA_k}{QI})}{F}$$

Logo,

$$\mu_{\textit{jitter-final(udp/dccp)}} = \mu_{\textit{jitter-parcial(udp/dccp)}} + S \times (\frac{\mu_{\textit{jitter-parcial(udp/dccp)}}}{T}) \tag{5.5}$$

Sendo,

- F, número de fluxos utilizados nos experimentos, sendo $F_{UDP} = F_{DCCP} = 2$ para os confrontos TCP $\times$ UDP/DCCP e $F_{DCCP} = 1$ para os confrontos UDP $\times$ DCCP;
- QI, quantidade de intervalos ($QI = T - 1$) entre cada medição do experimento, ou seja, entre dois segundos quaisquer consecutivos;
- VA, variação do atraso entre pacotes de um mesmo fluxo, por exemplo para $\textit{tempo}_1 = 10\,\textit{ms}$ e $\textit{tempo}_2 = 11\,\textit{ms}$, $VA = 1\,\textit{ms}$;
- T, o tempo total do experimento ($T = 100\,s$ ou $T = 300\,s$).

5.4 Metodologia Estatística para o Cálculo Final das Métricas Estudadas

Os resultados apresentados neste trabalho, por exemplo, para determinar que um protocolo obteve melhor desempenho que outro em termos da vazão média, foram baseados em amostras dos dados coletados em cada experimento. A metodologia adotada baseia-se no conceito de intervalo de confiança [Jan91], considerando $\rho = 95\,\%$ (nível de confiança) e portanto $\alpha = 5\,\%$ (nível de significância, ou erro).

Determinando o Intervalo de Confiança para $\rho = 95\,\%$

O princípio do intervalo de confiança é baseado no fato de que é impossível determinar uma média perfeita μ para uma população de infinitas amostras N, considerando um número finito n de amostras $\{x_1, ..., x_n\}$. Porém, em termos probabilísticos é possível determinar um intervalo em que μ estará dentro dele com probabilidade igual a ρ e que estará fora dele com probabilidade igual a α.

Para determinar o valor mínimo c_1 e um valor máximo c_2 deste intervalo, chamado de intervalo de confiança, considera-se uma probabilidade $1 - \alpha$, tal que o valor μ esteja dentro desde intervalo de confiança, para n repetições de um determinado experimento realizado. Assim, temos a seguinte relação:

$$Probabilidade\{c_1 \leq \mu \leq c_2\} = 1 - \alpha \tag{5.6}$$

onde,

- (c_1, c_2) é o intervalo de confiança;
- α é o nível de significância, expresso como uma fração e tipicamente perto de zero, por exemplo, $0,05$ ou $0,1$;
- $(1 - \alpha)$ é o coeficiente de confiança; e
- ρ = 100 * $(1-\alpha)$, é o nível de confiança, tradicionalmente expresso como porcentagem e tipicamente perto de 100 %, por exemplo, 90 % ou 95 %.

Assim, através do *Teorema do Limite Central*[2] [Jan91], se um conjunto de amostras $\{x_1, ..., x_n\}$ são independentes, tem uma média $\bar{x}$ e pertencem a uma mesma população N, com média μ e desvio padrão σ, então a média das amostras tende a distribuição normal com $\bar{x} = \mu$ e desvio padrão $\sigma/\sqrt{n}$:

$$\bar{x} \simeq N(\mu, \frac{\sigma}{\sqrt{n}}) \tag{5.7}$$

Então, tendo como base a relação 5.6 e o *Teorema do Limite Central* (5.7), obtem-se o intervalo de confiança (c_1, c_2) para $\rho = 95\%$ e $\alpha = 0.05$ da seguinte forma:

$$(\mu - z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}, \mu + z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}) \tag{5.8}$$

onde,

- μ é a média para n repetições;
- $z_{1-\alpha/2}$ é igual a 1.96. Esse valor determina 95 % para o nível de confiança, como definido na Tabela A.2, do Apêndice A, da referência [Jan91];
- n é igual ao número de repetições; e
- s é o desvio padrão das médias para as n repetições.

Com relação ao valor 1.96 para o termo $z_{1-\alpha/2}$, também chamado de quantil, este é baseado no *Teorema do Limite Central* e por ser freqüentemente utilizado, encontra-se na tabela de *Quantis da Unidade de Distribuição Normal*. Esta tabela pode ser encontrada no apêndice A, Tabela A.2, da referência [Jan91]. Para determinar este valor, temos:

[2]Teorema do Limite Central: expressa o fato de que qualquer soma de muitas variáveis aleatórias independentes e com mesma distribuição de probabilidade tende a distribuição normal.

$$z_{1-\alpha/2} = (1 - 0.05)/2 = 0.975 \tag{5.9}$$

O valor correspondente ao resultado da Equação 5.9, que será o valor da variável z, é igual a 1.96, segundo a tabela *Quantis da Unidade de Distribuição Normal*.

Portanto, baseando-se nos intervalos de confiança para cada média das métricas calculadas de acordo com a Seção 5.3.1, é possível realizar comparações com estes valores segundo os cenários de experimentos para 95 % de confiança com 5 % de erro.

Determinando o Valor de n para obter $\rho = 95\,\%$

O nível de confiança depende da quantidade n de amostras coletadas para um dado experimento. Assim, quanto maior o valor de n, maior será o nível de confiança. Entretanto, obter uma quantidade grande de amostras exige mais esforço e tempo. Portanto, é importante definir o valor de n de tal forma que consiga-se poupar esforço e tempo, porém mantendo o nível de confiança desejado, ou seja, $\rho = 95\,\%$.

Para iniciar o processo, utilizamos uma quantidade pequena $n_{base} = 3$ de amostras preliminares, por exemplo, 3 valores da vazão para um determinado fluxo transmitido. O objetivo é obter um valor alto para a variância, a qual é utilizada para determinar o valor de n repetições necessárias para 95 % de nível de confiança.

Como vimos através da relação 5.8, temos que o intervalo de confiança para uma quantidade n de amostras é defindo da seguinte forma:

$$\mu \pm z \times \frac{s}{\sqrt{n}} \tag{5.10}$$

Assim, para um nível de confiança $\rho = 95\,\%$ e $\alpha = 0.05$, o intervalo de confiança é:

$$(\mu(1 - 0.05), \mu(1 + 0.05)) \tag{5.11}$$

Então, igualando os intervalos de confiança 5.11 ao intervalo de confiança 5.10 (geral), obtemos a Equação 5.12.

$$\mu \pm z \times \frac{s}{\sqrt{n}} = \mu(1 \pm 0.05) \tag{5.12}$$

Portanto, organizando a expressão para isolar a variável n, cada experimento foi repetido n vezes, já contando com as 3 vezes iniciais (n_{base}), através da Equação 5.13, para um nível de confiança $\rho = 95\,\%$, o que implica em $z = 1.96$ (a partir da Equação 5.9).

$$n = (\frac{1.96 \times s}{0.05 \times \mu})^2 \tag{5.13}$$

5.5 Metodologia para Comparação da Qualidade de Áudio

Existem basicamente dois métodos para comparar a qualidade de um arquivo de áudio: o método subjetivo e o método objetivo.

Nas abordagens que utilizam o método subjetivo, geralmente o interessado em comparar os áudios solicita que um conjunto (geralmente grande) de pessoas escute os áudios transmitidos e comparem com o áudio original. Em seguida, essas pessoas fornecem uma pontuação indicando um valor de qualidade. Uma abordagem bastante utilizada é a MOS (*Mean Opinion Score*) [IT96b] que define uma tabela que relaciona uma pontuação de 0 à 100 com a qualidade do áudio, onde 0 significa muito ruim e 100 muito boa. Esse mecanismo é utilizado em um dos trabalhos relacionados que será discutido no Capítulo 7.

Por outro lado, a metodologia objetiva pode ser útil quando se deseja realizar utilizar métodos estatísticos para determinar a qualidade do áudio transmitido através da rede, também baseando-se na qualidade do áudio original. A comparações entre o áudio original e os áudios transmitidos na rede por cada protocolo é baseado em amostras recuperadas tanto do áudio original quanto do áudio transmitido.

Existem diversos mecanismo que são utilizados nesta abordagem, uma delas é a RMS (*Root Mean Square*). Esta foi a abordagem adotada neste trabalho para determinar a qualidade de um áudio transmitido na rede. Esta abordagem foi utilizada porque permite a plotagem de gráficos, o que nos habilitou realizar uma análise visual do conteúdo do áudio original com o áudio transmitido de acordo com a duração da transmissão. Além disso, utilizou-se essa abordagem porque com ela é possível relacionar as perdas de informações que ocorrem durante a transmissão do fluxo de áudio e verificar o impacto dessa perda na qualidade do áudio.

Especificamente a RMS é uma medida estatística da magnitude de uma quantidade variável. Pode-se calcular para uma série de valores discretos ou para uma função variável contínua. O nome deriva do fato de que é a raiz quadrada da média aritmética dos quadrados dos valores discretos, ou seja:

$$RMS = \sqrt{\frac{(x_1)^2 + (x_2)^2 + (x_3)^2 + ... + (X_n)^2}{n}} \tag{5.14}$$

Para o caso específico dos áudios utilizados nos experimentos, a cada $8\,ms$ o valor de RMS foi coletado dos dois canais do áudio, haja vista que o formato de áudio transmitido foi um arquivo *MP3* de $44100\,Mhz$, a uma taxa de $128\,Kbits/s$.

Capítulo 6

Resultados e Discussões dos Experimentos

"A coisa mais bela que podemos experimentar é o mistério. Essa é a fonte de toda a arte e ciências verdadeiras." (Albert Einstein)

Após as definições no Capítulo 4 dos cenários experimentais de rede e das metologias adotadas e discutidas no Capítulo 5, neste capítulo são apresentados os resultados e discussões acerca dos experimentos realizados neste trabalho.

Nos capítulos anteriores foram cobertos todos os conceitos importantes e considerações relacionadas a este trabalho. Por isso, as discussões elencadas neste capítulo serão focadas diretamente na apresentação dos resultados, procurando-se apontar as referências para cada ponto discutido utilizando os capítulos anteriores.

A apresentação dos resultados está dividida também em três fases, correspondendo as fases descritas no Capítulo 4:

1. os resultados dos experimentos utilizando dispositivos com recursos limitados;
2. os resultados dos experimentos utilizando computadores de maior poder de processamento de dados, se comparados com os dispositivos utilizados na primeira fase; e
3. os resultados dos experimentos realizados na Internet.

6.1 Resultados da Fase 1

Os resultados desta fase estão divididos em duas tabelas considerando o tamanho do pacote utilizado durante a transmissão (Seção 5.1). Na Tabela 6.1, apresenta-se os resultados das métricas avaliadas para as transmissões utilizando pacotes de 512 *bytes*. Na Tabela 6.2, apresenta-se os resultados para as transmissões utilizando pacotes de 1424 *bytes*.

Retomando as discussões realizadas na Seção 5.4, para os valores apresentados nestas tabelas, considere 95 % de confiança, com 5 % de erro. O intervalo de confiança é apresentado imediatamente abaixo do valor da métrica correspondente. Para os protocolos UDP e DCCP o intervalo de confiança para a métrica *Transmitido / Perdido*, corresponde ao valor da carga efetiva de dados transmitidos, ou seja, a subtração $Transmitido - Perdido$. Considere também que os valores apresentados nas duas tabelas correspondem a execução dos experimentos segundo os procedimentos descrito nas Seções 5.1, 5.2 e 5.3. Um resumo dessas considerações é apresentado a seguir:

- tempo de execução: 300 *s*;
- execução de *hand-off* em 100 *s* e em 200 *s*;
- confrontos entre os protocolos:
  - TCP $\times$ UDP
  - TCP $\times$ DCCP
  - UDP $\times$ DCCP
- algoritmos de controle de congestionamento dos protocolos:
  - TCP: Reno, Cubic e Veno
  - DCCP: CCID-2 e CCID-3
- métricas avaliadas:
  - vazão
  - quantidade de dados transmitidos e perdidos
  - jitter

Essas considerações também foram adotadas nos resultados da Fase 2, exceto duas: o tempo dos experimentos foi de 100 *s* e não ocorreram execuções de *hand-offs*.

Tabela 6.1: Sumário dos Resultados da Fase 1 para $\rho = 95\,\%$. Pacotes de tamanho 512 *bytes*, com execução de hand-off e considerando os confrontos dois-a-dois entre os protocolos TCP, UDP e DCCP.

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
1	**TCP Reno** × **UDP**	$3359,12$ $(3202,15 - 3516,09)$	$123006,72$ $(117258,69 - 128754,75)$	$2,11$ $(1,72 - 2,21)$	11
		$2956,09$ $(2935,81 - 2976,36)$	$394825,25/293967,67$ $(100164,37 - 101550,79)$	$1,60$ $(1,57 - 1,64)$	
2	**TCP Cubic** × **UDP**	$3364,12$ $(3232,15 - 3496,09)$	$121855,28$ $(118258,44 - 125452,12)$	$7,51$ $(6,32 - 8,7)$	5
		$2919,76$ $(2893,96 - 2945,57)$	$419803,17/320187$ $(98735,37 - 100496,97)$	$1,71$ $(1,69 - 1,74)$	
3	**TCP Veno** × **UDP**	$3121,69$ $(3042,26 - 3201,13)$	$114322,77$ $(111412,94 - 117232,60)$	$4,2$ $(3,4 - 5,0)$	7
		$3002,75$ $(2994,15 - 3011,36)$	$378833,83/276384,92$ $(102150,85 - 102746,97)$	$1,5$ $(1,50 - 1,54)$	
4	**TCP Reno** × **DCCP-2**	$3042,90$ $(2951,57 - 3134,23)$	$111433,62$ $(108090,21 - 114777,03)$	$4,8$ $(4,36 - 5,24)$	9
		$2162,51$ $(2138,56 - 2186,47)$	$73688,67/287,51$ $(73401,16 - 74234,14)$	$5,4$ $(5,02 - 5,75)$	
5	**TCP Cubic** × **DCCP-2**	$3862,58$ $(3775,82 - 3949,34)$	$104830,49$ $(101653,65 - 108007,32)$	$3,32$ $(3,11 - 3,53)$	9
		$2119,10$ $(2109,87 - 2128,33)$	$72256,17/367,92$ $(71888,25 - 72215,14)$	$4,2$ $(4,02 - 4,48)$	
6	**TCP Veno** × **DCCP-2**	$2395,97$ $(2289,27 - 2502,67)$	$87744,27$ $(83836,15 - 91652,39)$	$7,81$ $(7,33 - 8,29)$	20
		$2899,25$ $(2810,27 - 2988,24)$	$64653,08/314,42$ $(61289,36 - 67387,96)$	$6,1$ $(5,50 - 6,64)$	
7	**TCP Reno** × **DCCP-3**	$3291,67$ $(3265,40 - 3317,94)$	$120549,10$ $(119587,17 - 121511,03)$	$3,6$ $(3,42 - 3,78)$	6
		$2851,59$ $(2841,67 - 2861,52)$	$97234/1181,92$ $(95725,77 - 96378,39)$	$0,85$ $(0,83 - 0,87)$	
8	**TCP Cubic** × **DCCP-3**	$3598,81$ $(3496,79 - 3700,84)$	$131790,21$ $(128052,30 - 135528,13)$	$4,13$ $(3,77 - 4,49)$	8
		$2665,55$ $(2571,62 - 2759,48)$	$90895/1533,67$ $(86127,94 - 92594,72)$	$1,18$ $(0,95 - 1,41)$	
9	**TCP Veno** × **DCCP-3**	$3734,55$ $(3634,92 - 3834,18)$	$136765,27$ $(133115,41 - 140415,13)$	$2,31$ $(2,15 - 2,47)$	11
		$2824,84$ $(2815,48 - 2834,20)$	$96381,08/1472,58$ $(92753,94 - 97063,06)$	$0,89$ $(0,85 - 0,93)$	

Continuação na próxima página

Tabela 6.1 – continuação da página anterior

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
10	**DCCP-2** × **UDP**	$1792,20$ $(1749,55-1834,86)$	$65194,33/303,33$ $(62404,79-64891,03)$	$4,91$ $(4,75-5,08)$	11
		$2552,41$ $(2475,76-2629,06)$	$562465,08/475381$ $(84469,96-89698,20)$	$2,02$ $(1,87-2,18)$	
11	**DCCP-3** × **UDP**	$2519,84$ $(2461,98-2577,70)$	$91559,83/1696,83$ $(85041,99-94684,01)$	$1,01$ $(0,91-1,12)$	13
		$2898,34$ $(2841,75-2954,93)$	$427356,58/328470,17$ $(96902,29-100870,53)$	$1,82$ $(1,73-1,92)$	

Tabela 6.2: Sumário dos Resultados da Fase 1 para $\rho = 95\,\%$. Pacotes de tamanho 1424 *bytes*, com execução de hand-off e considerando os confrontos dois-a-dois entre os protocolos TCP, UDP e DCCP.

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
1	**TCP Reno** × **UDP**	$3162,41$ $(2968,61 - 3156,21)$	$112156,65$ $(108719,62 - 115593,67)$	$3,21$ $(3,09 - 3,33)$	4
		$5773,26$ $(5493,16 - 6053,36)$	$730078,67/659256,67$ $(67701,48 - 73942,52)$	$2,37$ $(2,28 - 2,47)$	
2	**TCP Cubic** × **UDP**	$3201,33$ $(3063,57 - 3339,09)$	$80614,73$ $(75572,17 - 85657,29)$	$4,51$ $(4,42 - 4,60)$	6
		$2575,34$ $(2544,25 - 2606,43)$	$1213656,83/1182062,33$ $(27053,38 - 36135,62)$	$3,17$ $(3,05 - 3,29)$	
3	**TCP Veno** × **UDP**	$3221,97$ $(3072,96 - 3370,99)$	$117998,49$ $(112542,56 - 123454,43)$	$5,12$ $(5,04 - 5,2)$	8
		$2301,01$ $(2288,97 - 2313,06)$	$682801,67/605503,92$ $(73433,54 - 81161,96)$	$2,27$ $(2,20 - 2,35)$	
4	**TCP Reno** × **DCCP-2**	$3776,63$ $(3717,94 - 3835,31)$	$166004,14$ $(163776,65 - 168231,63)$	$6,3$ $(6,21 - 6,34)$	4
		$3372,64$ $(3217,38 - 3527,90)$	$141343,33/375,92$ $(139722,68 - 142963,98)$	$6,68$ $(5,40 - 7,97)$	
5	**TCP Cubic** × **DCCP-2**	$3969,39$ $(3901,75 - 4037,04)$	$172124,00$ $(169648,29 - 174599,72)$	$4,12$ $(4,03 - 4,21)$	3
		$3611,59$ $(3578,69 - 3644,49)$	$187001,58/1303,25$ $(182247,13 - 189149,53)$	$5,16$ $(4,90 - 5,43)$	
6	**TCP Veno** × **DCCP-2**	$4561,88$ $(4218,12 - 4905,64)$	$180149,85$ $(178025,55 - 182274,15)$	$6,51$ $(5,81 - 7,21)$	3
		$2685,21$ $(2442,01 - 2928,41)$	$13600,533/1150$ $(133295,41 - 136415,25)$	$6,25$ $(5,47 - 7,02)$	
7	**TCP Reno** × **DCCP-3**	$2735,82$ $(2576,05 - 2895,60)$	$100189,53$ $(94338,37 - 106040,69)$	$3,71$ $(3,28 - 4,14)$	7
		$3469,30$ $(3299,87 - 3638,73)$	$118268,52/5685,50$ $(111483,27 - 113682,77)$	$2,80$ $(2,56 - 3,05)$	
8	**TCP Cubic** × **DCCP-3**	$2980,47$ $(2950,09 - 3010,85)$	$109147,54$ $(108034,15 - 110260,92)$	$4,1$ $(3,49 - 4,71)$	5
		$3482,36$ $(3319,60 - 3645,13)$	$118835,16/1549,83$ $(112236,44 - 122334,22)$	$2,63$ $(2,34 - 2,92)$	
9	**TCP Veno** × **DCCP-3**	$2998,39$ $(2867,97 - 3128,80)$	$109806,54$ $(105029,84 - 114583,24)$	$2,33$ $(2,21 - 2,45)$	6
		$4831,47$ $(4576,48 - 5086,45)$	$184765,16/4835,08$ $(175018,62 - 180930,08)$	$1,69$ $(1,65 - 1,73)$	

Continuação na próxima página

Tabela 6.2 – continuação da página anterior

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
10	**DCCP-2** × **UDP**	$3452,93$ $(3315,24 - 3590,62)$	$125611,37/482,50$ $(122601,76 - 127655,98)$	$7,81$ $(7,55 - 8,07)$	11
		$5236,62$ $(4892,45 - 5580,79)$	$806485,67/742278,17$ $(62959,4 - 65455,6)$	$4,30$ $(3,70 - 4,90)$	
11	**DCCP-3** × **UDP**	$4053,65$ $(3904,76 - 4202,54)$	$147460,73/3415$ $(142043,27 - 146048,19)$	$1,84$ $(1,63 - 2,04)$	13
		$5935,79$ $(5884,42 - 5987,15)$	$699595,58/626779,58$ $(71211 - 74421)$	$2,46$ $(2,42 - 2,51)$	

6.1.1 Discussões sobre os Experimentos da Fase 1

Nesta fase existem basicamente 3 pontos a serem discutidos:

1. verificar se a mudança do tamanho do pacote ocasiona alguma alteração considerável durante as transmissões com os protocolos estudados;

2. analisar o impacto causado em termos da vazão e da quantidade de dados transmitidos e perdidos quando efetua-se *hand-off* durante a transmissão de dados;

3. analisar o comportamento dos protocolos TCP, UDP e DCCP em termos de eqüidade (*fairness*).

Para o primeiro item, não foram constatadas fortes alterações de comportamento dos fluxos transmitidos por nenhum dos protocolos isoladamente, principalmente com relação às métricas vazão e *jitter*. Porém, observa-se algumas alterações no desempenho dos algoritmos TCP $Reno$, $Cubic$ e $Veno$ para os confrontos TCP × UDP e TCP × DCCP.

Para os confrontos TCP × UDP, a quantidade de dados transmitidos utilizando os algoritmos $TCPReno$, $Cubic$ e $Veno$ não foi satisfatória, considerando o agrupamento dos experimentos em dois grupos: um com os experimentos utilizando pacotes de tamanho 512 *bytes* (Tabela 6.1) e o outro contendo os experimentos utilizando pacotes de tamanho 1424 *bytes* (Tabela 6.2). Ora, se as vazões dos protocolos TCP e UDP são praticamente as mesmas para os dois grupos de experimentos – isto pode ser observado nas linhas 1, 2 e 3 – então quanto maior o tamanho do pacote, maior a quantidade de dados que aquele fluxo deveria transmitir, considerando-se que o MTU da conexão *802.11g* é de 1500 *bytes*, não ocorrendo portanto fragmentação de pacotes na camada de rede.

Por outro lado, não se observa este fato para os confrontos TCP × DCCP, independente dos algoritmos utilizados. Comparando-se os valores das vazões obtidas nos confrontos

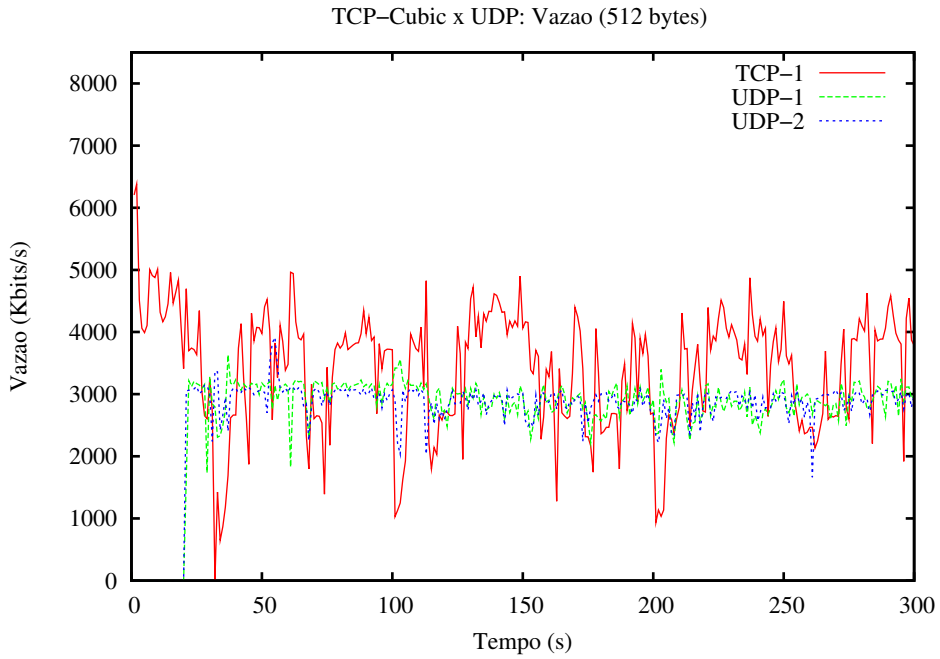
TCP $\times$ UDP e TCP $\times$ DCCP apresentadas nas Tabelas 6.1 e 6.2, observa-se um equilíbrio entre estes valores. Por exemplo, nas Tabelas 6.1 e 6.2, a vazão média do TCP $Reno$ foi de 3359,12 *Kbits/s* e 3162,41 *Kbits/s*, respectivamente. Assim, se as vazões são praticamente as mesmas, então quanto maior o tamanho do pacote, mais dados aquele fluxo deveria ter transmitido. Isto aconteceu apenas para os confrontos TCP $\times$ DCCP. Neste caso, os três algoritmos do TCP transmitiram mais dados quando o tamanho do pacote foi de 1424 *KBytes*. Isto era de se esperar, uma vez que o protocolo DCCP implementa controle de congestionamento e portanto permite que os fluxos TCP transmitam mais dados à medida que se aumenta o tamanho do pacote, considerando que o tamanho máximo para um pacote o valor do MTU menos o espaço ocupado no pacote pelos cabeçalhos dos protocolos das camadas de rede, transporte e de aplicação.

Portanto, pode-se concluir que, em transmissões onde os protocolos TCP e UDP compartilham o mesmo canal de comunicação, aumentar o tamanho do pacote de 512 *KBytes* para 1424 *KBytes* não obtem-se melhorias consideráveis, mesmo considerando o algoritmo de controle de congestionamento $Veno$. Isto sugere que internamente o TCP perdeu mais dados quando foram utilizados pacotes de 1424 *KBytes*. Neste caso, futuros estudos poderão avaliar qual deve ser o tamanho ideal do pacote de dados para minimizar a quantidade de pacotes perdidos por parte do TCP. Uma discussão neste contexto é apresentada em [WCSY07].

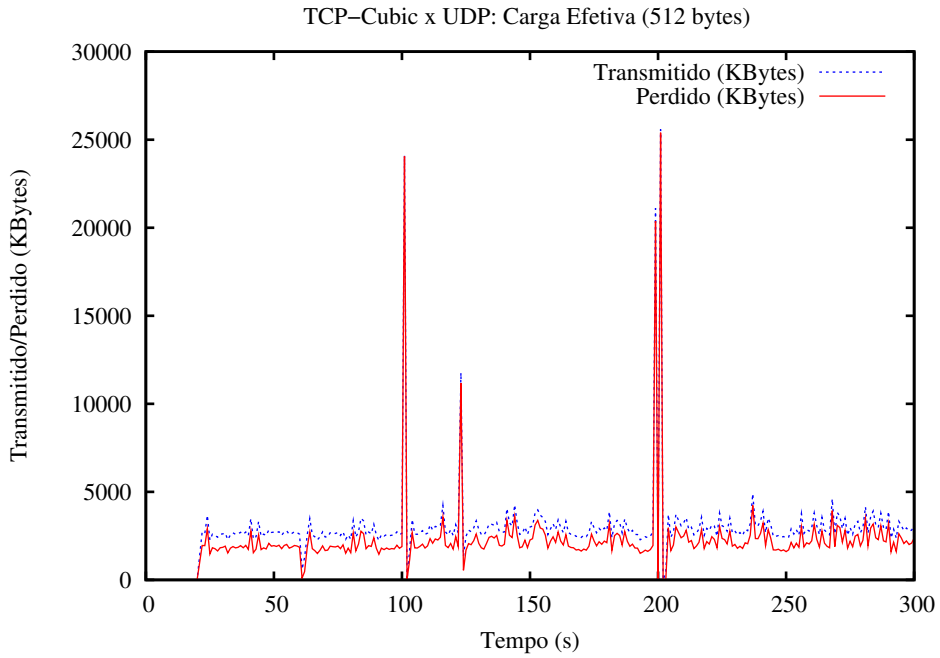
Com relação às execuções dos *hand-offs*, nos gráficos da Figura 6.1 é apresentada a evolução da transmissão do confronto TCP Cubic $\times$ UDP, que corresponde à linha 2 da Tabela 6.1. No gráfico da Figura 6.1(a), apresenta-se os valores médios da vazão alcançada pelos protocolos TCP e UDP. No gráfico da Figura 6.1(b), apresenta-se a relação quantidade de dados transmitidos pela quantidade de dados perdidos pelo protocolo UDP para este confronto. Os valores dos pontos plotados nos gráficos da Figura 6.1, foi calculado através da média aritimética dos valores de cada ponto para todas as repetições realizadas do experimento, neste caso para $n = 5$.

Note que, no gráfico da Figura 6.1(a), a taxa de transmissão do protocolo TCP diminuiu devido às execuções de *hand-offs*, onde ocorreram perdas de pacotes, o que se reflete nos algoritmos de controle de congestionamento do TCP e do DCCP. No caso do protocolo UDP, a taxa de transmissão permaneceu constante durante todo o tempo. No gráfico da Figura 6.1(b), constata-se um nível elevado de perda de dados quando se utiliza o protocolo UDP, sobretudo durante a execução dos *hand-offs*. É possível observar em torno do segundo 35, outro ponto de declínio da taxa de transmissão do protocolo TCP. Este fato pode ser explicado pelo início das transmissões dos dois fluxos UDP, onde também ocorreram perdas de pacotes.

Já nos gráficos da Figura 6.4 é apresentada a evolução da transmissão para o confronto TCP-Cubic $\times$ DCCP, que corresponde à linha 5 da Tabela 6.1. Os valores de cada ponto destes gráficos foram calculados da mesma forma que nos gráficos anteriores, com $n = 9$. Esta procedimento também foi adotado em todos os outros gráficos apresentados neste

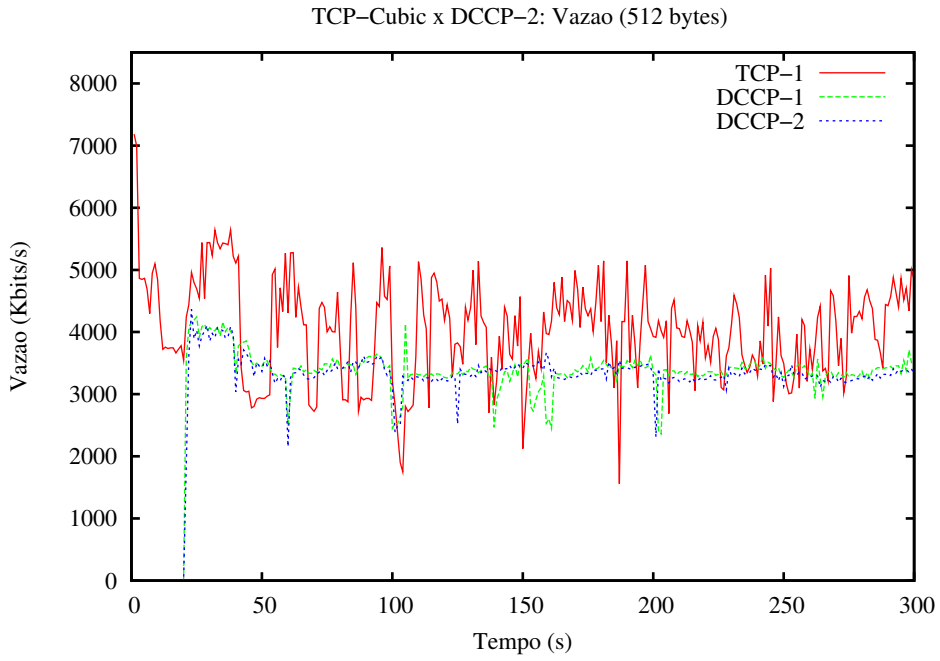


(a) Vazão alcançada pelos protocolos TCP e UDP.

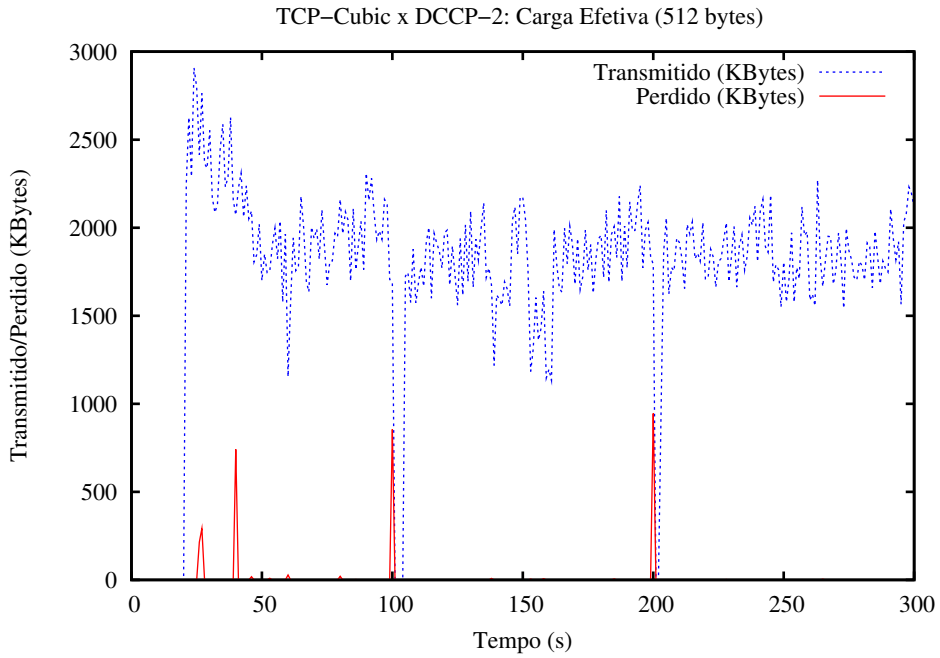


(b) Relação da quantidade de dados transmitidos pela quantidade de dados perdidos do protocolo UDP.

Figura 6.1: Fase 1: TCP × UDP: vazão e carga efetiva do confronto TCP-Cubic × UDP durante uma transmissão de 300 *s*, com tamanho de pacote 512 *bytes* e execução de *hand-offs* em 100 *s* e em 200 *s*.



(a) Vazão alcançada pelos protocolos TCP e DCCP.



(b) Relação da quantidade de dados transmitidos pela quantidade de dados perdidos do protocolo DCCP.

Figura 6.2: Fase 1: TCP × DCCP: vazão e carga efetiva do confronto TCP-Cubic × DCCP-2 durante uma transmissão de 300 *s*, com pacotes de 512 *bytes* e *hand-offs* em 100 *s* e em 200 *s*.

capítulo, exceto para os gráficos apresentados como resultados da Fase 3, que corresponde a uma execução do confronto.

No gráfico da Figura 6.2(a), a taxa de transmissão do protocolo TCP também diminuiu devido às execuções de *hand-offs*, pelos mesmos motivos do confronto TCP $\times$ UDP. No caso do protocolo DCCP, ocorreu uma leve diminuição na taxa de transmissão deste protocolo nos momentos de *hand-off*. No gráfico da Figura 6.2(b), constata-se um nível baixíssimo de perda de dados quando se utiliza o protocolo DCCP em confronto com o protocolo TCP. É possível observar que durante todo o tempo de transmissão, os protocolos TCP e DCCP compartilham a utilização do canal de transmissão de forma eqüinime.

Com relação a eqüidade entre os protocolos em termos de quantidade de dados transmitido por cada um dos protocolos, esperava-se que o protocolo UDP congestionasse a rede quando em confrontos com o TCP e com o DCCP, mas este fato ocorreu. Conclui-se então que há contenção de dados na fonte geradora, neste caso os dispositivos $N800$. Como o poder de processamento de dados destes dispositivos é limitado, há uma limitação da velocidade com que os dados são processados e transmitidos na rede, considerando-se que neste caso o processo da aplicação *IPerf* ocupou menos tempo de utilização da CPU se comparado ao tempo de utilização da CPU em um computador *desktop*, por exemplo. Serão apresentadas discussões neste sentido na Seção 6.2.1, onde observou-se um comportamento diferente: o protocolo UDP congestiona a rede para alguns casos e impede que os protocolos TCP e DCCP transmitam dados na rede.

Além das discussões apresentadas nesta seção, cabe relatar dois outros fatos ocorridos durante a execução dos experimentos nesta fase:

- observou-se desconexões repentinas da rede sem fio por partes dos dispositivos N800. Isto provavelmente está associado mais uma vez a capacidade de processamento e de gerenciamento de recursos por parte das aplicações em execução neste dispositivo, em particular da *aplicação* controladora (*driver*) da placa de rede sem fio deste dispositivo. Para uma justificativa mais elaborada deste fato, sugere-se um estudo mais aprofundado, onde os focos seriam analisar se existe alguma situação em que o processador deste dispositivo fica sobre-carregado, o que pode ocasionar tais desconexões; e também verificar se estas desconexões ocorrem devido às execuções dos *hand-offs*, onde por motivos desconhecidos no momento, o *driver* controlador da placa de rede sem fio desconecta do ponto de acesso sem fio;

- além do desempenho ruim do protocolo UDP para transmissão de dados em redes sem fio, principalmente no tocante à quantidade de dados transmitidos e à quantidade dos dados efetivamente recebidos – isto pode ser observado nas Tabelas 6.1 e 6.2, campo *Transmitido / Perdido* – em todas as transmissões realizadas utilizando o protocolo UDP, observou-se que este realizou entrega de pacotes de dados fora de ordem. A entrega de pacotes fora de ordem também ocorreu por parte do protocolo DCCP, porém

em proporções bem menores se comparado ao protocolo UDP. Essas proporções são equivalentes às perdas de pacotes do protocolo DCCP nos confrontos TCP $\times$ DCCP – Tabelas 6.1 e 6.2, campo *Transmitido / Perdido*, entre as linhas 4 e 9 (inclusive).

6.2 Resultados da Fase 2

Como mencionado, as considerações feitas na Seção 6.2 também são aplicadas nesta fase, exceto que não houve execução de *hand-offs*. Desta forma, não faz sentido apresentá-las novamente nesta seção, sendo os resultados apresentados nas Tabelas 6.3 e 6.4.

De acordo com as discussões apresentadas na Seção 6.1.1, constatou-se que o protocolo UDP não degrada a rede se utilizado em dispositivos com recursos limitados, como o $N800$. Assim, os mesmos experimentos realizados na Fase 1 foram repetidos, porém utilizando-se computadores com maior poder de processamento de dados, cujas configurações estão descritas na Seção 4.2.

Desta forma, foi possível avaliar o desempenho dos protocolos TCP e DCCP diante da presença de congestionamento na rede, causado pelo protocolo UDP e por um ponto de saturação, disposto propositalmente entre a rede cabeada e a rede sem fio, ambas formando uma rede local híbrida (WLAN + LAN). Este é um cenário típico em diversas redes em funcionamento atualmente. Nesta fase, os fluxos de dados foram transmitidos a partir de dois computadores conectados à rede LAN para outros dois computadores, conectados à rede sem fio. Este esquema é ilustrado na Figura 4.2. Para discussões acerca dos resultados desta fase, leia a Seção 6.2.1.

Tabela 6.3: Sumário dos Resultados da Fase 2 para $\rho = 95\,\%$ com pacotes e 512 *bytes*, sem execução de hand-off e considerando os confrontos dois-a-dois entre os protocolos TCP, UDP e DCCP.

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
1	**TCP Reno** × **UDP**	$403,08$ $(398,22 - 407,94)$	$7137,00$ $(7065,67 - 7208,33)$	$19,3$ $(18,5 - 20,1)$	4
		$8653,15$ $(8621,21 - 8685,08)$	$991671,33/665869,50$ $(324897,15 - 326706,51)$	$3,25$ $(2,51 - 3,99)$	
2	**TCP Cubic** × **UDP**	$3160,98$ $(3063,34 - 3258,63)$	$35806,33$ $(34882,09 - 36730,58)$	$12,12$ $(11,52 - 12,72)$	6
		$9739,85$ $(9675,66 - 9804,03)$	$993706,58/767118,58$ $(225964,24 - 227211,76)$	$3,45$ $(2,09 - 4,81)$	
3	**TCP Veno** × **UDP**	$3404,60$ $(3396,71 - 3412,49)$	$37139,67$ $(37051,35 - 37227,98)$	$8,09$ $(7,49 - 8,69)$	3
		$9644,66$ $(9589,37 - 9699,94)$	$993140,75/767471,25$ $(225130,83 - 226208,17)$	$5,58$ $(4,26 - 6,90)$	
4	**TCP Reno** × **DCCP-2**	$7929,91$ $(7894,87 - 7964,95)$	$133420,00$ $(132991,36 - 133848,64)$	$2,21$ $(1,55 - 2,87)$	4
		$7108,86$ $(7090,56 - 7127,16)$	$128466,42/1157,05$ $(124094,49 - 130524,25)$	$4,88$ $(4,01 - 5,74)$	
5	**TCP Cubic** × **DCCP-2**	$7231,15$ $(7128,01 - 7334,29)$	$137099,33$ $(135840,99 - 138357,68)$	$2,84$ $(1,64 - 4,03)$	4
		$7071,50$ $(5650,18 - 8492,82)$	$122672,42/2138,83$ $(118693,44 - 122373,74)$	$3,36$ $(2,43 - 4,28)$	
6	**TCP Veno** × **DCCP-2**	$8916,46$ $(7034,24 - 10798,68)$	$133257,00$ $(131990,40 - 134523,60)$	$1,31$ $(1,19 - 1,43)$	5
		$7066,03$ $(7040,40 - 7091,65)$	$122055,00/1149,17$ $(117058,72 - 124752,94)$	$4,52$ $(3,62 - 5,41)$	
7	**TCP Reno** × **DCCP-3**	$6047,47$ $(5886,58 - 6208,37)$	$90477,33$ $(89512,22 - 91442,44)$	$4,03$ $(3,32 - 4,74)$	7
		$8613,24$ $(8541,47 - 8685,00)$	$125339,75/5490,00$ $(118015,7 - 121683,8)$	$4,34$ $(3,10 - 5,58)$	
8	**TCP Cubic** × **DCCP-3**	$7887,37$ $(7747,19 - 8027,56)$	$169521,33$ $(167810,32 - 171232,35)$	$3,21$ $(2,37 - 4,04)$	6
		$6819,78$ $(6665,72 - 6973,83)$	$127375,50/6501,67$ $(119421,28 - 122326,38)$	$1,78$ $(1,08 - 2,48)$	
9	**TCP Veno** × **DCCP-3**	$8551,42$ $(8357,57 - 8745,27)$	$165421,67$ $(163055,34 - 167787,99)$	$2,91$ $(2,72 - 3,1)$	3
		$7743,27$ $(7666,13 - 7820,41)$	$126615,17/5554,25$ $(119728,81 - 122393,03)$	$3,00$ $(1,59 - 4,42)$	

Continuação na próxima página

Tabela 6.3 – continuação da página anterior

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
10	**DCCP-2** × **UDP**	$689,54$ $(665,74 - 713,34)$	$8063,50/63,17$ $(7780,62 - 8220,04)$	$5,37$ $(5,36 - 5,38)$	3
		$8509,98$ $(8955,67 - 8064,28)$	$993123,33/663914,17$ $(28682,90 - 329735,42)$	$3,54$ $(2,62 - 4,46)$	
11	**DCCP-3** × **UDP**	$2195,17$ $(1980,48 - 2409,85)$	$16074,83/1100,83$ $(13507,53 - 16440,47)$	$0,46$ $(0,41 - 0,52)$	6
		$8311,96$ $(8218,60 - 8405,33)$	$993347,50/851499,08$ $(140946,91 - 142749,93)$	$2,13$ $(1,51 - 2,76)$	

Tabela 6.4: Sumário dos Resultados da Fase 2 para $\rho = 95\,\%$ com pacotes e 1424 *bytes*, sem execução de hand-off e considerando os confrontos dois-a-dois entre os protocolos TCP, UDP e DCCP.

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
1	**TCP Reno** × **UDP**	$605,31$ $(519,58 - 691,03)$	$9561,33$ $(9261,94 - 9860,72)$	$23,5$ $(22,57 - 24,43)$	6
		$9653,15$ $(9621,21 - 9685,08)$	$941621,17/615819,85$ $(324957,10 - 326645,74)$	$5,52$ $(4,63 - 6,41)$	
2	**TCP Cubic** × **UDP**	$3379,11$ $(3274,56 - 3483,67)$	$48378,23$ $(47226,18 - 49530,29)$	$16,7$ $(16,37 - 17,03)$	3
		$8490,58$ $(7649,53 - 9331,63)$	$958806,67/732223,67$ $(225359,79 - 227806,21)$	$4,99$ $(4,22 - 5,75)$	
3	**TCP Veno** × **UDP**	$3044,84$ $(2935,93 - 3153,74)$	$45710,47$ $(42292,43 - 49128,50)$	$12,3$ $(11,81 - 12,79)$	7
		$8116,44$ $(8093,97 - 8138,90)$	$907636,75/896732,50$ $(118599,05 - 123209,45)$	$4,25$ $(3,37 - 5,12)$	
4	**TCP Reno** × **DCCP-2**	$5900,49$ $(5799,93 - 6001,06)$	$117914,33$ $(113486,91 - 122341,76)$	$3,2$ $(2,94 - 3,45)$	3
		$8659,69$ $(7297,07 - 9022,31)$	$171000,50/560,50$ $(169580,94 - 171299,06)$	$4,72$ $(4,28 - 4,17)$	
5	**TCP Cubic** × **DCCP-2**	$6171,10$ $(5486,89 - 6855,31)$	$119492,67$ $(115523,62 - 123461,72)$	$4,55$ $(3,99 - 5,11)$	4
		$8788,84$ $(8075,73 - 9501,95)$	$150194,71/1143$ $(148128,05 - 149975,37)$	$3,63$ $(2,60 - 4,66)$	
6	**TCP Veno** × **DCCP-2**	$5981,80$ $(5833,30 - 6130,30)$	$102771,00$ $(98402,81 - 107139,19)$	$4,58$ $(4,34 - 4,82)$	4
		$7419,01$ $(7061,02 - 7777,00)$	$148361,08/1438,67$ $(145482,91 - 148361,91)$	$3,81$ $(2,64 - 4,98)$	
7	**TCP Reno** × **DCCP-3**	$5526,10$ $(5507,88 - 5544,32)$	$83633,60$ $(81323,30 - 85943,90)$	$5,22$ $(4,69 - 5,75)$	4
		$6830,60$ $(6630,57 - 7030,62)$	$95164,17/990,83$ $(95013,15 - 94324,36)$	$6,23$ $(5,34 - 7,12)$	
8	**TCP Cubic** × **DCCP-3**	$6545,19$ $(6164,13 - 6926,25)$	$95965,60$ $(87114,60 - 104816,60)$	$5,21$ $(4,11 - 6,31)$	3
		$7861,36$ $(7136,26 - 8586,47)$	$119854,84/1227,33$ $(117985,30 - 119269,72)$	$3,32$ $(2,64 - 4,00)$	
9	**TCP Veno** × **DCCP-3**	$5532,73$ $(5522,74 - 5542,72)$	$84882,73$ $(84026,41 - 85739,06)$	$4,56$ $(3,9 - 5,22)$	4
		$6908,37$ $(6837,93 - 6978,81)$	$115169,71/1272,67$ $(113703,92 - 114090,16)$	$2,47$ $(1,92 - 3,02)$	

Continuação na próxima página

Tabela 6.4 – continuação da página anterior

#	Confrontos	Vazão (Kbits/s)	Transmitido / Perdido (KBytes)	*Jitter* (ms)	n
10	**DCCP-2** × **UDP**	$1346,47$ $(1157,44 - 1535,51)$	$12377,3/17,17$ $(11928,01 - 12792,25)$	$5,96$ $(5,86 - 6,06)$	7
		$6467,94$ $(6353,74 - 6582,14)$	$911298,75/858344,29$ $(50652,28 - 55256,64)$	$4,38$ $(3,68 - 5,08)$	
11	**DCCP-3** × **UDP**	$1283,19$ $(1257,53 - 1308,85)$	$15350,8/63,43$ $(15044,57 - 15530,17)$	$8,67$ $(8,65 - 8,68)$	8
		$7413,20$ $(7210,90 - 7615,50)$	$809061,67/783139,25$ $(23925,16 - 27919,68)$	$2,95$ $(2,01 - 3,89)$	

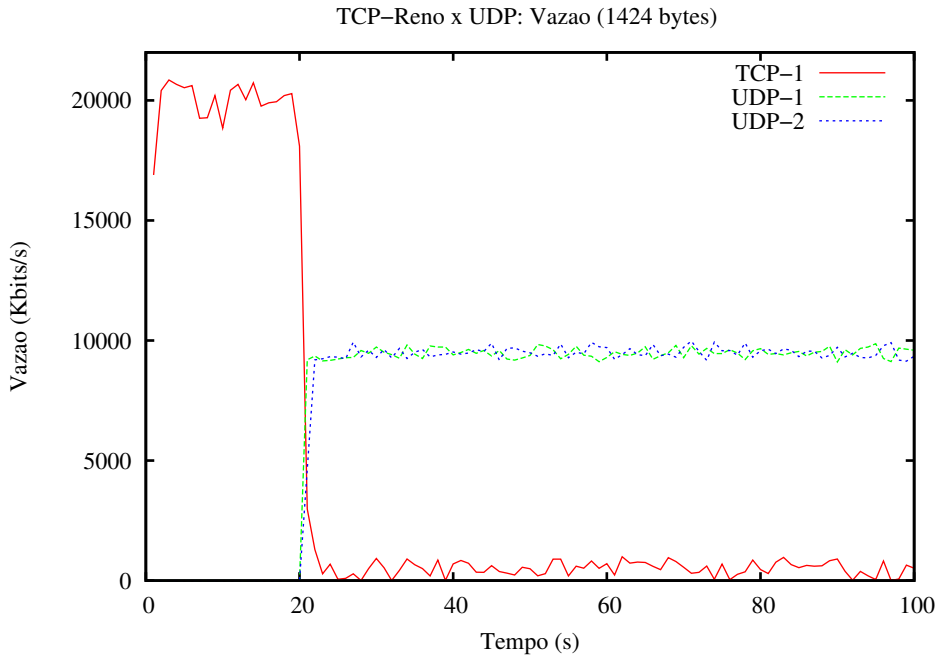
6.2.1 Discussões sobre os Experimentos da Fase 2

A discussão apresentada na Fase 1 com relação à variação do tamanho do pacote, aplica-se também nesta fase. É possível observar nas Tabelas 6.3 e 6.4, linhas 1, 2 e 3, que para as transmissões com pacotes de 1424 *bytes* não houve um aumento significativo na quantidade de dados transmitidos se comparado às transmissões com pacotes de 512 *bytes*. Isto aconteceu de forma bastante acentuada no confronto TCP-Reno × UDP: com pacotes de 512 *bytes*, o TCP Reno transmitiu, em média, 7137 *KBytes*, ao passo que foram transmitidos, em média, apenas 9561,33 *KBytes* quando utilizou-se pacotes de 1424 *bytes*. Para estes casos, a taxa de transmissão do TCP Reno foi de 403,08 *Kbits/s* e 605 *Kbits/s*, respectivamente. Estes valores são muito inferiores quando se compara às taxas de transmissão alcançadas pelos fluxos do protocolo UDP, que em média foi de 9653,15 *Kbits/s*.

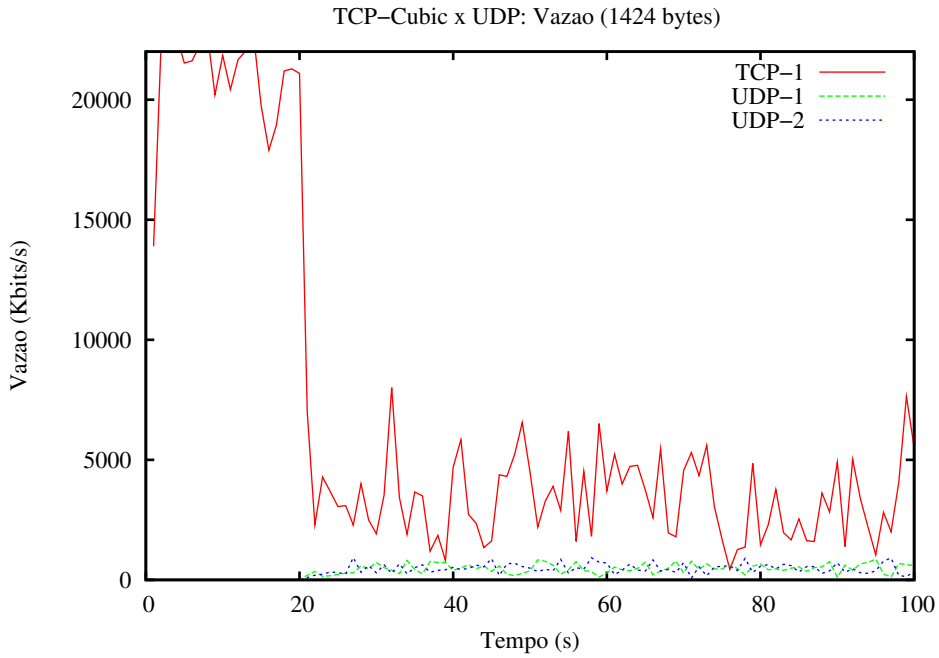
Embora o protocolo UDP tenha alcançado níveis bastante elevados da taxa de transmissão, a quantidade de dados efetivamente entregues no destino foi muito baixa. Por exemplo, para o confronto TCP Veno × UDP com pacotes de 512 *Bytes*, o protocolo UDP transmitiu 993140,75 *KBytes* e perdeu 767471,25 *KBytes*, que corresponde a uma carga efetiva de apenas 225669,50 *KBytes*. Este mesmo fato também ocorreu no confronto TCP Cubic × UDP com pacotes de 1424 *bytes*, porém não ocorreu para o confronto TCP Reno × UDP.

Nos gráficos ilustrados na Figura 6.3 apresenta-se a evolução da taxa de transmissão durante 100 *s* de experimento para estes dois últimos confrontos. Note que, no gráfico da Figura 6.3(a), a taxa de transmissão do protocolo TCP foi muito baixa. Em contrapartida, no gráfico da Figura 6.3(b) ocorreu praticamente uma inversão no comportamento dos protocolos. Note que para ambos os casos, houve uma diminuição bastante acentuada na taxa de transmissão do protocolo TCP quando iniciou-se a transmissão do protocolo UDP.

Por outro lado, para os confrontos TCP × DCCP o comportamento dos protocolos foi claramente diferente. Por exemplo, na Tabela 6.4, ambos os protocolos obtiveram uma vazão



(a) Vazão alcançada pelos protocolos TCP Reno e UDP.



(b) Vazão alcançada pelos protocolos TCP Cubic e UDP.

Figura 6.3: Fase 2: TCP $\times$ UDP: vazão do confronto TCP-Cubic $\times$ UDP durante uma transmissão de 100 *s* utilizando pacotes de 1424 *bytes*.

média bastante equilibrada, sobretudo com relação ao confronto TCP Veno × DCCP-3 (linha 9). Neste caso, a vazão média foi 5532,73 *Kbits/s* e 6908,37 *Kbits/s*, respectivamente. Neste confronto, o TCP Veno transmitiu, em média, uma carga efetiva de 84882,73 *KBytes* e o DCCP, 113897,04 *KBytes*. Isto aconteceu de forma bastante semelhante no confronto TCP Cubic × DCCP-3, como apresentado na linha 8 das Tabelas 6.3 e 6.4.

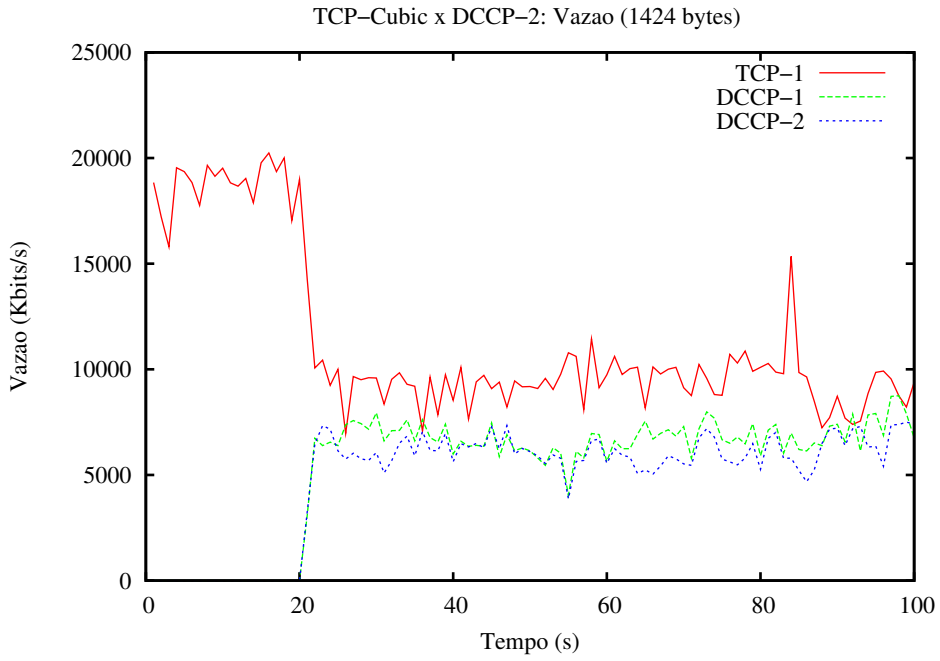
Nos gráficos ilustrados na Figura 6.4, apresenta-se a evolução da taxa de transmissão dos protocolos TCP Cubic e DCCP-3 e a carga efetiva do protocolo DCCP durante 100 *s* de experimento. Note que, no gráfico da Figura 6.4(a), a taxa de transmissão para o protocolo TCP foi bastante similar ao do protocolo DCCP, sendo que o TCP apresentou um desempenho levemente melhor com relação ao DCCP. Já no gráfico da Figura 6.4(b), observa-se que o protocolo DCCP perdeu poucos pacotes de dados, diferentemente dos confrontos TCP × UDP.

Já nas linhas 10 e 11 da Tabela 6.4, apresenta-se os resultados para os confrontos DCCP × UDP. Embora o desempenho do protocolo DCCP tenha sido um pouco melhor que o protocolo TCP nos confrontos TCP × UDP, constata-se similaridades em termos da carga efetiva de dados transmitido pelos protocolos TCP Reno e DCCP, onde o TCP Reno transmitiu, em média, 9561,33 *KBytes* e o DCCP 12360,13 *KBytes*. Neste confronto, a vazão média do DCCP-2 foi de 1346,47 *Kbits/s*, um pouco mais que o dobro do protocolo TCP Reno, que alcançou uma vazão de apenas 605,21 *Kbits/s*. Um comportamento similar é observado no confronto DCCP-3 × UDP.

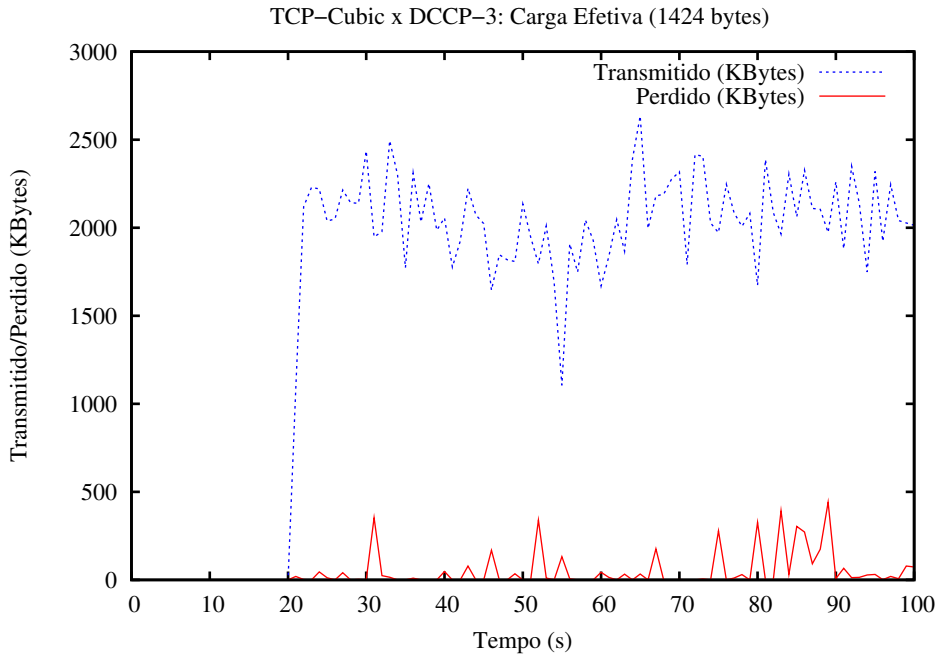
Contudo, observa-se que nos confrontos TCP Cubic × UDP e TCP Veno × UDP, o desempenho do protocolo TCP foi bem superior ao protocolo DCCP, independente do tamanho do pacote utilizado. De acordo com a Tabela 6.4, linhas 2, 3, 10 e 11, utilizando o TCP Cubic e o TCP Veno, a carga de dados efetiva de cada um foi em torno de 47 *MBytes*, enquanto que a carga efetiva do protocolo DCCP foi de aproximadamente 14 *MBytes*, para cada um dos algoritmos de controle de congestionamento do DCCP. Apesar disso, note que o *jitter* médio para as transmissões utilizando DCCP foi relativamente menor que nas transmissões utilizando TCP, o que é melhor para as aplicações multimídia. Um dos motivos para o DCCP obter um *jitter* médio melhor que o TCP estar relacionado ao fato do TCP realizar retransmissão de pacotes.

6.3 Resultados e Discussões da Fase 3

Na última fase dos experimentos, foram transmitidos fluxos TCP, UDP e DCCP na Internet, entre a Universidade Federal de Campina Grande (Brasil) e a Universidade de São Francisco (Estados Unidos), como ilustrado nas Figuras 4.3 e 4.4. O foco foi estudar o desempenho do protocolo DCCP na Internet, uma vez que a proposta é que o DCCP substitua o protocolo UDP em transmissões multimídia na Internet. Foi transmitido um arquivo de áudio no



(a) Vazão alcançada pelos protocolos TCP e DCCP.



(b) Relação da quantidade de dados transmitidos pela quantidade de dados perdidos do protocolo DCCP.

Figura 6.4: Fase 2: TCP $\times$ DCCP: vazão e carga efetiva do confronto TCP Cubic $\times$ DCCP-3 durante uma transmissão de 100 *s* utilizando pacote de 512 *bytes*.

formato mp3, que por sua vez foi armazenado no destino para em seguida ser comparado com o arquivo de áudio original. Para realizar esta comparação, utilizou-se uma abordagem objetiva conhecida por RMS, a qual está descrita na Seção 5.5.

No gráfico ilustrado na Figura 6.5, apresenta-se a evolução da transmissão do arquivo de áudio utilizando o protocolo TCP, enquanto que o protocolo DCCP transmitia para o mesmo sistema remoto fluxos de dados gerados aleatoriamente com pacotes de 1424 *bytes*. Note que o protocolo TCP conseguiu enviar completamente o áudio e com a mesma qualidade do arquivo original, pois observa-se uma sobreposição da linha tracejada azul, que representa o fluxo do áudio transmitido utilizando-se TCP, em RMS (dB), e a linha contínua vermelha, que representa o arquivo original. Caso tivesse ocorrido perdas de pacotes durante a transmissão com o protocolo TCP, haveria um atraso no fluxo de dados, e como resultado, a linha tracejada em azul do respectivo gráfico não alcançaria os 100 *s*. Este fato pode ser observado na Figura 6.6. Neste caso, quando confrontou-se o protocolo TCP Reno e o UDP, com o TCP transmitindo o arquivo de áudio, ocorreu um atraso na transmissão do arquivo multimídia devido a um nível elevado de perda de pacotes por parte do protocolo TCP, que neste caso enviou apenas 30 % do arquivo de áudio original.

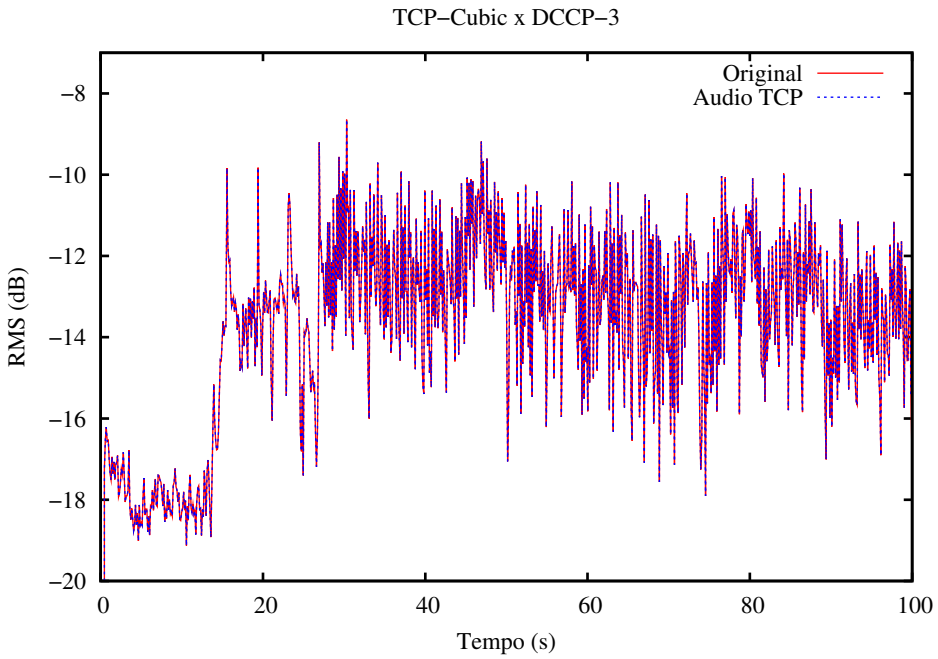


Figura 6.5: Fase 3: TCP Cubic × DCCP-3, sendo o TCP enviando um arquivo de áudio com duração de 100 *s*.

Por outro lado, no confronto TCP Cubic × DCCP-3, com o protocolo DCCP-3 enviando o arquivo de áudio, este conseguiu enviar praticamente todo o áudio. No gráfico da

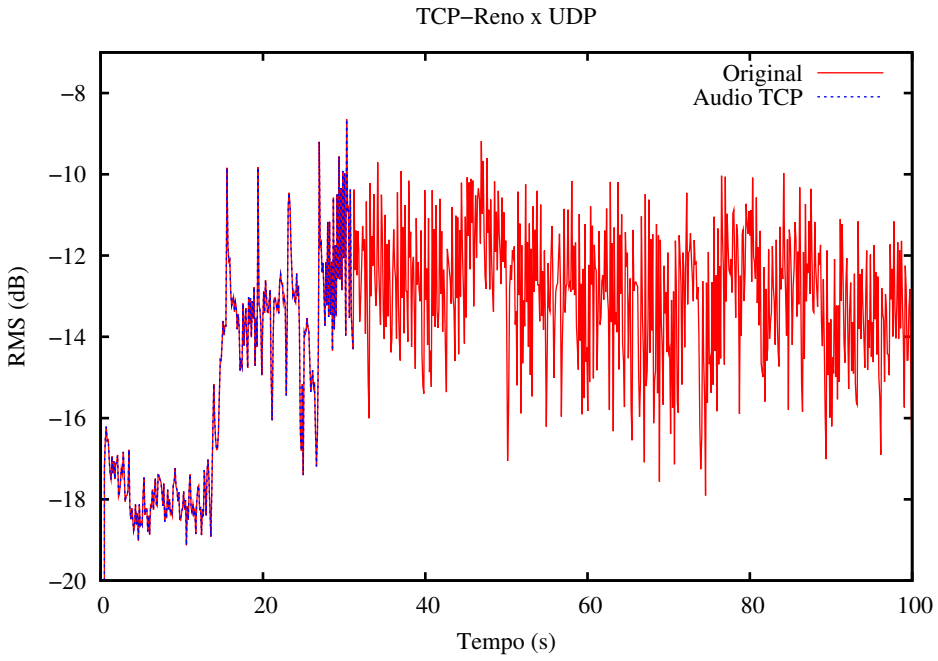


Figura 6.6: Fase 3: TCP Reno × UDP, sendo o TCP enviando um arquivo de áudio.

Figura 6.7, apresenta-se o comportamento deste confronto. Observa-se perdas ocasionais e muito rápidas de dados nos primeiros 10 *s*, entre os segundos 20 *s* e 25 *s* e em três outros pequenos trechos: entre os segundos 70-73 *s*, 78-79 *s* e 89-93 *s*.

Para os confrontos DCCP × UDP, observa-se perdas de informação por parte de ambos os protocolos, sobretudo quando se utilizou o protocolo UDP para enviar o arquivo de áudio. No gráfico ilustrado na Figura 6.8, que representa o arquivo de áudio transmitido utilizando o protocolo DCCP-2, observa-se grandes perdas de informação. E na Figura 6.9, quando o UDP foi utilizado para transmitir o arquivo de áudio, observa-se perdas de informação do áudio ainda mais acentuadas. Em resumo, quanto mais a linha vermelha contínua for visível nos gráficos apresentados nesta seção, maior é a quantidade de informações perdidas durante a transmissão.

Por fim, com relação às estatísticas gerais dos experimentos realizados, foram executados 295 experimentos, sendo 180 na Fase 1, 104 na Fase 2 e 11 na Fase 3.

6.4 Conclusões sobre os Resultados Obtidos

Considerando as discussões apresentadas neste capítulo, pode-se apresentar as seguintes conclusões:

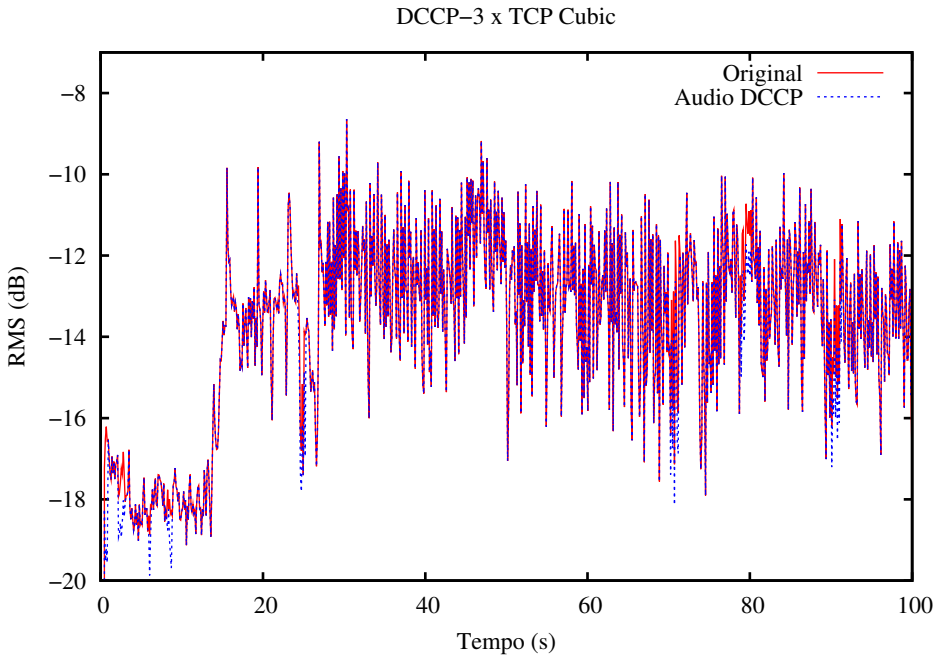


Figura 6.7: Fase 3: TCP Cubic × DCCP-3, sendo o DCCP enviando o arquivo de áudio.

1. o protocolo UDP quando utilizado em dispositivos com recursos limitados não congestiona a rede, justificando-se tal fato através da contenção na fonte geradora dos dados, neste caso nos dispositivos N800.

2. embora o protocolo UDP não tenha congestionado a rede quando utilizado nos experimentos da Fase 1, sua utilização é fortemente desencorajada, pelo menos nas topologias de rede utilizadas nesta dissertação. Esta recomendação é baseada nas constatações de que o UDP perde uma grande quantidade de informação, sobretudo quando a rede está congestionada. Isto se reflete diretamente na qualidade do fluxo multimídia sendo transmitido, o que foi discutido na seção de resultados da Fase 3 dos experimentos executados. Além dos altos índices de perda de dados por parte do protocolo UDP, ele ainda impede um melhor funcionamento de outros protocolos que implementam controle de congestionamento, como o TCP e o DCCP.

3. diferentemente do que foi discutido no trabalho referenciado em [SAPJ08], onde também são apresentados partes dos resultados obtidos nesta dissertação, a execução dos *hand-offs* durante a transmissão dos dados não afetou o estado dos algoritmos de controle de congestionamento dos protocolos TCP e DCCP. Nos resultados dos experimentos apresentados em [SAPJ08], não foram utilizados dispositivos com recursos limitados, e sim *notebooks*. Existem duas hipótese que explica este fato: utilizando

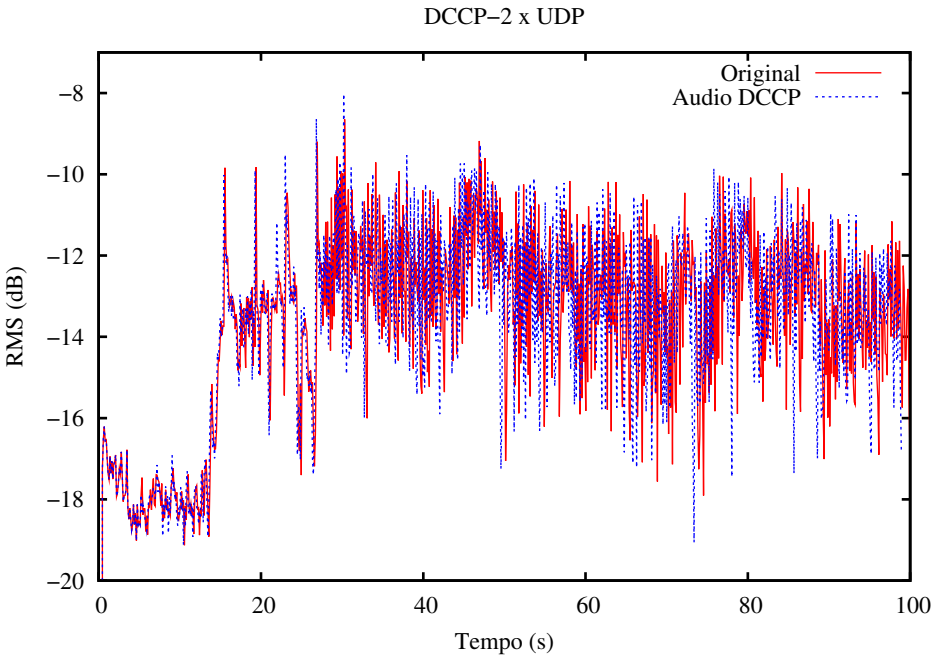


Figura 6.8: Fase 3: DCCP-2 × UDP, sendo o DCCP enviando um arquivo de áudio.

dispositivos como o N800, a execução do *hand-off* acontece de forma muito rápida, a ponto de não ocorrer muitas perdas de pacotes se comparada a quantidade de pacotes perdidos durante a execução de *hand-offs* utilizando computadores de maior porte, como os *notebooks*. Assim, como a quantidade de pacotes perdidos utilizando o N800 é pequena, principalmente porque este não tem capacidade de transmitir a mesma quantidade de dados que os *notebooks*, a pouca quantidade de dados perdidos não afeta os algoritmos de controle de congestionamento dos protocolos TCP e DCCP. A segunda hipótese é um complemento da primeira: como os dispositivos N800 são projetados para funcionar em ambientes móveis, o *driver* ou o tipo de *hardware* da placa de rede sem fio destes dispositivos possui algum mecanismo que otimiza internamente a execução do *hand-off*. Para obter conclusões mais precisas, faz-se necessário um estudo mais aprofundado nesta direção;

4. variar o tamanho do pacote TCP em transmissões de dados em conjunto com o protocolo UDP não obtem-se melhorias consideráveis, pois como visto nos resultados apresentados, não há aumento significativo da quantidade de dados transmitidos quando foram utilizados pacotes de 1424 *bytes*, ao invés de pacotes de 512 *bytes*. Já para os confrontos TCP × DCCP, considara-se que alterar o tamanho do pacote obtem-se um aumento na quantidade de dados transmitidos, portanto este procedimento é en-

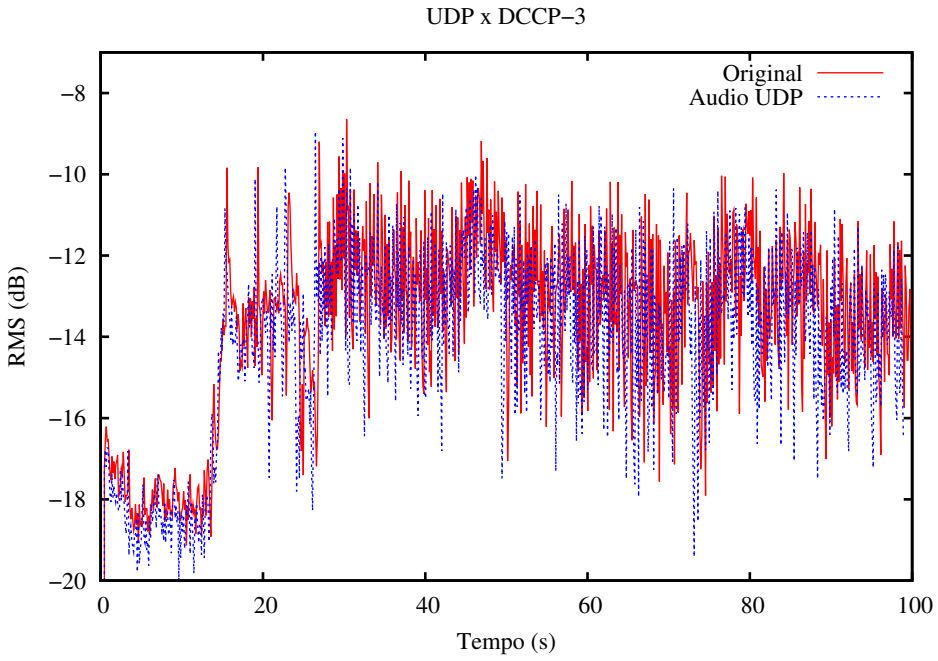


Figura 6.9: Fase 3: DCCP-3 $\times$ UDP, sendo o UDP enviando um arquivo de áudio.

corajado. Embora foi possível observar tal fato, faz-se necessário um estudo mais aprofundado neste ponto.

5. os atuais algoritmos para controle de congestionamento do protocolo DCCP apresentaram-se inferiores aos do TCP quando utilizados para competir com tráfego de dados UDP (exceto para o TCP Reno).

6. supondo que a maior parte do tráfego na Internet seja TCP Veno ou mais fortemente TCP Cubic, o protocolo UDP terá que ser eqüinime com relação ao TCP, uma vez que o TCP Cubic e o TCP Veno apresentaram-se de forma bastante agressiva em termos da utilização da largura de banda disponível. Até o final desta dissertação, não foram encontradas nenhuma referência que apresente propostas para controle de congestionamento em transmissões UDP, tampouco publicações oficiais de estudos comparativos do TCP Cubic e Veno contra o protocolo UDP.

7. é desencorajado a utilização do TCP $Reno$ em transmissões da dados, principalmente na Internet. Para transmissões TCP, considere utilizar o TCP $Cubic$ ao TCP $Veno$, uma vez que o primeira tem um propósito mais geral e funcionou muito bem nas redes sem fio, foco do TCP $Veno$.

8. o protocolo DCCP apresentou-se de forma bastante satisfatória quando utilizado em

conjunto com o protocolo TCP, compartilhando a utilização do canal de transmissão com outros protocolos, por exemplo, com o TCP.

9. é necessário analisar os algoritmos de controle de congestionamento do DCCP a fim de otimizar o desempenho destes algoritmos e/ou prover novos algoritmos de controle de congestionamento para DCCP equivalentes ao TCP Cubic e TCP Veno, preferencialmente. Já existem esforços por parte dos autores do TCP Cubic neste sentido. Embora seja um trabalho em estágio inicial e sem referências públicas até a fase final da escrita desta dissertação, está sendo desenvolvida uma versão experimental do TCP Cubic para o protocolo DCCP.

10. embora o desempenho do protocolo TCP tenha sido melhor que o DCCP em confrontos com o protocolo UDP, isto não que dizer que o protocolo DCCP não seja útil, principalmente quando considera-se que podem ocorrer melhorias nos algoritmos de controle de congestionamento do DCCP, a exemplo do que ocorreu com o próprio protocolo TCP, que até há pouco tempo dispunha apenas do TCP $Reno$ e de algumas outras variantes baseadas no TCP $Reno$.

Capítulo 7

Trabalhos Relacionados

"A idéia é tentar dar todas as informações que ajudem os outros a julgar o valor da sua contribuição; não apenas as informações que levem o julgamento a uma direção em particular." (Richard Feynman)

Neste trabalho as pesquisas foram realizadas considerando a perspectiva do protocolo DCCP. Sendo assim, neste capítulo são apresentadas as pesquisas cujo principal foco está no protocolo DCCP. Com relação a essas pesquisas, é de se esperar que existam trabalhos recentes acerca do protocolo DCCP, considerando que se trata de um protocolo novo e recentemente padronizado pela IETF.

Porém, ainda são poucas as pesquisas no contexto de análise de desempenho do protocolo DCCP em redes sem-fio, sobretudo quando se trata de avaliações deste protocolo na Internet. Até a finalização deste trabalho, nenhuma referência foi encontrada no contexto de avaliação experimental do protocolo DCCP na Internet. Portanto, este trabalho figura como sendo um dos primeiros, se não o primeiro, que se estuda o desempenho do protocolo DCCP quando utilizado para transmissão de dados multimídia na Internet.

Portanto, neste capítulo são apresentados os trabalhos que acreditamos ser as principais referências para as discussões elencadas ao longo desta pesquisa, considerando tanto aqueles que adotaram uma abordagem experimental quanto as baseadas em simulações. Neste contexto, foram selecionados três trabalhos: o primeiro apresenta os resultados do desempenho do protocolo DCCP para transmissão de vídeo, considerando uma abordagem de simulação; o segundo segue a mesma linha do primeiro, porém as discussões giram em torno da qualidade de áudio considerando um ambiente experimental; e por fim, o terceiro apresenta discussões acerca dos protocolos TCP, UDP e DCCP quando utilizados para transmitir fluxos de dados em redes *mesh* sem fio.

7.1 Estimativa da Qualidade de Vídeo em Fluxos DCCP sobre Redes Sem-Fio

No trabalho referenciado em [LMB+06], os autores descrevem um modelo para estimar a qualidade de vídeo para transmissão em redes sem-fio. Através da utilização do simulador NS2 [DAR], definiu-se parâmetros específicos do protocolo DCCP. Após este processo, criou-se um mecanismo de controle de congestionamento capaz de diminuir ou aumentar a qualidade do vídeo de acordo com o congestionamento na rede. A estratégia dos autores foi de adaptar o fluxo multimídia de acordo com a largura de banda disponível na rede.

De acordo com a largura de banda disponível, o pacote transmitido é codificado com uma determinada taxa de bit (*Variable Bit Rate* - VBR), tal como descrito na Seção 2.5.3. Isto permite diminuir tanto a taxa de perda de pacotes quanto melhorar o fluxo para atender às restrições de tempo das aplicações multimídia. O algoritmo de controle de congestionamento utilizado foi o DCCP/CCID-3 (veja Seção 2.5.3).

O gráfico ilustrado na Figura 7.1 apresenta a vazão conseguida por um fluxo de vídeo transmitido utilizando o protocolo DCCP, de acordo com os parâmetros definidos anteriormente. No gráfico são apresentados 3 estágios diferentes da relação vazão/largura de banda disponível. Nos primeiros $50\,s$ a vazão foi menor que a largura disponível na rede devido ao processo inicial de estimar a largura de banda disponível, segundo a Equação TFRC do CCID-3, definida na Seção 2.1.

Depois dessa fase inicial, a vazão melhorou e praticamente acompanhou a largura de banda disponível na rede. Em seguida acontece um comportamento similar aos primeiros $50\,s$. Segundo os autores, um fator que contribui para esta variação é o meio em que os dados estão sendo transmitidos, onde a taxa de perda de pacotes é geralmente maior se comparado às redes cabeadas.

Nas Figuras 7.2 e 7.3, são ilustrados dois gráficos onde é possível comparar o protocolo DCCP e o protocolo UDP na presença de congestionamento na rede. A duração da simulação foi de $150\,s$ de transmissão, considerando o envio de um arquivo de vídeo. Duas métricas foram avaliadas, a perda de pacotes e a qualidade do vídeo transmitido.

Na Figura 7.2, ilustra-se a perda de pacotes durante o tempo de simulação. Na transmissão utilizando o protocolo UDP houve perda de 2063 pacotes, ao passo que utilizando o protocolo DCCP foram perdidos apenas 180 pacotes. No caso do protocolo UDP, estes resultados apenas comprovam a ineficiência do protocolo na presença de congestionamento na rede, como discutido no Capítulo 6. Segundo as discussões sobre os resultados obtidos nesse trabalho, o momento em que mais se perdeu pacotes UDP aconteceu quando a largura de banda disponível na rede era menor do que a taxa de bits do pacote codificado, ou seja, quando o UDP transmitira mais pacotes que a capacidade de transmissão suportada pela rede.

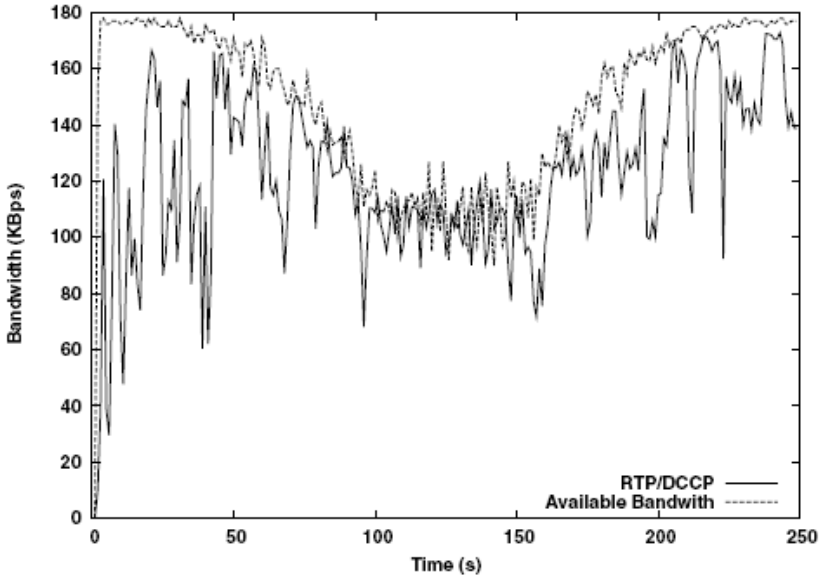


Figura 7.1: Vazão conseguida pelo protocolo DCCP relacionada à largura de banda real disponível. Figura extraída de [LMB+06].

Com relação à taxa de bits, trata-se de uma medida de qualidade de um arquivo comprimido por um codificador de áudio ou vídeo (Seção 3.2.1). Em geral a taxa de bits é apresentada em *Kbps* e representa o tamanho final desejado para o arquivo. Ou seja, 1 *Kbps* significa que a cada segundo o codificador tem 1000 *bits* do arquivo final para utilizar. Por exemplo, se um arquivo de vídeo tem 8 *s* e é comprimido a uma taxa de 1 *Kbps*, o arquivo final terá 8 *Kbits* ou aproximadamente 1 *Kbyte*. Assim, quanto maior for a taxa de bits, melhor será a qualidade do arquivo final, uma vez que o codificador terá mais espaço para comprimir o arquivo original, podendo adicionar "mais detalhes" do arquivo original ao arquivo gerado [KR06].

Nesse contexto, os autores utilizaram na simulação um codificador capaz de variar a taxa de bit do pacote de acordo com a largura de banda disponível na rede, ou seja, adaptar o fluxo de acordo com o nível de congestionamento da rede. Para medir a qualidade do vídeo foi utilizado o *Peak Signal-to-Noise Ratio* - PSNR, um padrão para estimar a qualidade de vídeo, similar ao RMS, discutido na Seção 5.5.

Na Figura 7.3, ilustra-se a evolução da qualidade do vídeo transmitido na rede de acordo com a largura de banda disponível. A técnica de adaptar a qualidade do fluxo transmitido na rede demonstrou ser uma estratégia bastante interessante para manter a relação entre a perda de pacotes e a qualidade do áudio/vídeo satisfatória. Os mecanismos de adaptação de fluxo apresentados pelos autores desse trabalho, serviu para definir as estratégias de adaptação de fluxo que estão sendo aplicadas aos projetos desenvolvidos no laboratório Embedded, principalmente no tocante ao projeto E-Phone[1].

[1]Projeto E-Phone: `http://garage.maemo.org/projects/ephone`

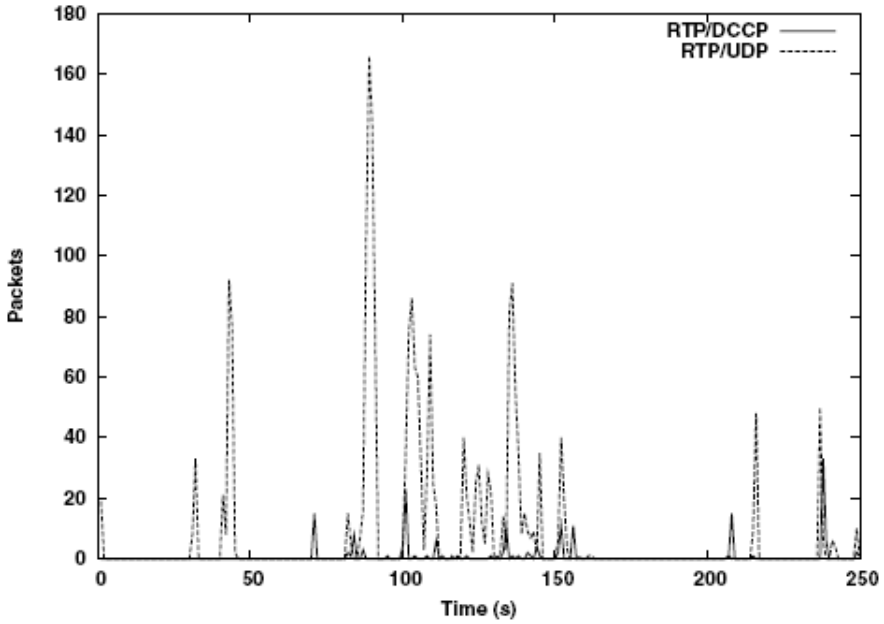


Figura 7.2: Perda de pacote em dois fluxo: um DCCP e outro UDP. Figura extraída da referência de [LMB+06].

Por fim, pelo gráfico de medida PSNR é possível concluir que a qualidade do vídeo é melhor quando se utiliza o fluxo adaptado e o protocolo DCCP. Isto ocorre porque a taxa de perda de pacotes do protocolo UDP é maior que a taxa de perda de pacotes do protocolo DCCP.

7.2 Uma Avaliação Experimental da Qualidade de Voz sobre o Protocolo DCCP

No trabalho referenciado em [BENB07], os autores discutem diversos temas também discutidos neste trabalho, tais como a ineficiência dos protocolos TCP e UDP quando são utilizados para transportar dados multimídia. Em seguida, eles apresentam uma introdução ao protocolo DCCP, com ênfase nos algoritmos de controle de congestionamento disponíveis para DCCP.

O objetivo do trabalho foi de avaliar a qualidade do áudio transmitido tendo como base as aplicações de voz sobre IP. A abordagem utilizado foi realizar experimentos de transmissão de fluxos de voz variando as condições de congestionamento da rede e utilizando diferentes codificadores de áudio disponíveis.

Com relação as condições de rede, variou-se o nível de perda (a taxas entre 0.01 % e 10 %) e o atraso (entre 0 ms e 400 ms) entre pacotes do fluxo de voz transmitido na rede. Uma das diferenças desse trabalho com relação às discussões realizadas nos Capítulos 4 e 6,

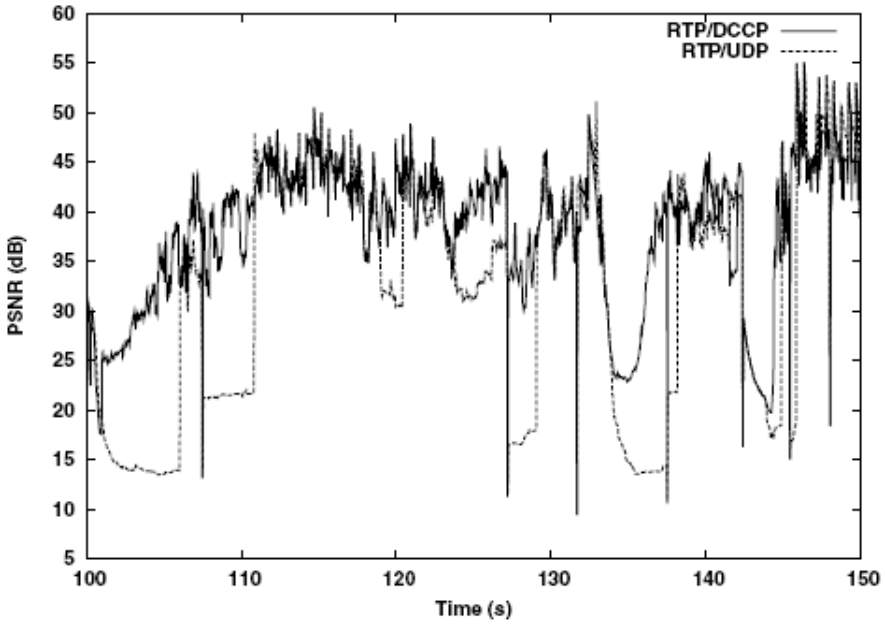


Figura 7.3: Qualidade (PSNR) do vídeo recebido pelo usuário. Figura extraída de [LMB+06].

é que os autores utilizaram uma ferramenta para emular as condições da rede[2], ao passo que nos experimentos realizados neste trabalho, não foram utilizados nenhum emulador de rede, sendo as condições da rede geradas considerando as topologias apresentadas no Capítulo 4.

Quanto a parte do áudio, os autores utilizaram dois codificadores de áudio, o $G.711$ [IT88] e o $G.729$ [IT96a], sendo os parâmetros para cada codificador apresentado na Tabela 7.1. A metodologia utilizada para a avaliação do áudio foi subjetiva e considerou-se a abordagem MOS (veja Seção 5.5).

Com relação aos protocolos de transporte, foram utilizados o TCP, o UDP e o DCCP. O algoritmo de controle de congestionamento do protocolo DCCP foi o CCID-3, considerando as suas variações, a original TFRC [FKP06] (veja Seção 2.5.3), a TFRC-SP [KHF07] (veja Seção 2.5.3) e a TFRC-SP com reinício rápido (FR - *Faster Restart*) [KFS07].

A ferramenta utilizada para transmitir os fluxos de voz foi o *ttcp*[3], que possui funcionalidades similares ao *IPerf* (veja Seção 4.3). Na execução dos experimentos, foi utilizado o sistema operacional FreeBSD, que possui a primeira implementação do protocolo DCCP.

Após a configuração da rede e definição dos parâmetros, são apresentados os resultados dos experimentos. Nas discussões acerca desses resultados, os autores apresentam que o protocolo DCCP obteve um nível baixo de desempenho se comparado ao desempenho dos protocolos TCP e UDP.

O gráfico ilustrado na Figura 7.4 apresenta o resultado para a transmissão do áudio durante $400\,s$ de experimento. Neste experimento foi utilizado o codificador de áudio $G.729$,

[2] A ferramenta utilizada foi a DummyNet Router

[3] `http://www.pcausa.com/Utilities/pcattcp.htm`

CoDec	Larg. Banda	Período Amostral	Tamanho do Quadro	Quadro/Pacote
G.711	64 *Kbits/s*	20 *ms*	160 *bytes*	1
G.729	8 *Kbits/s*	10 *ms*	10 *bytes*	2

Tabela 7.1: Parâmetros Utilizados nos Codificadores de Áudio e Características dos Codificadores

sendo cada ponto a média aritmética para cada ponto considerando 15 repetições. O eixo Y do gráfico representa a pontuação subjetiva MOS (*R-Score*) para a qualidade do áudio transmitido. Através do gráfico, conclui-se que o desempenho do protocolo DCCP foi pior que os protocolos TCP e UDP para qualquer uma das variantes do algoritmo CCID-3. Para esse gráfico foi considerada a variação do atraso e sem perda de pacotes.

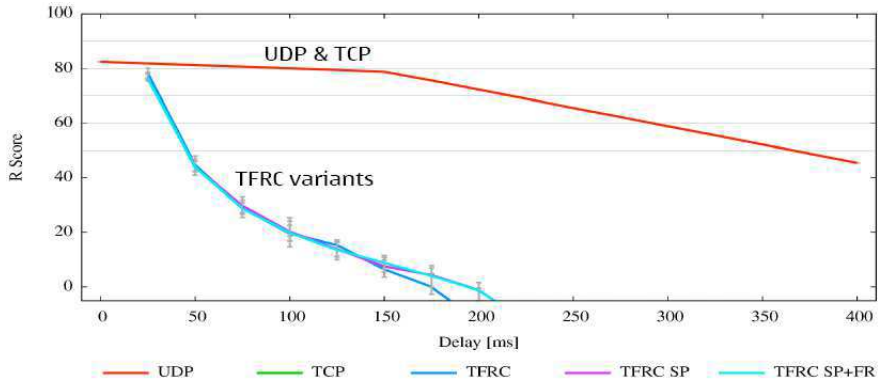


Figura 7.4: Qualidade do Áudio Baseado em pontuação subjetiva MOS para os Fluxos UDP, TCP e DCCP com variação do atraso e sem perda de pacotes.

Já no gráfico ilustrado na Figura 7.5, onde foi considerado 0.1 % de perda de pacotes, o fluxo TFRC e TFRC SP apresentaram comportamentos similares ao gráfico ilustrado na Figura 7.4. Porém, o fluxo TFRC SP+FR apresentou leve melhorias, mas ainda significativemente ruim se comparado aos fluxos dos protocolos TCP e UDP. Além disso, através deste gráfico é possível concluir que a qualidade do áudio transmitido pelo TCP diminuiu. Isso ocorreu devido as retransmissões realizadas pelo TCP, uma vez que para este experimentos foi considerado um ambiente com 0.1 % de perda de pacotes.

A principal contribuição desse trabalho pode ser explicado através dos gráficos ilustrados nas Figuras 7.6 e Figuras 7.7. Ao perceberem o baixo nível de desempenho do protocolo DCCP para todas as variantes do algoritmo CCID-3, os autores propuseram uma nova variante, baseada no TFRC SP+FR. Para essa variante deu-se o nome de TFRC SP+FR+MD.

Essa melhoria está embasada no erro de cálculo para definição da taxa de transmissão das variantes do TFRC. O cálculo da taxa de transmissão é baseado no tempo em que um transmissor fica sem transmitir dados na rede (*idle period*) e no primeiro evento de perda detectado pelo transmissor.

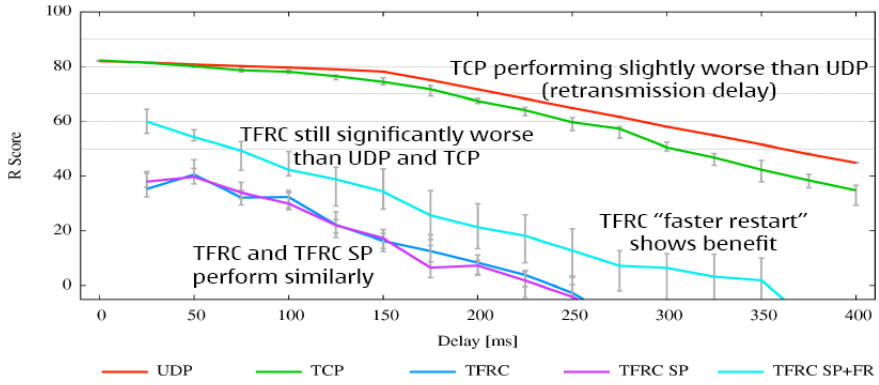


Figura 7.5: Qualidade do Áudio Baseado em pontuação subjetiva MOS para os Fluxos UDP, TCP e DCCP com variação do atraso e 0.1 % de perda de pacotes.

Na Equação TFRC (veja Seção 2.5.3, Equação 2.1), utilizada para calcular a taxa de transmissão, é realizada uma divisão do número de bytes recebidos desde o último relatório de recepção pelo tempo transcorrido que esse relatório foi enviado pelo receptor. Nas variantes de TFRC para DCCP, o valor desse período transcorrido inclui o período sem transmissão de dados, e portanto dependendo desse tempo o valor de taxa de transmissão será consideravelmente menor se comparado com o valor real para a taxa de transmissão desconsiderando o tempo sem transmissão de dados na rede.

Neste contexto, os gráficos ilustrados nas Figuras 7.6 e 7.7 apresentam o comportamento dos fluxos TCP, UDP e DCCP considerando todas as variantes, considerando a TFRC SP+FR+MD, que inclui as correções realizadas no algoritmo considerando-se a discussão feita no parágrafo anterior. Para esses resultados, percebe-se uma melhora significativa do algoritmo CCID-3 TFRC SP+FR+MD. Note no gráfico ilustrado na Figura 7.6 que quando houve variação no atraso e considerando a rede sem perda de pacotes, a versão melhorada do TFRC obteve um desempenho satisfatório se comparado às versões anteriores do algoritmo, aproximando-se do nível de desempenho dos protocolos TCP e UDP.

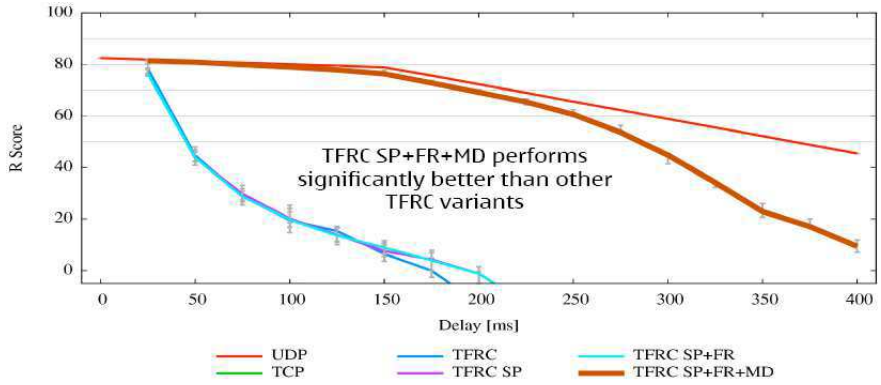


Figura 7.6: Qualidade do Áudio Baseado em pontuação subjetiva MOS para os Fluxos UDP, TCP e DCCP considerando variação do atraso e sem perda de pacotes.

Por fim, a partir do gráfico ilustrado na Figura 7.7, onde o atraso foi fixado em 50 s e

variando a perda de pacotes, o protocolo DCCP obteve um nível de desempenho identico ao protocolo UDP, ao passo que foi melhor que o protocolo TCP, uma vez que as perdas de pacotes resultou em retransmissão por parte do protocolo TCP.

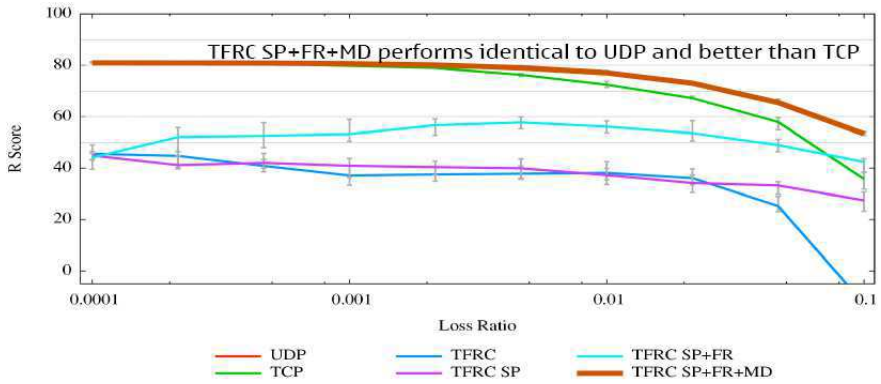


Figura 7.7: Qualidade do Áudio Baseado em pontuação subjetiva MOS para os Fluxos UDP, TCP e DCCP com atraso fixo de 50 *ms* e variando a perda de pacotes.

Embora esse trabalho seja recente[4] e apresente notórias contribuições, os resultados apresentados nesse trabalho são baseados em uma implementação do protocolo DCCP muito antiga, de 2005, e portanto antes da publicação oficial do RFC para DCCP [KHF06a]. Essa observação também é aplicada ao algoritmo de controle de congestionamento CCID-3. Os resultados desse trabalho serviram para corrigir a especificação do algoritmo CCID-3, cujo documento oficial foi publicado em Março de 2006.

Como mencionado no Capítulo 4, a versão do CCID-3 utilizada em nossos experimentos foi a disponível em Linux, a qual os problemas discutidos nesta seção já havíam sido resolvidos.

7.3 Desempenho do DCCP em Redes Mesh Sem-Fio

No trabalho referenciado em [NAT06], os autores apresentam resultados acerca de simulações realizadas para avaliar o desempenho do protocolo DCCP em redes *mesh* sem-fio. Uma rede mesh é formada por pontos de acesso sem-fio fixos e os dados são transmitidos de um ponto para outro utilizando um processo chamado de roteamento multi-saltos *multi-hop*. Nesse tipo de roteamento os dados são transmitidos através dos pontos de acesso fixos da rede até o sistema final de destino, sem utilizar nenhuma infra-estrutura de rede cabeada.

Os autores analisam o comportamento do protocolo DCCP quando é utilizado em aplicações multimídia e executado em redes *mesh* sem-fio. Um dos maiores desafios para os desenvolvedores de aplicações multimídia nesse tipo de rede está em dois fatores: as características intrínsecas do roteamento *multi-salto* e o retardo na transmissão do pacote à medida que ele é propagado através dos pontos de acesso até chegar no destino.

[4]Trabalho realizado no final de 2006 e publicado em 2007

O atraso causado por este tipo de rede pode ocasionar problemas às aplicações multimídia que, como foi discutido em várias partes deste documento, possuem restrições de tempo quanto à entrega de pacotes de dados até o sistema final de destino. Foi nesse contexto que os autores estudaram o comportamento do protocolo DCCP.

A topologia da rede *mesh* sem-fio utilizada na simulação é ilustrada na Figura 7.8, onde os dados foram transmitidos de acordo com o processo de múltiplos saltos descritos acima e os círculos representam os pontos de acesso.

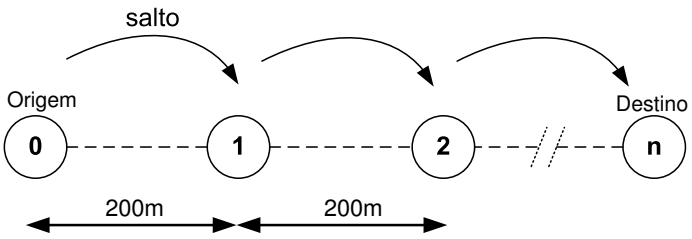


Figura 7.8: Topologia da rede *mesh* utilizada para simulação. Figura adaptada de [NAT06].

O gráfico ilustrado na Figura 7.9 apresenta a vazão alcançada por dois fluxos DCCP e um TCP à medida que os pacotes de dados são roteados na rede pelo processo de *multi-hop*. O fluxo TCP transferiu via FTP um arquivo de 1 *GB* através do ponto de acesso 0 até o 7. O fluxo DCCP-CBR transmitiu dados em uma taxa constante do ponto de acesso 1 até o 7 e o fluxo DCCP-FTP transferiu o mesmo arquivo transmitido pelo protocolo TCP, do ponto de acesso 0 para o ponto de acesso 7.

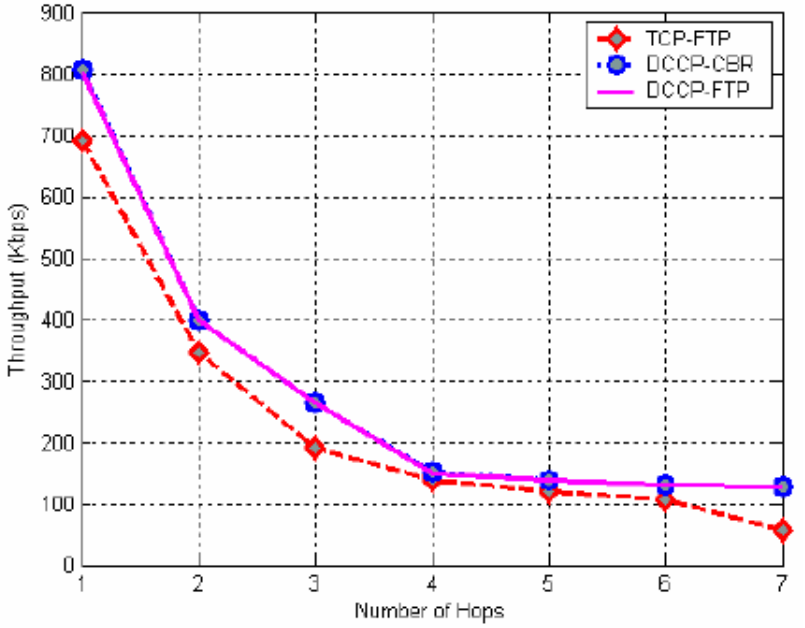
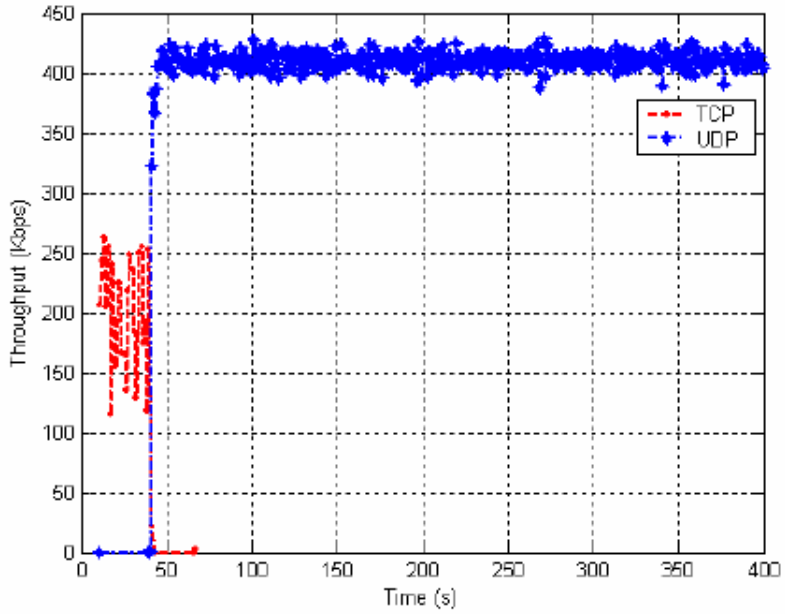
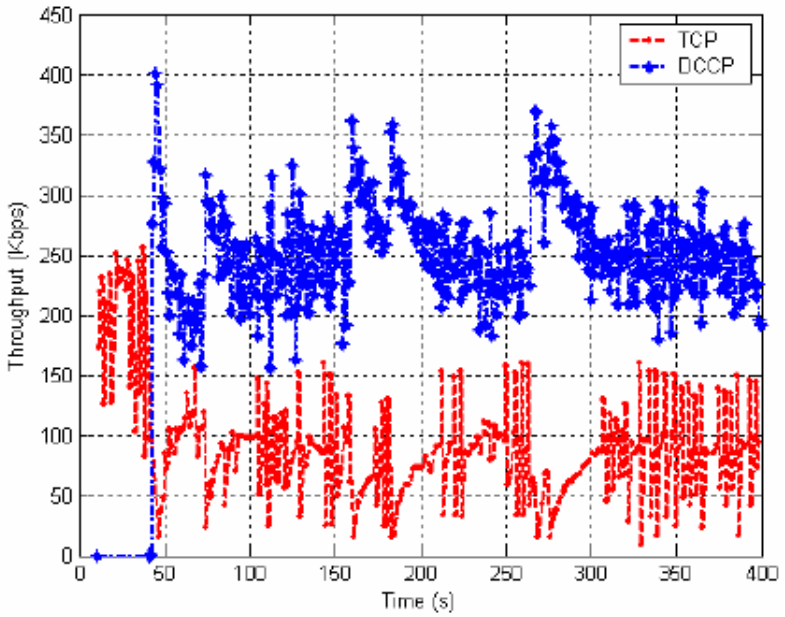


Figura 7.9: Média de vazão para fluxos DCCP simulados separadamente. Figura extraída de [NAT06].



(a) TCP × UDP. Após 50s do início da simulação, o fluxo UDP ocupa toda a largura de banda disponível na rede.



(b) TCP × DCCP. Ambos os protocolos conseguem transmitir dados na rede de forma justa.

Figura 7.10: Comparação entre os fluxos TCP, UDP e DCCP quanto ao conceito de *fairness*. Figura extraída de [NAT06].

É possível concluir pelo gráfico que a vazão de transmissão das conexões DCCP degrada com o aumento do número de saltos realizados na rede. O padrão de comportamento do protocolo TCP é similar ao do protocolo DCCP. Ambos os fluxos DCCP iniciaram com uma vazão de 800 *Kbps*, porém após 7 saltos a vazão não ultrapassava 150 *Kbits/s*.

Um outro estudo foi avaliar como os fluxos de conexão DCCP afetam outros fluxos na rede e virse-versa. Os gráficos ilustrado na Figura 7.10 mostram a avaliação da vazão pelo tempo durante os primeiros 400 *s* de simulação.

No gráfico da Figura 7.10(a) ilustra que o fluxo UDP pode degradar consideravelmente um fluxo TCP. É possível perceber que 50 *s* após o fluxo UDP ser iniciado, o desempenho do fluxo TCP começou a ser degradado e a vazão decaiu rapidamente. Já no gráfico da Figura 7.10(b) esta mesma simulação foi realizada entre um fluxo TCP e um DCCP. O fluxo DCCP em convívio com um fluxo TCP apresenta-se de forma justa. É possível concluir que a vazão de ambos os fluxos foi satisfatória. Através destes gráficos, as semelhanças são evidentes quando são comparadas as discussões apresentadas neste documento, em especial no Capítulo 6.

Neste capítulo foram apresentados os trabalhos relacionados às discussões elencadas nesta dissertação. Outros trabalhos porém, oferecem diversas discussões acerca do protocolo DCCP, mas que não estão intimamente relacionados às nossas pesquisas. Para maiores informações sobre esses trabalhos, consulte as referências [TKI+05], [LCLZ06], [HLRX07], [XLZ05] e [Niv05], para trabalhos no âmbito de transmissão de dados multimídia em tempo real e qualidade de serviço; [CSMP05], [CSMP06] e [HBMP07], para trabalhos de avaliação de desempenho de algoritmos baseados em TFRC; e [KL05] e [Cam05], para trabalhos que abordam propostas de implementação das funcionalidades do protocolo DCCP.

Capítulo 8

Considerações Finais

"Isso não é o fim. Isto não é sequer o começo do fim. Isto é, talvez,
o fim do começo." (Winston Churchill)

Neste trabalho, apresentou-se uma avaliação experimental do protocolo DCCP sobre redes sem-fio e na Internet. Foram discutidos os principais problemas que acontecem quando utiliza-se os protocolos TCP e UDP para transmitir dados multimídia sobre essas redes e em seguida, apresentou-se os objetivos desta dissertação. Após a parte introdutória deste documento, apresentou-se uma visão geral da camada de transporte TCP/IP, focando nos conceitos relacionados ao controle de congestionamento em redes IP e oferecendo detalhes sobre o funcionamento do protocolo DCCP. Ainda nesta etapa, apresentou-se brevemente os protocolos UDP e TCP, sendo que para estes, discutiu-se sobre os algoritmos de controle de congestionamento TCP $Reno$, $Cubic$ e $Veno$, os quais foram utilizados na execução dos experimentos realizados neste trabalho.

Em seguida, apresentou-se discussões acerca das aplicações multimídia quando executadas sobre as redes sem fio e em redes híbridas, como a Internet. No capítulo seguinte, apresentou-se os procedimentos adotados para construção das redes onde os experimentos foram executados. Discutiu-se as topologias de rede adotadas e os objetivos por trás de cada uma delas, os equipamentos utilizados e principalmente as ferramentas desenvolvidas para dar suporte a execução dos experimentos e para processar os dados coletados. Neste ponto, apresentou-se também a estratégia adotada para a execução dos experimentos, os quais foram divididos em três fases: experimentos executados utilizando dispositivos com recursos limitados de memória e processador; experimentos utilizando computadores *desktops* com maior poder de processamento; e experimentos transmitindo dados multimídia através da Internet.

Em seguida, foram apresentadas discussões sobre os métodos utilizados para executar os experimentos, tais como os cálculos para obter o intervalo de confiança para cada uma das métricas estudadas e a quantidade de repetições necessárias para se obter 95 % de certeza em

todos os valores das métricas estudadas.

Nos capítulos seguintes, foram apresentados os resultados e discussões, assim como os trabalhos relacionados. Com relação aos resultados apresentados e as conclusões elencadas no Capítulo 6, Seção 6.4, pode-se destacar as seguintes considerações finais:

1. embora o protocolo UDP não tenha congestionada a rede quando foram utilizados os N800, a quantidade de dados perdidos foi muito maior se comparada a quantidade de dados perdidos pelo protocolo DCCP. Portanto, é desencorajado o uso do protocolo UDP para transmissões de dados multimídia, principalmente devido a disponibilização de novos algoritmos de controle de congestionamento para o protocolo TCP, tais como o TCP $Cubic$ e o TCP $Veno$.

2. considerando as topologias de rede utilizadas para executar os experimentos, aumentar o tamanho do pacote em transmissões TCP $\times$ UDP não obtem-se resultados consideráveis em termos da carga de dados efetivamente transmitida pelo protocolo TCP. Por outro lado, em confrontos TCP $\times$ DCCP foi evidente que o protocolo TCP conseguiu transmitir mais dados à medida que aumentou-se o tamanho do pacote de 512 *bytes* para 1424 *bytes*.

3. pelo item anterior e considerando os experimentos executados na Internet, quando o TCP foi utilizado para transmitir dados multimídia ao mesmo tempo em que foram transmitidos fluxos UDP, o protocolo TCP conseguiu transmitir apenas 30 % do arquivo de áudio transmitido na Internet. Isto não aconteceu nos confrontos TCP $\times$ DCCP. Embora foi possível escutar os 30 *s* de áudio transmitido pelo protocolo TCP depois que ele foi salvo em arquivo, caso o áudio estive sendo escutado em tempo real por uma pessoa, esta pessoa perceberia interrupções no áudio sendo transmitido, pois como discutido nos Capítulos 1 e 2, quando um pacote é perdido em uma transmissão TCP, ele é retransmitido, o que causa um atraso no fluxo contínuo dos dados, e como conseguinte isso gera um impacto na percepção auditiva do conteúdo multimídia sendo transmitido.

4. recomenda-se pesquisas para analisar possíveis melhorias nos atuais algoritmos de controle de congestionamento do protocolo DCCP, principalmente estudos que analisem o desempenho do protocolo DCCP se for utilizado um algoritmo para controle de congestionado baseado no TCP $Cubic$.

5. considerando o nível satisfatório do desempenho do TCP $Cubic$ quando em confronto com o protocolo UDP e que o $Cubic$ está sendo utilizado em milhares de servidores no mundo, pois desde 2005 é o algoritmo padrão para controle de congestionamento em Linux, isto sugere que o protocolo UDP torne-se obsoleto em transmissões de dados

multimídia, sendo utilizado apenas por aplicações com pouca troca de dados, como os servidores DNS, e em aplicações que transmitem tráfego *multicast*.

Portanto, recomenda-se fortemente a utilização dos protocolos de transporte TCP e DCCP, principalmente em redes contendo uma grande demanda em aplicações orientadas a dados e ao mesmo tempo uma grande demanda em aplicações com requisitos de tempo real. Nestes casos, consideração particularmente importanto o uso dos algoritmos de controle de congestionamento TCP $Cubic$ e DCCP $CCID-3$, pelo menos até enquanto não se disponibiliza uma versão do TCP $Cubic$ para o protocolo DCCP.

8.1 Trabalhos em Andamento: Habilitando DCCP em Plataformas Abertas

Nesta seção são apresentadas algumas contribuições e trabalhos em andamento no contexto de desenvolvimento de aplicações utilizando o protocolo DCCP.

Projeto E-Phone

O Projeto E-Phone (Embedded-Phone) contempla o desenvolvimento de uma aplicação de voz sobre IP para dispositivos móveis baseada no protocolo DCCP. O alvo são os dispositivos disponível na plataforma maemo, tais como aqueles utilizados na Fase 1 dos experimentos realizados.

Um dos principais objetivos do projeto E-Phone é guiar o desenvolvimento do protocolo DCCP com relação as funcionalidades implementadas no núcleo do Linux. Por exemplo, quando o componente de DCCP para Gstreamer estava sendo desenvolvido, constatou-se que não era possível obter, através da API de *socket* disponível no núcleo do Linux, a quantidade de dados disponível para leitura em uma conexão DCCP. O reflexo disso no núcleo do Linux foi a implementação de uma função apresentada no trecho de código 8.1. Através desta função, é possível obter a quantidade de dados disponível para leitura em um *socket* DCCP utilizando-se a opção *FIONREAD*.

Codigo 8.1: Implementação do FIONREAD para sockets DCCP

```
int dccp_ioctl(struct sock *sk, int cmd, unsigned long arg)
{
        int rc = -ENOTCONN;

        lock_sock(sk);

        if (sk->sk_state == DCCP_LISTEN)
                goto out;
```

```
        switch (cmd) {
        case FIONREAD: {
                struct sk_buff *skb;
                unsigned long amount = 0;

                skb = skb_peek(&sk->sk_receive_queue);
                if (skb != NULL) {
                        amount = skb->len;
                }
                rc = put_user(amount, (int __user *)arg);
        }
                break;
        default:
                rc = -ENOIOCTLCMD;
                break;
        }
out:
        release_sock(sk);
        return rc;
}
EXPORT_SYMBOL_GPL(dccp_ioctl);
```

Um outro exemplo de contribuição foi a descoberta de um estado desnecessário utilizado na máquina de estados do CCID-3/CCID-4 implementada no núclo do Linux. Essa máquina de estados é utilizada para controlar o estado desses algoritmos durante um transmissão DCCP. Na Figura 8.1, ilustra-se o diagrama de estados original para um transmissor DCCP que utiliza o algoritmo CCID-4. Este diagrama de estados mapeia as mudanças de estado entre as chamadas de funções da implementação do CCID-3/CCID-4 no núcleo do Linux. As mudanças de estado são representadas pelos nomes das funções definidas na implementação e as elipses representam os estados.

Na implementação do DCCP no núcleo do Linux, as ações realizadas em cada estado são de fato ativadas e controladas pelo núcleo da implementação do DCCP, e não pelos CCIDs. Para estes, apenas é delegada a tarefa de determinar a taxa de transmissão para um certo instante da conexão. Assim, o estado TTX_STATE_TERM é desnecessário, porém vinha sendo utilizado na implementação desses algoritmos no núcleo do Linux.

Através desta constatação, as devidas mudanças foram realizadas na implementação dos algoritmos CCID-3 e CCID-4. Estas mudanças entraram em vigor na última versão disponível do DCCP no núcleo do Linux. A primeira vista, pode-se imaginar que o impacto disso não é grande para um servidor DCCP com poucas conexões. A questão é que considerando

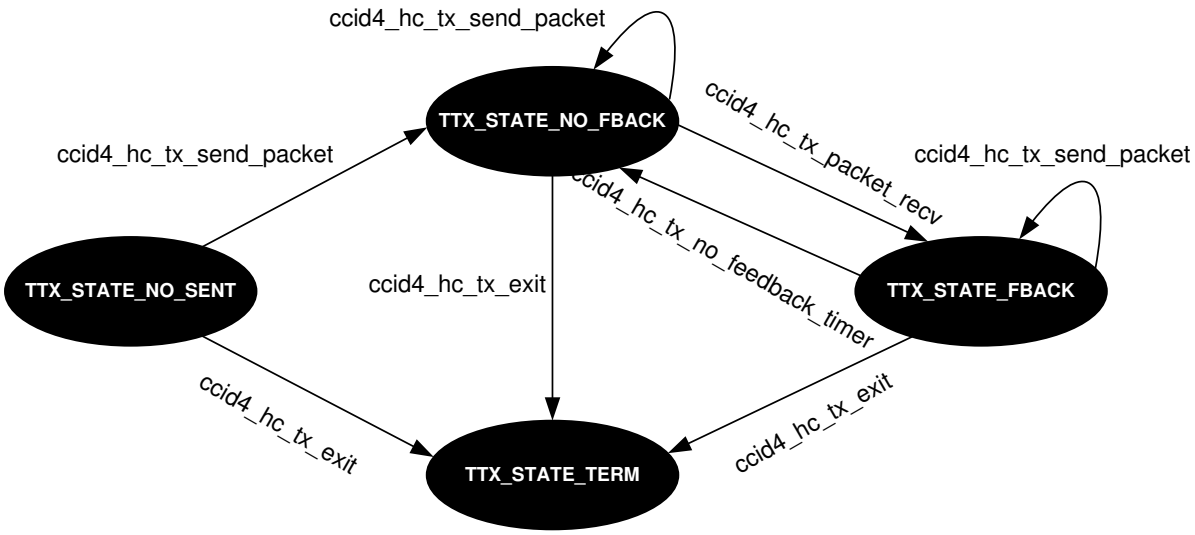


Figura 8.1: Diagrama de Estados dos algoritmos CCID-3/CCID-4 com o estado desnecessário TTX_STATE_TERM.

um servidor DCCP com 4 placas de rede 10 *gigabit/ethernet*, servindo uma rede metropolitana e com 1 milhão de conexões abertas, mudar para um estado inútil pode causar um impacto no desempenho do protocolo DCCP e também na utilização dos recursos de processamento do servidor. É nesta linha que outras tarefas estão em andamento, principalmente no desenvolvimento do algoritmo CCID-4.

Com relação ao E-Phone como aplicação, o projeto faz uso dos projetos Telepathy[1] e do projeto Farsight[2]. Ambos os projetos utilizam como base o arcabouço GStreamer, discutido na Seção 4.3 e apresentado com mais detalhes no Apêndice B.

O Telepathy é uma biblioteca que permite o desenvolvimento de solução para aplicações que lidam com comunicação em tempo real, como as aplicações de voz sobre IP. Já o Farsight é uma biblioteca que permite uso facilitado de protocolos de transmissão de dados multimídia. O componente de DCCP para GStreamer está sendo acoplado a este projeto.

Além dos protocolos UDP e DCCP, o projeto Farsight também contempla a implementação de outros protocolos de aplicação, como o RTP [SW99; DPC+97] e o SIP [WAR04; RSC02], dois importantes protocolos utilizados para transmitir dados multimídia em redes IP. O RTP é útil para adicionar informações de tempo nos pacotes de dados multimídia, ao passo que o SIP é utilizado para sinalização de controle em uma chamada telefônica em redes IP, como por exemplo sinalizar o tom de discagem, sinal de ocupado etc.

Além dos recursos de voz sobre IP utilizando o protocolo DCCP, na aplicação E-Phone está sendo desenvolvido um mecanismo para realizar *hand-off*. Trata-se de um *daemon* capaz de detectar a presença de pontos de acesso sem-fio no ambiente e ordenar estes pontos de acordo com a intensidade do sinal da conexão sem fio entre o dispositivo móvel e cada ponto

[1] http://telepathy.freedesktop.org/
[2] http://farsight.freedesktop.org

de acesso. Esta informação será útil para decidir qual é o melhor momento de iniciar o processo de *hand-off* e qual é o melhor ponto de acesso disponível para um certo instante. Na Figura 8.2 são ilustradas algumas telas da aplicação do E-Phone.

(a) Menu Principal (b) Páginas Amarelas (c) Teclado Numérico (d) Chamada Telefônica

Figura 8.2: Algumas tela da aplicação VoIP E-Phone que está sendo implementada para o dispostivo Nokia N800 e N810 Internet Tablet. (a) tela de menu do sistema, também é possível realizar *hand-off* manual. (b) tela para consultar e buscar usuários conectados. (c) teclado numérico para realizar uma chamada. (d) durante uma chamada telefônica VoIP.

DCCP para Maemo

O Projeto DCCP para maemo contempla o desenvolvimento e manutenção do protocolo DCCP para o núcleo do Linux. Como o Linux é um sistema operacional que pode ser utilizado em diversas plataformas, as contribuições em nível de implementação que estão sendo feitas também contempla a plataforma maemo. As atividades nesse contexto são manter as últimas funcionalidades do protocolo DCCP disponíveis no núcleo do Linux, prover melhorias e correções de erros na implementação do protocolo DCCP e manter o código do protocolo DCCP para o núcleo do Linux disponível para maemo.

Além disso, nesse projeto também é contemplado o desenvolvimento do algoritmo de controle de congestionamento CCID-4 (Seção 2.5.3). Para utilizar o núcleo do Linux com suporte a DCCP, acesse o endereço `https://garage.maemo.org/projects/ephone`. Para utilizar o protocolo DCCP com suporte ao CCID-4 acesse `git://eden-feed.erg.abdn.ac.uk/dccp_exp` e selecione o *branch*[3] *ccid4*.

Com relação a implementação do algoritmo de controle de congestionamento CCID-4 para DCCP, contribuições estão sendo feitas. As duas principais contribuções são a participação na es-

[3]Em sistemas de controle de versão, por exemplo o GIT e o SVN, um *branch* é uma linha de desenvolvimento que ocorre em paralelo a linha de desenvolvimento principal, nesse caso chamada de *trunk*.

crita da RFC do CCID-4 [KHF07] – consulte o preâmbulo e a seção 14 desse documento para mais informações das contribuições realizadas – e a implementação do CCID-4 no núcleo do Linux.

DCCP para Java

O projeto DCCP para Java (jDCCP) foi criado para permitir o uso do protocolo DCCP através da linguagem Java. A biblioteca jDCCP está sendo desenvolvida utilizando as linguagens Java através da API Java/JNI[4] (*Java Native Interface*) e a linguagem C. A estrátégia é implementar as funções de *socket* DCCP na linguagem C, para iniciar uma conexão, enviar e receber dados etc., e invocar essas funções através de uma classe escrita em Java utilizando JNI. Para utilizar a biblioteca de DCCP para Java, acesse o endereço `http://code.google.com/p/jdccp/`.

8.2 Trabalhos Futuros

Ao longo deste trabalho, procurou-se contemplar diversos pontos em abertos relacionados ao protocolo DCCP, seja no âmbito de avaliação do desempenho do protocolo DCCP em redes sem fio e na Internet ou em pontos que requereram esforços de implementação tanto no núcleo do Linux quanto na implementação do E-Phone.

Contudo, ainda existem diversos tópicos que podem ser investigados, os quais podem ser considerados como trabalhos futuros:

- avaliações do uso do protocolo DCCP em redes $IPv6$ [KR06]. Os estudos realizados nesta dissertação contemplou o uso do protocolo $IPv4$.

- como há interesse em disponibilização de aplicações multimídia em dispositivos com recursos limitados de processador e memória, também considera-se importante avaliar se utilizar o protocolo DCCP aumenta consideravelmente ou não o consumo de energia desses dispositivos, uma vez que há um aumento do uso de processamento e memória.

- realizar pesquisas utilizando o mecanismo de escolha tardia de dados (veja Seção 2.5.2) disponível no DCCP. Uma possível estratégia que pode ser adotada nesse trabalho é desenvolver uma aplicação que utilize os recursos de escolha tardia de dados e quando esta detectar congestionamento, priorizar apenas os pacotes que transportam informações de áudio. Nesse trabalho, deve-se ter como principal objetivo avaliar se a aplicação pode, de fato, beneficiar-se com esse mecanismo.

- avaliar o uso do TCP Cubic como um possível algoritmo de controle de congestionamento para DCCP. As pesquisas realizadas pela IETF no contexto dos protocolos TCP e DCCP, especificamente para o CCID-3, estão focadas nos algoritmos baseados em TFRC. Os criadores do TCP Cubic estão trabalhando no TEAR (*TCP Emulation At Receivers*), uma solução concorrente para os algoritmos baseados em TFRC. O autor do TEAR foi contactado e informou que um artigo sobre o TEAR está sendo confeccionado. Como não foram

[4] http://java.sun.com/j2se/1.5.0/docs/guide/jni/

encontradas publicações oficiais sobre o TEAR, considere a leitura do documento `http://www.princeton.edu/~yyi/pubs/tear.pdf` como um ponto de partida. Este documento foi escrito pelo autor.

- avaliações do uso do protocolo DCCP em conjunto com o protocolo SCTP (*Stream Control Transmission Protocol*) [SXM+00; Jun03]. O protocolo SCTP também é padronizado pela IETF e tem como principal objetivo transmitir pacotes de dados sem erro e sem duplicação; suporte a fragmentação baseado no tamanho máximo do segmento (ver Capítulo 2); e suporte a múltiplos fluxos em uma mesma conexão capacidade para *multihoming*.

- avaliações do uso do protocolo DCCP em redes padrão 802.16 WiMax - *(Worldwide Interoperability for Microwave Access*) [Hos07]. O padrão WiMax tem como objetivo promover acesso a Internet em redes metropolitanas. Este padrão permite altas taxas de transmissão de dados.

8.3 Algo para refletir: em meio aos apagões no Brasil, enfrentaremos algum na Internet?

Pode-se dizer que é possível. O que dizer se pensarmos no tráfego de dados na Internet de hoje e as conseqüências da grande demanda de acesso aos serviços disponíveis na Internet em um futuro próximo? Além das estatísticas apresentadas na introdução deste documento com relação ao crescimento do acesso à Internet no Brasil, uma pesquisa realizada em Novembro de 2007 apontou que a Internet pode ficar sem capacidade de suportar a grande quantidade de acesso em 2010 [Ser07]. Segundo esse estudo, o acesso a Internet em 2007 gerou cerca de 161 *exabytes* de dados, equivalente a 50 mil anos de vídeos com qualidade de DVD. Outro resultado dessa pesquisa é que o uso corporativo e doméstico da Internet pode levar a um esgotamento da capacidade atual da Internet, caso não sejam feitos investimentos que ultrapassam 137 bilhões de dólares.

Com base na analogia sobre o trânsito na cidade de Maceió, utilizada para explicar o conceito de escolha tardia de dados no DCCP no capítulo introdutório desta dissertação, considere o seguinte fato real. Segundo dados do Departamento Nacional de Trânsito [Res07], desde 2004 a frota de veículos circulando no país aumentou em média 1,5 milhão por ano. Os estados de São Paulo, Minas Gerais e Rio de Janeiro são os que lideram com maior número de automóveis. Apenas entre Janeiro e Novembro de 2007, as vendas de carros novos passaram de 2,1 milhões, um crescimento de 29 % em relação aos 11 primeiros meses de 2006. Caso o crescimento continue nesse ritmo, os especialistas em tráfego de trânsito prevêem apagões no trânsito das principais capitais do estados brasileiros. Em cinco anos, São Paulo pode chegar a uma paralização, aumentando ainda mais os rodízios de veículos na cidade. Em 10 anos, o trânsito do Rio de Janeiro ameaça se igualar ao de São Paulo hoje, e Belo Horizonte pode chegar à mesma situação em 15 anos.

Não está se falando mais em analogias, está se falando agora em fatos reais e neste caso existem semelhanças entre esses dois cenários. De um lado, o trânsito de veículos nas capitais brasileiras e do outro o tráfego de dados nas grandes redes, em especial na Internet. Com relação ao trânsito nas capitais, o objeto de estudo é a quantidade de veículos que trafegam nas grandes cidades, ao passo

que para as redes de computadores é o grande volume de carros, ou melhor, de dados transmitidos nessas redes. Estamos presenciando um aumento significativo da quantidade de pessoas que acessam a Internet, da mesma forma que aumenta a quantidade de veículos nas grandes cidades brasileiras, e também nas principais capitais do mundo. Por exemplo, a prefeitura da cidade de Nova Iorque cobrará pedágio de carros para quem entrar na ilha de *Manhattam* [Onl08]. Um estudo da prefeitura da referida cidade apontou que nos próximos 25 anos Nova Iorque receberá mais 1 de novos moradores. Para este caso, uma outra semelhança: na Internet tem-se o aumento na demanda pelos serviços providos, e no trânsito a necessidade que as pessoas têm de se locomoverem nas cidades, onde a maioria utiliza seus automóveis, que diga-se de passagem, já estão sendo fabricados com dispositivos que permitem o acesso à Internet através de rede sem fio, aumentando ainda mais o acesso à Internet.

Portanto, essas semelhanças nos leva a acreditar que, estudar novas soluções para controle de congestionamento na Internet pode fornecer mecanismos que ajudem a sanar os problemas gerados pela evolução dos serviços de rede e pelo aumento da demanda do acesso à Internet. Em meio aos problemas enfrentados pelo nosso país de tantos "apagões", finalizamos com a certeza que a mais importante contribuição deste trabalho são os resultados e discussões apresentados, considerando todo os esforços que tem sido feito para o desenvolvimento dos trabalhos ainda em andamento, os quais certamente contribuirão para evitar um futuro e possível apagão na Internet.

Bibliografia

[AFP02] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window, 10 2002. Último acesso, Abril de 2008.

[APS99] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, 4 1999. `http://www.ietf.org/rfc/rfc2581.txt`. Último acesso, Abril de 2008.

[Aut08] Internet Assigned Numbers Authority. Assigned Internet Protocol Numbers, 2008. `http://www.iana.org/assignments/protocol-numbers`. Último acesso, Abril de 2008.

[BA03] C. Bouras and A. Gkam As. Multimedia Transmission with Adaptive QoS based on Real-time Protocols. In *International Journal of Communications Systems, Wiley InterScience*, volume 16, pages 225 – 248, 2003.

[Bal98] Hari Balakrishnan. Challenges to Reliable Data Transport Over Heterogeneous Wireless Networks. 1998. Univ. California, Berkeley, CA.

[Bap02] Joe Baptista. DDoS Attack: What The Media Did Not Tell You, 4 2002. http://www.cynikal.net/users/baptista/papers/rootserverattack_Nov_20_2002 .pdf. Último acesso, Abril de 2008.

[BENB07] Vlad Balan, Lars Eggert, Saverio Niccolini, and Marcus Brunner. An Experimental Evaluation of Voice Quality over the Datagram Congestion Control Protocol. In *INFOCOM 07*, volume 1, pages 455–463, 5 2007.

[BMN+04] M. Baugher, D. McGrew, M. Naslund, E. Carrara, and K. Norrman. Domain Names - Implementation and Specification, 3 2004. Último acesso, Abril de 2008.

[BOP94] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. volume 1, pages 24–35, 10 1994.

[BPSK97] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links. volume 5, pages 756–769, 8 1997.

[Bra89] R. Braden. Requirements for Internet Hosts - Communication Layers, 10 1989. Último acesso, Abril de 2008.

[BV05] Gerassimos Barlas and Bharadwaj Veeravalli. Optimized Distributed Delivery of Continuous-Media Documents over Unreliable Communication Links. volume 16, pages 982–994, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[Cam05] Marie Camier. Development of a DCCP-Based Adaptive VoIP Client, 8 2005. Dissertação de Mestrado. Master of Engineering in Telecommunications Engeneering. Dublin City University. School of Electronic Engeneering.

[CHM+05] Tyson Condie, Joseph M. Hellerstein, Petros Maniatis, Sean Rhea, and Timothy Roscoe. Finally, a Use for Componentized Transport Protocols. In *Proceedings of the Fourth Workshop on Hot Topics in Networks*, volume 1, pages 201–207, 11 2005.

[Com04] Douglas E. Comer. *Interligação em Redes com TCP/IP: Princípios, Protocolos e Arquitetura*. ELSEVIER, 2 edition, 9 2004.

[Cor05] Steven Corbet. Linux gets DCCP, 2 2005. `http://lwn.net/Articles/149756/`. Último acesso, Abril de 2008.

[CSMP05] Lin Cai, Xuemin Shen, Jon W. Mark, and Jianping Pan. Performance Analysis of TCP-Friendly AIMD Algorithms for Multimedia Applications. In *IEEE Transactions Multimedia, Issue 2*, volume 7, pages 339–355, 4 2005.

[CSMP06] Lin Cai, Xuemin Shen, Jon W. Mark, and Jianping Pan. QoS Support in Wireless/Wired Networks Using the TCP-Friendly AIMD Protocol. In *IEEE Transactions on Wireless Communication, Issue 2*, volume 5, pages 469–480, 7 2006.

[CY88] D. Comer and R. Yavatkar. A Congestion Filtering Scheme for Packet Switched Networks. volume 1, West Lafayette, IN 47907, 10 1988. Purdue University,.

[DAR] DARPA Group. Network Simulator 2. `http://www.isi.edu/nsnam/ns/`. Último acesso, Abril de 2008.

[dCLF06] Arlindo F. da Conceição, Jin Li, and Dinei A. Florêncio. Transmissão de Voz sobre Redes IEEE 802.11: Um Levantamento dos Principais Problemas e Restrições. In *Simpósio Brasileiro de Sistemas Multimídia e Web*, volume 1, pages 422–432, 2006.

[dF04] Nelson Luís Saldanha da Fonseca. Investigação da Efetividade de Explicit Congestion Notification, 10 2004. `http://www.gta.ufrj.br/quaresma/atividades/13.htm`. Último acesso, Abril de 2008.

[DPC+97] J. Du, D. Putzolu, L. Cline, D. Newell, M. Clark, and D. Ryan. An Extensible Framework for RTP-based Multimedia Applications. In *Proceedings do 7th International Workshop on Network and Operating System Support for Digital Audio and Video*, volume 1, pages 53–60, 1997.

[DYP07] A.G Dan Ye; Parlos. Predictive Path Switching Control for Improving the Quality of Service in Real-Time Applications. In *IEEE Journal of Selected Topics in Signal Processing*, volume 1, pages 308–318, 8 2007.

[EFE07] EFE. China Já Tem 144 Milhões de Internautas e Prevê 200 Milhões em 2010, 5 2007. `http://tecnologia.terra.com.br/interna/0,,OI1625324-EI4802,00.html`. Último acesso, Abril de 2008.

[Emb08] Embedded. Capacitação para Desenvolvimento de Software para Aplicações Móveis. http://embedded.ufcg.edu.br/, 4 2008. http://embedded.ufcg.edu.br/. Último acesso em Abril de 2008.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP, 6 1999. `http://www.ietf.org/rfc/rfc2616.txt`. Último acesso, Abril de 2008.

[FHK06] Sally Floyd, Mark Handley, and Eddie Kohler. Problem Statement for the datagram congestion control protocol (DCCP), 2006. `http://www.ietf.org/rfc/rfc4336.txt`. Último acesso, Abril de 2008.

[Fil98] Reinaldo Penno Filho. Desvendando o TCP, 4 1998. `http://www.rnp.br/newsgen/9804/tcp.html`. Último acesso, Abril de 2008.

[FK06] Sally Floyd and Eddie Kohler. Profile for DCCP Congestion Control ID 2: TCP-like Congestion Control, 5 2006. `http://www.ietf.org/rfc/rfc4341.txt`. Último acesso, Abril de 2008.

[FKP06] Sally Floyd, Eddie Kohler, and Jitendra Padhye. Profile for DCCP Congestion Control ID 3: TFRC Congestion Control, 5 2006. `http://www.ietf.org/rfc/rfc4342.txt`. Último acesso, Abril de 2008.

[FL03] Cheng Peng Fu and Soung C. Liew. TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks. volume 21, pages 216–228, 2 2003.

[Flo00] Sally Floyd. Congestion Control Principles, 2000. `http://www.ietf.org/rfc/rfc2914.txt`. Último acesso, Abril de 2008.

[GDL+04] Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall. System Support for Pervasive Applications. volume 22, pages 421–486, New York, NY, USA, 2004. ACM Press.

[GDW05] X. Gu, P. Di, and L. Wolf. Performance Evaluation of DCCP: A Focus on Smoothness and TCP-friendliness. In *Annals of Telecommunications Journal, Special Issue on Transport Protocols for Next Generation Networks*, volume 1, pages 191–206, 1 2005.

[GS07] Christian Grimm and Harald Schwier. Empirical Analysis of TCP Variants and Their Impact on GridFTP Port Requirements. In *3th International Conference on Networking and Services 2007*, volume 1, pages 24–28, 6 2007.

[HBMP07] Simon Heimlicher, Rainer Baumann, Martin May, and Bernhard Plattner. The Transport Layer Revisited. In *2nd IEEE International Conference on Communication System Software and Middleware 2007*, pages 1233–1270, Bangalore, India, 1 2007. IEEE COMSWARE 2007.

[HDS05] Dongwoo Hong, Cameron Dryden, and Gordon Saksena. An Efficient Random Jitter Measurement Technique Using Fast Comparator Sampling. volume 00, pages 123–130, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[HLRX07] Sangtae Ha, Long Le, Injong Rhee, and Lisong Xu. Impact of Background Traffic on Performance of High-speed TCP Variant Protocols. volume 51, pages 1748–1762, New York, NY, USA, 2007. Elsevier North-Holland, Inc.

[Hos07] Ekram Hossain. IEEE802.16/WiMAX-Based Broadband Wireless Networks: Protocol Engineering, Applications, and Services. In *Fifth Annual Conference on Communication Networks and Services Research, 2007 (CNSR).*, volume 1, pages 3–4. IEEE, 1 2007.

[Hua06] Jiun-Lang Huang. A Random Jitter Extraction Technique in the Presence of Sinusoidal Jitter. volume 0, pages 318–326, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[IET06] IETF. IETF Overview, 2 2006. `http://www.ietf.org/overview.html`. Último acesso, Abril de 2008.

[IN07] Ibope and NetRatings. Internet Brasileira já Atinge 39 Milhões de Pessoas no País, 12 2007. Último acesso, Abril de 2008.

[IT88] ITU-T. Pulse Code Modulation (PCM) of Voice Frequencies, 11 1988. ITU-T Recomendations G.711.

[IT96a] ITU-T. Coding of Speech at 8 Kbit/s Using Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP), 3 1996. ITU-T Recomendations G.729.

[IT96b] ITU-T. Methods for Subjective Determination of Transmission Quality. ITU-T P.800 Recomendation, 8 1996.

[Jac98] V. Jacobson. Congestion Avoidance and Control. In *SIGCOMM '88. ACM*, volume 1, pages 43–51, 1998.

[Jan91] Raj Jan. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Inc, 1 edition, 3 1991.

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance, 5 1992. Último acesso, Abril de 2008.

[JF96] S. Jha and M. Fry. Continuous Media Playback and Jitter Control. volume 1, pages 245–267, Los Alamitos, CA, USA, 1996. IEEE Computer Society.

[Jun03] A. Jungmaier. *SCTP for Beginners*. Computer Network Tecnology Group, 1 edition, 9 2003.

[KBS+98] T. Kostas, M. Borella, I. Sidhu, G. Schuster, J. Grabiec, and J. Mahler. Real-time Voice Over Packet Switched Networks. volume 12, pages 18–27, 1 1998.

[KFS07] Eddie Kohler, Sally Floyd, and A. Sathiaseelan. Faster Restart for TCP Friendly Rate Control (TFRC), 7 2007. http://www.ietf.org/proceedings/07jul/IDs/draft-ietf-dccp-tfrc-faster-restart-03.txt. Último acesso, Abril de 2008.

[KHF06a] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), 3 2006. `http://www.ietf.org/rfc/rfc4340.txt`. Último acesso, Abril de 2008.

[KHF06b] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion Control Without Reliability. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–38, New York, NY, USA, 2006. ACM Press.

[KHF07] Eddie Kohler, Mark Handley, and Sally Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 4: TCP-Friendly Rate Control for Small Packets, 6 2007. `http://tools.ietf.org/wg/dccp/draft-ietf-dccp-ccid4`. Último acesso, Abril de 2008.

[KL05] E. Kohler and J. Lai. Efficiency and Late Data Choice in a User-kernel Interface for Congestion-Controlled Datagrams. In *12th Annual SPIE Conference on Multimedia Computing and Networking (MMCN 05)*, pages 925–931, San Jose, Califórnia, 2005. `http://www.cs.ucla.edu/~kohler/pubs/lai04efficiency.pdf`. Último acesso, Abril 2008.

[KMS95] P.J. Kyees, R.C. McConnell, and K. Sistanizadeh. ADSL: A New Twisted-Pair Access to the Information Highway. volume 33, pages 52–60, 4 1995.

[KR06] James F. Kurose and Keith W. Ross. *Redes de Computadores e a Internet: Uma Abordagem Top-Down*. Addison Wesley, São Paulo, trad. 3 ed. edition, 2006.

[LAS03] Anders Lindgren, Andreas Almquist, and Olov Schelén. Quality of Service schemes for IEEE 802.11 wireless LANs - An Evaluation. volume 8, pages 223–235, 6 2003.

[LCLZ06] Qi Li, Di Chen, Yuncai Liu, and Lina Zheng. Jitter Ratio Based TFRC Scheme in Wireless-Wired Hybrid Network. volume 1, pages 38 – 38, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[LMB+06] Sébastien Linck, Emmanuel Mory, Julien Bourgeois, Eugen Dedu, and François Spies. Video Quality Estimation of DCCP Streaming over Wireless Networks. In *14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, volume 1, pages 455–463, 2006.

[Mel05] Arnaldo Carvalho Melo. DCCP On Linux. In *Linux Symposium 2005 Proceedings*, volume 1, pages 313–320, 2005.

[MMFR96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options, 10 1996. Último acesso, Abril de 2008.

[Moc87] P. Mockapetris. Domain Names - Implementation and Specification, 11 1987. Último acesso, Abril de 2008.

[MPSWR96] Ketan Mayer-Patel, David Simpson, David Wu, and Lawrence A. Rowe. Synchronized Continuous Media Playback through the World Wide Web. In *MULTIMEDIA '96: Proceedings of the Fourth ACM International Conference on Multimedia*, pages 435–436, New York, NY, USA, 1996. ACM Press.

[NAT06] P. Navaratnam, N. Akhtar, and R. Tafazolli. On the Performance of DCCP in Wireless Mesh Networks. In *MobiWac '06: Proceedings of the International Workshop on Mobility Management and Wireless Access*, pages 144–147, New York, NY, USA, 2006. ACM Press.

[Niv05] Frederic Nivor. Experimental Study of DCCP for Multimedia Applications. In *International Conference On Emerging Networking Experiments And Technologies (CoNEXT'05)*, pages 272–273. ACM, 2005.

[Onl08] Globo Online. Nova Iorque Vai Cobrar Pedágio de Carros que Lotam o Centro da Cidade, 3 2008. `http://jornalnacional.globo.com/Jornalismo/JN/0,,AA1676949-3586,00.html%`. Último acesso, Abril de 2008.

[OSL+06] Loreno Oliveira, Leandro Sales, Emerson Loureiro, Hyggo Almeida, and Angelo Perkusich. Filling the Gap Between Mobile and Service Oriented Computing - Issues for Evolving Mobile Computing Towards Wired Infrastructures and Vice Versa. volume 2, pages 355–378, 2006.

[Per98] Charles Perkins. *Mobile IP: Design Principles and Practice*, volume 1. Addison Wesley, 1 edition, 7 1998.

[Per02] Charles Perkins. IP Mobility Support for IPv4, 1 2002. `http://www.ietf.org/rfc/rfc3220.txt`. Último acesso, Abril de 2008.

[Pra03] Joseph Pranevich. The Wonderful World of Linux 2.6, 2003. `http://www.kniggit.net/wwol26.html`. Último acesso, Abril de 2008.

[Res07] Paulo Resende. Carros Demais, 12 2007. http://jornalhoje.globo.com/JHoje/0,19125,VJS0-3076-20071204-312154,00.html. Último acesso, Abril de 2008.

[RFB01] K. Ramakrishnan, S. Floyd, and D. Black. ECN (Explicit Congestion Notification), 9 2001. `http://www.ietf.org/rfc/rfc3168.txt`. Último acesso, Abril de 2008.

[RFC80] RFC0768. UDP: User Datagram Protocol, 8 1980. Último acesso, Abril de 2008.

[RFC81] TCP: Tranmission Control Protocol, 9 1981. Último acesso, Abril de 2008.

[RFC90] RFC1191. Path MTU Discovery, 11 1990. Último acesso, Junho de 2005.

[RFC94] RFC1631. The IP Network Address Translator (NAT), 5 1994. `http://www.ietf.org/rfc/rfc1631.txt`. Último acesso, Abril de 2008.

[RFC03] RFC3540. Robust Explicit Congestion Notification (ECN) Signaling with Nonces to IP, 6 2003. Último acesso, Junho de 2005.

[RSC02] J. Rosenberg, H. Schulzrinne, and G. Camarillo. SIP: Session Initiation Protocol, 6 2002. `http://www.ietf.org/rfc/rfc3261.txt`. Último acesso, Abril de 2008.

[RWHM03] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), 2003. `http://www.ietf.org/rfc/rfc3489.txt`. Último acesso, Abril de 2008.

[RX05] Injong Rhee and Lisong Xu. CUBIC: A New TCP-Friendly High-Speed TCP Variant. volume 22, pages 921–932, 7 2005.

[SAP08] Leandro Sales, Hyggo Almeida, and Angelo Perkusich. The DCCP Protocol in Three Steps. *Linux Magazine*, (92), 7 2008. A ser publicado.

[SAPJ08] Leandro M. Sales, Hyggo O. Almeida, Angelo Perkusich, and Marcello Sales Jr. On the Performance of TCP, UDP and DCCP over 802.11g Networks. In *In Proceedings of the SAC 2008 23rd ACM Symposium on Applied Computing Fortaleza, CE*, pages 2074–2080, 1 2008.

[Sem96] C. Semeria. Understanding IP Addressing: Everything You Ever Wanted to Know, 4 1996. `http://www.3com.com/other/pdfs/infra/corpinfo/en_US/501302.pdf`. Último acesso, Abril de 2008.

[Ser07] IDG News Service. Internet Pode Ficar Sem Capacidade em 2010, 11 2007. Último acesso, Abril de 2008.

[SFR05] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. *Programação de Rede UNIX: API para Soquetes de Rede*. Bookman, Porto Alegre, trad. 3 edition, 2005.

[SW99] Dorgham Sisalem and Adam Wolisz. Towards TCP-friendly Adaptive Multimedia Applications based on RTP. In *Fourth IEEE Symposium on Computers and Communications (ISCC'99). Egito*, 7 1999.

[SXM+00] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol, 10 2000. `http://www.ietf.org/rfc/rfc2960.txt`. Último acesso, Abril de 2008.

[SZ06] Farhan Siddiqui and Sherali Zeadally. Mobility Management Across Hybrid Wireless Networks: Trends and Challenges. volume 29, pages 1363–1385, 2006.

[Tan03] Andrew S. Tanenbaum. *Redes de Computadores*. Elsevier, Rio de Janeiro, trad. 4 ed. edition, 2003.

[Tan04] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall - PTR, 4 edition, 8 2004.

[Tel07] Associação Brasileira Telecomunicações. Acesso a Banda Larga Dobra Até 2010, 7 2007. `http://tecnologia.terra.com.br/interna/0,,OI1625324-EI4802,00.html`. Último acesso, Abril de 2008.

[TIoEEE05a] IEEE The Institute of Electrical and Electronics Engineers. IEEE 802.11, IEEE Standard for Information Technology, 10 2005. `http://standards.ieee.org/getieee802/download/802.11-1999.pdf`. Último acesso, Abril de 2008.

[TIoEEE05b] IEEE The Institute of Electrical and Electronics Engineers. IEEE 802.11a, IEEE Standard for Information Technology, 10 2005. `http://standards.ieee.org/getieee802/download/802.11a-1999.pdf`. Último acesso, Abril de 2008.

[TIoEEE05c] IEEE The Institute of Electrical and Electronics Engineers. IEEE 802.11b, IEEE Standard for Information Technology, 10 2005. `http://standards.ieee.org/getieee802/download/802.11b-1999.pdf`. Último acesso, Abril de 2008.

[TIoEEE05d] IEEE The Institute of Electrical and Electronics Engineers. IEEE 802.11e, IEEE Standard for Information Technology, 10 2005. `http://standards.ieee.org/getieee802/download/802.11e-2005.pdf`. Último acesso, Abril de 2008.

[TIoEEE05e] IEEE The Institute of Electrical and Electronics Engineers. IEEE 802.11g, IEEE Standard for Information Technology, 10 2005. `http://standards.ieee.org/getieee802/download/802.11g-2003.pdf`. Último acesso, Abril de 2008.

[TKI+05] Shigeki Takeuchi, Hiroyuki Koga, Katsuyoshi Iida, Youki Kadobayashi, and Suguru Yamaguchi. Performance Evaluations of DCCP for Bursty Traffic in Real-Time Applications. volume 1, pages 142–149, 2005.

[WAR04] Q. Wang and M.A. Abu-Rgheff. Interacting Mobile IP and SIP for Efficient Mobility. In *Fifth IEEE International Conference On 3G Mobile Communication Technologies*, pages 664–668, 2004.

[WCSY07] D. Wu, Song Ci, H. Sharif, and Yang Yang. Packet Size Optimization for Goodput Enhancement of Multi-Rate Wireless Networks. In *Consumer Communications and Networking Conference Proceedings*, pages 3575–3580, 3 2007.

[Wel05] Michael Welzl. *Network Congestion Control: Managing Internet Traffic*, volume 1. Wiley, 1 edition, 5 2005.

[WF94] A. Wilde and U. C. Fiebig. Bit Error Rate Estimation for Low Rate Services in a Mobile Radio Environment. volume 3, pages 816–820, 9 1994.

[XD05] Lei Xiao and Xiaodai Dong. The Exact Transition Probability and Bit Error Probability of Two-Dimensional Signaling. volume 4, pages 2600–2609, 9 2005.

[XLZ05] Changbin Xu, Ju Liu, and Caihua Zhao. Performance Analysis of Transmitting H.263 Over DCCP. In *IEEE International VLSI Design & Video Tech*, Suzhou, China, 5 2005.

[XW01] Ying Xu and Wenjun Wu. Simulations of TCP Congestion Control Mechanism Model with Large Buffer Size. In *2001 International Conference on Computer Networks and Mobile Computing*, volume 1, pages 105–110, 2001.

Apêndice A

O Núcleo do Linux e o Protocolo DCCP

O Linux é um sistema operacional criado em 1991 por Linus Torvalds na universidade de Helsinki na Finlândia. É um sistema Operacional de código aberto distribuído gratuitamente pela Internet. Este apêndice foi escrito com base na referência [Pra03].

Linus originalmente quis nomear o novo sistema de Freax, mas isso não aconteceu. A primeira versão oficial do Linux 1.0 foi lançada em março de 1994, que incluia suporte oficial somente para o $i386$ e somente para máquinas com um único processador. O Linux 1.2 foi lançado em março de 1995 e foi a primeira versão a incluir suporte oficial para diferentes tipos de arquitetura de sistemas (Alpha, Sparc, e MIPS, especificamente). O Linux 2.0 foi lançado em junho de 1996 e incluia suporte para um número de novas arquiteturas, mas principalmente foi a primeira revisão a suportar máquinas multi-processadas (SMP).

O Linux 2.2 foi lançado em janeiro de 1999 como diversas melhorias, sobretudo para computadores com multi-processador e suportava uma quantidade maior de hardware. E, finalmente, o Linux 2.4 foi lançado em janeiro de 2001 como um grande melhoramento de escalabilidade com o SMP, mas também incluindo a integração de muitas funcionalidades de desktop na linha principal, como suporte a dispositivos USB, suporte a cartões PCMCIA, *plug-and-play* interno, etc.

Recentemente surgiu o Linux 2.6, o que não significa apenas um melhoramento nestas funcionalidades, mas também está sendo um grande evolução à medida que tem se trabalhado bastante no melhoramento do suporte tanto a sistemas significativamente maiores (*mainframes*, supercomputadores etc.) e significativamente menores (PDAs e celulares).

Um dos pontos fortes do Linux é que ele suporta uma ampla variedade de hardware e plataformas. Todas as versões desde a 1.2 têm incluído suporte para novos tipos de processadores e funcionalidades. A versão 2.6 do kernel Linux não é exceção a esta tendência.

Algumas características desse sistema operacional são:

- é livre e desenvolvido voluntariamente por programadores experientes, *hackers*, e contribuidores espalhados ao redor do mundo que tem como objetivo a contribuição para a melhoria e crescimento deste sistema operacional;
- multitarefa real e multiusuário;

- modularização: o Linux somente carrega para a memória o que é usado durante o processamento, liberando totalmente a memória assim que o programa/dispositivo é finalizado;
- acessa corretamente vários sistemas de arquivos de outros sistemas operacionais: Novell, OS/2, Windows, DOS, SunOS, Amiga, Atari, Mac etc;
- rede TCP/IP mais rápida que no Windows e tem sua pilha constantemente melhorada. O Linux tem suporte nativo a redes TCP/IP e não depende de uma camada intermediária como o WinSock. Em acessos via modem a Internet, a velocidade de transmissão é 10% maior;
- suporte a dispositivos infravermelho;
- suporte a rede sem fio;
- suporte a dispositivos USB;
- firewall de alta qualidade e com bastante flexibilidade para adicionar novos recursos de segurança;
- roteamento estático e dinâmico de pacotes;
- ponte entre redes;
- proxy tradicional e transparente;
- recursos para atender a mais de um endereço IP na mesma placa de rede, sendo muito útil para situações de manutenção em servidores de redes ou para a emulação de "mais computadores" virtualmente;

A.1 DCCP no Núcleo do Linux

A primeira versão oficial do núcleo do Linux 2.6 que oferece suporte ao protocolo DCCP é a $2.6.14$. Nesta Seção são apresentados os procedimentos para utilizar o protocolo DCCP no Linux, como o processo de compilação, o carregamento dos módulos necessários para a execução de uma aplicação simples que utiliza o protocolo DCCP.

Para a composição desta seção foi utilizado o kernel $2.6.25 - rc8$, que pode ser obtida através do endereço `http://www.kernel.org/`.

Após baixar o código fonte do kernel, basta descompactar o arquivo (*linux-2.6.25.8.tar.bz2*), como descrito no trecho de código A.1, assumindo que o arquivo foi baixado para o diretório */usr/src*.

Codigo A.1: Descompactando o código fonte do kernel do Linux

```
ephone01:~# cd /usr/src <enter>
ephone01:/usr/src# tar -zjvf linux-2.6.25-rc8.tar.bz2 <enter>
ephone01:/usr/src# ln -s linux-2.6.25-rc8 linux <enter>
ephone01:/usr/src# cd linux <enter>
```

Ao realizar este procedimento, é possível iniciar o processo de compilação, que consiste na escolha dos módulos que estarão disponível no kernel, como apresentado no trecho de código A.2.

Codigo A.2: Utilizando o comando make menuconfig

```
ephone01:/usr/src# make menuconfig <enter>
scripts/kconfig/mconf arch/i386/Kconfig
## using defaults found in .config #
```

Neste estágio aparecerá uma tela como mostrado na Figura A.1.

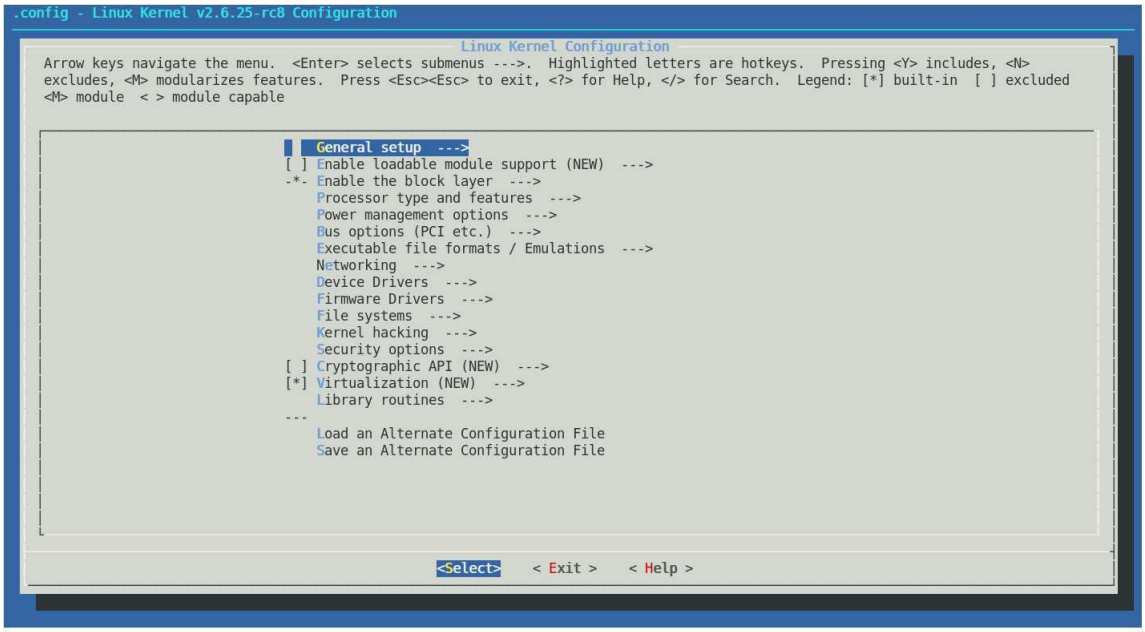


Figura A.1: Tela principal para compilação do núcleo do Linux.

Várias opções podem ser selecionadas. Para a compilação do protocolo DCCP, as seguintes opções devem está selecionadas, semelhante ao que é ilustrado na Figura A.2. Note que na figura mencionada não são mostradas as opções internas. Uma das opções internas que podem ser escolhidas são os algoritmos de controle de congestionamento do DCCP.

Networking —>
Network Support —>
DCCP Configuration —>
<M> The DCCP Protocol
DCCP CCIDs Configuration —>
<M> CCID2 (TCP-Like) (se existir)
<M> CCID3 (TFRC)

Para saber detalhes acerca de como proceder para compilar o kernel do linux, pode-se consultar o arquivo $README$ que é descompactado junto com os fontes.

É possível baixar a versão do kernel com os últimos códigos de implementação do protocolo DCCP antes de serem disponibilizados na versão oficial do kernel do linux. Esta versão pode

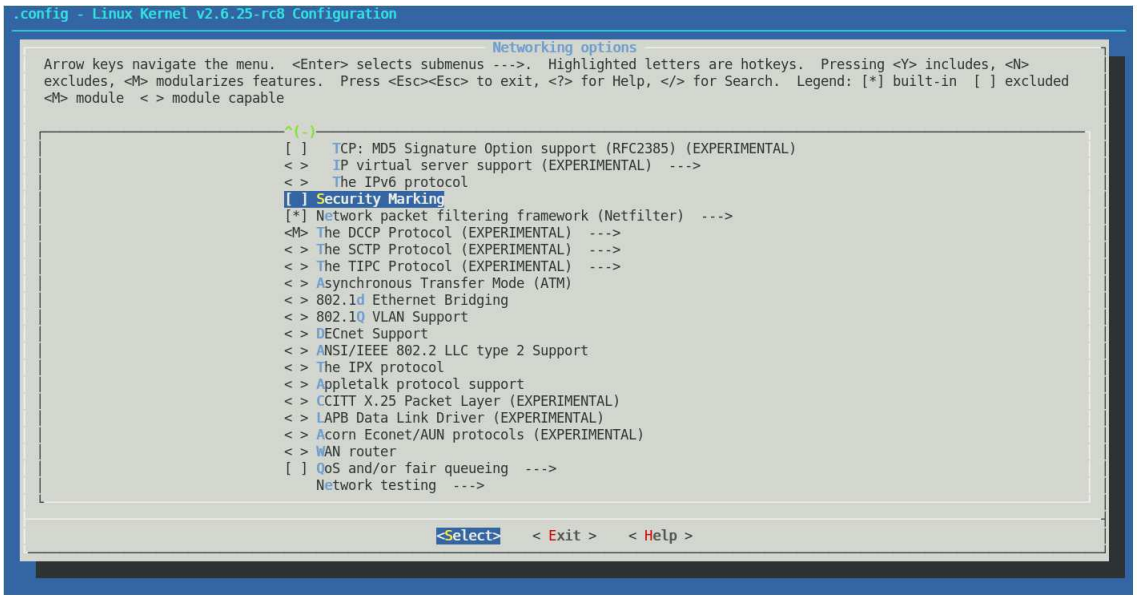


Figura A.2: Opção para ativação do protocolo DCCP no núcleo do Linux.

ser obtida através do endereço `http://www.kernel.org/git/?p=linux/kernel/git/acme/net-2.6.git`.

A.2 Exemplo de Socket DCCP em C, Python, PHP

Nesta seção são apresentados exemplos de como utilizar o protocolo DCCP através das linguagens de programação C, Python e PHP.

A.2.1 Linguagem C

Codigo A.3: Exemplo Cliente/Servidor DCCP em C

```
-- src/servidor_dccp.c

#include <arpa/inet.h>
#include <errno.h>
#define IPPROTO_DCCP 33
#define SOL_DCCP 269
#define MAX_DCCP_CONNECTION 5

(...)
int result = 0;
if ((server_sock_fd = socket (AF_INET, SOCK_DCCP,
                                        IPPROTO_DCCP)) > 0) {
    int result = connect(sock_fd, (struct sockaddr *)&serverSockIn,
```

```
                        sizeof(serverSockIn));
    return result;
} else {
        printf("Conexão Recusada.");
}

// Iniciando o servidor
int result = setsockopt(sock_fd, SOL_DCCP, SO_REUSEADDR);
// local_name é sockaddr do endereço IP local e porta
result = bind(sock_fd, (struct sockaddr *)&local_name,
                                        sizeof(local_name));
result = listen(server_sock_fd, MAX_DCCP_CONNECTION);
// Aceitando conexões de clientes
sock_client_fd = accept(server_sock_fd,
                (struct sockaddr *)&remote_name, &remote_length);

// Recebe dados do cliente
char receive_buff[100];
int rec_size = recv(sock_client_fd, receive_buff, 100, 0);
(...)

-- src/client_dccp.c
int src_client_fd = socket(AF_INET, SOCK_DCCP, IPPROTO_DCCP);
int on = 1;
char sendbuffer[] = "Alo Mundo!!!";
int status;

int result = setsockopt(mSocketHandle, SOL_DCCP, SO_REUSEADDR);

result = connect(src_client_fd, (struct sockaddr *)&remote_name,
                                            sizeof(remote_name));
do {
   status = send(src_client_fd, sendbuffer, strlen(sendbuffer), 0);
} while((status < 0) && (errno == EAGAIN));

(...)
```

Linguagem Python

Codigo A.4: Exemplo Cliente/Servidor DCCP em Python

```
-- src/servidor_cliente_dccp.py

#!/usr/bin/python

import socket

socket.DCCP_SOCKOPT_PACKET_SIZE = 1
socket.DCCP_SOCKOPT_SERVICE     = 2
socket.SOCK_DCCP                = 6
socket.IPPROTO_DCCP             = 33
socket.SOL_DCCP                 = 269
packet_size                     = 256
address                         = (socket.gethostname(),12345)

# Cria dois sockets, um cliente e um servidor
server,client = [socket.socket(socket.AF_INET, socket.SOCK_DCCP,
                               socket.IPPROTO_DCCP) for i in range(2)]
for s in (server,client):
    s.setsockopt(socket.SOL_DCCP, socket.DCCP_SOCKOPT_PACKET_SIZE,
                                                    packet_size)
    s.setsockopt(socket.SOL_DCCP, socket.DCCP_SOCKOPT_SERVICE,
                                                    True)
# Conecta os sockets
server.bind(address)
server.listen(1)
client.connect(address)
s,a = server.accept()

# O cliente envia mensagens e o servidor imprime na tela
while True:
    client.send(raw_input("Digite uma mensagem: "))
    print "Recebendo:", s.recv(1024)

(...)
```

PHP

Codigo A.5: Exemplo Cliente/Servidor DCCP em PHP

```
------ ServidorDCCP.php ------
```

```
#!/usr/bin/php

$DCCP_SOCKOPT_PACKET_SIZE = 1
$DCCP_SOCKOPT_SERVICE     = 2
$SOCK_DCCP                = 6
$IPPROTO_DCCP             = 33
$SOL_DCCP                 = 269

$bind_address = "192.168.1.1";
$port = 9011;

// Connect sockets
$server_socket_fd = socket_create(AF_INET,
                          $DCCP_SOCKOPT_PACKET_SIZE, $IPPROTO_DCCP);
socket_bind ($server_socket_fd, $bind_address, $port);
socket_listen($server_socket_fd);
$client_socket_fd = socket_accept($server_socket_fd);
$recv = socket_read($client_socket_fd, 11);
echo $recv;
socket_close($server_socket_fd);

----- ClienteDCCP.php -----
#!/usr/bin/php
$bind_address = "192.168.1.1";
$port = 9011;

$client_socket_fd = socket_create(AF_INET,
                          $DCCP_SOCKOPT_PACKET_SIZE, $IPPROTO_DCCP);

$message = "Alô mundo!!!";

if (socket_connect ($client_socket_fd, $bind_address, $port)) {
     socket_send($client_socket_fd, $message, strlen($message), 0);
}

socket_close($client_socket_fd);
```

A.3 Aplicações Conhecidas que Suportam DCCP

Diversas aplicações já suportam o protocolo DCCP. Segue uma lista de algumas aplicações conhecidas.

GStreamer: `http://garage.maemo.org/projects/ephone`

IPerf: `http://www.erg.abdn.ac.uk/users/gerrit/dccp/apps/`

IPTables: `http://www.netfilter.org/`

TCPDump: `http://www.tcpdump.org/`

TTCP: `http://www.erg.abdn.ac.uk/users/gerrit/dccp/apps/`

VLC: `http://www.videolan.org/`

EtherReal/Wireshark: `http://www.wireshark.org/`

D-ITG: `http://www.grid.unina.it/software/ITG`

NetCat: `http://netcat.sf.net/`

SpeexComm: `http://tuomas.kulve.fi/projects/speexcomm/`

Para maiores informações a respeito de como utilizar DCCP em cada uma dessas aplicações, acesse o endereço `http://www.linux-foundation.org/en/Net:DCCP`.

Apêndice B

Utilizando IPerf e GStreamer para Transmitir Dados via DCCP

Neste apêndice é apresentado um guia de como utilizar duas aplicações usadas nos experimentos realizados nesta dissertação. Uma é a IPerf, utilizada nos experimentos para gerar e transmitir dados na rede. A outra é a GStreamer, um arcabouço multimídia baseado em componentes para processamento de conteúdos multimídia.

B.1 IPerf

O Iperf é uma aplicação de análise de desempenho de rede e relatórios estatísticos sobre uma determinada transmissão de dados na rede. O IPerf é mantido pela Universidade de Illinois sob licença GPL. Ele pode ser obtido através do endereço `http://dast.nlanr.net/Projects/Iperf/`.

O IPerf pode ser entendido como uma aplicação cliente/servidor adequada para medições de rede. Mas além dessas medições, o IPerf pode ser usado com um gerador simples de carga na rede, quando não há preocupação com o perfil do tráfego que está sendo gerado. Além disso, é possível medir o jitter (variação do atraso) e a quantidade de pacotes perdidos. Originalmente ele suporta apenas os protocolos UDP e o TCP, sendo capaz de permitir múltiplas conexões simultâneas. Para que o IPerf suporte o protocolo DCCP, utilize a versão do IPerf disponível através da endereço `http://www.erg.abdn.ac.uk/users/gerrit/dccp/apps/`.

Para utilizar o protocolo DCCP com o IPerf existem dois modos: o modo cliente e o modo servidor. Para iniciar os modos cliente e servidor do IPerf, utilize os comandos apresentados no trecho de comandos B.1.

Codigo B.1: Comando para executar o IPerf com suporte a DCCP

```
Servidor:
  # iperf -d -s -i 1 -t 0 -l 1424 -b 10m
Cliente:
  # iperf -d -c 192.168.1.1 -l 1424 -b 10m
```

onde,

- **-d** determina que o protocolo de transporte é o DCCP;
- **-s** determina que o IPerf deve ativar o modo servidor;
- **-i** imprime relatório a cada x segundos, neste caso a cada segundo;
- **-l** tamanho do pacote, neste caso 1424;
- **-b** limite da taxa de transmissão, neste caso 10 Mbps;
- **-c** determinar que o IPerf deve ativar o modo cliente;

B.2 Arcabouço GStreamer

O GStreamer é um arcabouço de código aberto que permite um programador desenvolver aplicações multimídia de vários tipos e em várias linguagens, dentre elas C, C++, Python e outras. Existem diversas aplicações bastante conhecidas que fazem uso do GStreamer, tais como a Kaffeine[1], amaroK[2], Phonon[3], Rhythmbox[4] e Totem[5]. O GStreamer facilita o processo de desenvolvimento de aplicações multimídia, considerando os tipos mais conhecidos, como reprodutores de áudio e vídeo, editores multimídia e aplicações que transmitem fluxos multimídia na rede.

O GStreamer é um arcabouço baseado em componentes, sendo cada componente composto por elementos. Esses elementos é responsável por uma função específica, como codificação de áudio, reprodução de um vídeo, visualização de uma imagem, conversão de um formato para outro, escrita e leitura de arquivos, etc.

Através da ligação desses elementos, o programador pode construir um *pipeline* e assim realizar funções mais complexas. Por exemplo, é possível criar um *pipeline* que ler um áudio de um arquivo *MP3*, decodificar esse arquivo e reproduzí-lo nas caixas de som do computador ou transmití-lo na rede. Na Figura B.1 é ilustrada uma representação típica de um *pipeline* no GStreamer para transmitir um arquivo de áudio utilizando o componente DCCP. Neste exemplo, o *pipeline* é composto por três elementos, o *Elemento A*, o *B* e o *C*. O *elemento A* é a fonte de dados, neste caso um elemento capaz de ler um arquivo. O *elemento B*, capaz de interpretar os dados lidos do arquivo como sendo no formato MP3. E por último o *elemento C*, um elemento que transmite os dados codificados pelo *elemento B* para um sistema remoto capaz de receber dados via DCCP.

Em termos do componente DCCP para GStreamer, ele oferece 4 elementos: o *dccpserversrc*, o *dccpserversink*, o *dccpclientsrc* e o *dccpclientsink*. Os elementos **fontes** (src) (*dccpserversrc* e *dccpclientsrc*) são responsáveis por ler dados de um *socket* DCCP e repassar para o próximo elemento do

[1] http://kaffeine.kde.org/
[2] http://amarok.kde.org/
[3] http://phonon.kde.org/
[4] http://www.gnome.org/projects/rhythmbox/
[5] http://www.gnome.org/projects/totem/

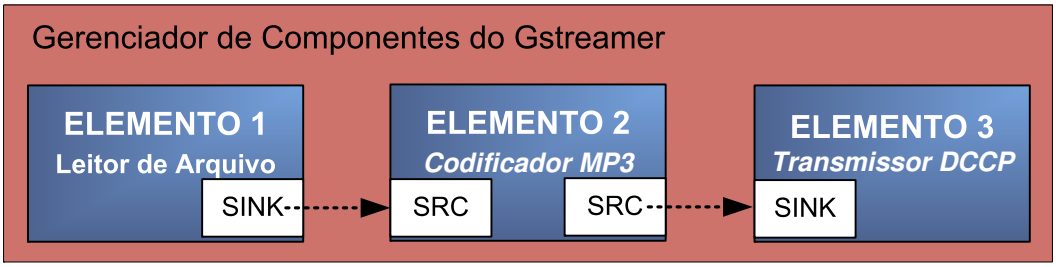


Figura B.1: *Pipeline* do GStreamer contendo 3 elementos.

pipeline, enquanto que os elementos **transmissores** (*dccpserversink* and *dccpclientsink*) são responsáveis por receber os dados do elemento anterior dentro do *pipeline* e escrever no *socket* DCCP.

Por exemplo, considere o trecho de código B.2, bastante similar aos comandos citados na Seção 4.3. Para o primeiro comando, os elementos do servidor são *filesrc*, *mad* e *dccpserversink*. O primeiro ler os dados de áudio do arquivo e o repassa para o segundo elemento. Este por sua vez decodifica os dados *MP3* recebidos pelo *filesrc* e repassa os dados para o próximo elemento. Neste caso, o elemento *dccpserversink* inicia um servidor DCCP e aguarda por conexões de clientes DCCP. Quando um se conecta no servidor, este começa a transmitir o áudio correspondente.

Para o segundo comando, os elementos do cliente são o *dccpclientsrc* e o *alsasink*. Se em vez de reproduzir, o usuário preferisse salvar o arquivo recebido (o que foi feito nos experimentos da Fase 3), bastaria substituir o último elemento (*alsasink*) pelo elemento *filesink*, considerando o parâmetro *location*, o qual é utilizado para informar o nome do arquivo a ser salvo.

Codigo B.2: Comando para executar o componente de DCCP para GStreamer

```
Servidor:
  # gst-launch -v filesrc location=yourmusic.mp3 ! mp3parse ! \
        dccpserversink port=9011 ccid=2
Cliente:
  gst-launch -v dccpclientsrc host=192.168.1.1 port=9011 \
        ccid=2 ! alsasink
```

O comando $gst - launch$ é uma aplicação disponibilizada pelo arcabouço GStreamer que permite criar *pipelines* através da linha de comandos. Para utilizar este comando e o componente de DCCP para GStreamer, é necessário baixar na Internet e instalar o arcabouço do GStreamer e o componente de DCCP para GStreamer, sendo este último disponibilizado pelo projeto E-Phone, apresentado no Capítulo 8. Para instalar o arcabouço GStreamer acesse o endereço `http://gstreamer.freedesktop.org`. Com relação ao componente DCCP, é possível obtê-lo através do endereço `https://garage.maemo.org/frs/?group_id=297`. Para instalar o componente de DCCP para GStreamer, proceda da seguinte forma:

Codigo B.3: Compilar e instalar o componente DCCP para GStreamer

```
./configure --prefix=/usr
make
```

```
make install
```

Como já foi dito, é possível utilizar os componentes GStreamer para desenvolver aplicações multimídia. Para o caso do componente DCCP para GStreamer, o processo é bastante simples. Por exemplo, os trechos de código B.4 representa uma aplicação que realiza a mesma função do cliente via linha de comandos, exemplificado nos trechos de código B.2. Essa aplicação conecta em um servidor que transmite áudio em *MP3* utilizando o protocolo DCCP e o reproduz no sistema local.

Codigo B.4: Exemplo de código para uma aplicação cliente baseada no componente de DCCP para GStreamer

```
static gboolean bus_message_handler (GstBus *bus,
                                     GstMessage *msg, gpointer data){
    GMainLoop *loop = (GMainLoop *) data;
    switch (GST_MESSAGE_TYPE (msg)) {
    case GST_MESSAGE_EOS:
        g_print ("End-of-stream\n");
        g_main_loop_quit (loop);
        break;
    case GST_MESSAGE_ERROR: {
        gchar *debug;
        GError *err;
        gst_message_parse_error (msg, &err, &debug);
        g_free (debug);
        g_print ("Error: %s\n", err->message);
        g_error_free (err);
        g_main_loop_quit (loop);
        break;
    }
    return TRUE;
}

int main(int argc, char *argv[]) {
    /* Inicia o GStreamer */
    gst_init (&argc, &argv);
    loop = g_main_loop_new (NULL, FALSE);

    /* verifica dados de entrada */
    if (argc != 3) {
        g_print ("%s\n", "Uso: ip porta");
        return -1;
    }
```

```
/* cria os elementos */
pipeline = gst_pipeline_new ("audio-sender");
alsasink = gst_element_factory_make ("alsasink", "alsa-sink");
dccpclientsrc = gst_element_factory_make ("dccpclientsrc",
                                          "client-source");

if (!pipeline || !alsasink || !dccpclientsrc) {
    g_print ("Erro ao criar um ou mais elementos\n");
    return -1;
}

/* define o tipo de dado a ser recebido, nesse caso uma MP3 */
caps = gst_caps_from_string("audio/x-raw-int,
                            endianness=(int)1234,
                            signed=(boolean)true,
                            width=(int)32,
                            depth=(int)32,
                            rate=(int)44100,
                            channels=(int)2");
g_object_set (G_OBJECT (dccpclientsrc), "caps", caps, NULL);
gst_object_unref (caps);
g_object_set (G_OBJECT (dccpclientsrc), "host", argv[1], NULL);
g_object_set (G_OBJECT (dccpclientsrc), "port", atoi(argv[2]),
                                                NULL);

/* adiciona todos os elementos no pipeline */
gst_bin_add_many (GST_BIN (pipeline), dccpclientsrc, alsasink,
                                                NULL);
gst_element_link_many (dccpclientsrc, alsasink, NULL);
bus = gst_pipeline_get_bus (GST_PIPELINE (pipeline));
gst_bus_add_watch (bus, bus_message_handler, loop);
gst_object_unref (bus);

/* alterando o estado do pipeline para play */
g_print ("Setting to PLAYING\n");
gst_element_set_state (pipeline, GST_STATE_PLAYING);
g_print ("Running\n");
g_main_loop_run (loop);

/* libera recurso quando acaba a transmissão */
```

```
    g_print ("Returned, stopping playback\n");
    gst_element_set_state (pipeline, GST_STATE_NULL);
    g_print ("Deleting pipeline\n");
    gst_object_unref (GST_OBJECT (pipeline));

    return 0;
}
```

Após criar o arquivo de exemplo acima, para compilar siga as seguintes instruções:

Codigo B.5: Exemplo de execução da aplicação cliente baseada no componente de DCCP para GStreamer

```
# gcc $(pkg-config --cflags --libs gstreamer-0.10) cliente_dccp.c
  -o cliente_dccp
# ./client_dccp 192.168.1.1 9011
```

Esta dissertação foi composta na tipologia
Times New Roman, em corpo 10-25, e impressa
em papel Sulfite 75g/m^2.