

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Transporte de Fluxos de Dados Controlados e Não
Confiáveis para Distribuição de Conteúdo
Multimídia em Tempo Real

Leandro Melo de Sales

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

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

Angelo Perkusich
(Orientador)

Campina Grande, Paraíba, Brasil
©Leandro Melo de Sales, 01/08/2011

Resumo

A transmissão de conteúdo multimídia em tempo real através da Internet é de fundamental importância para as diversas aplicações atuais como voz sobre IP, videoconferência, jogos e TV pela Web. Os protocolos de transporte mais conhecidos - TCP e UDP - não são efetivos quando se deseja transmitir dados destas aplicações. Neste sentido, a IETF tem trabalhado em novos protocolos de transporte que aumentem a qualidade destas aplicações multimídia. Dentre estes novos protocolos, o DCCP (RFC 4340) é efetivo em transmissões de conteúdo multimídia através da Internet. Todavia, o DCCP não é efetivo em cenários onde existam muitos nós receptores e apenas um nó transmissor. Portanto, este artigo propõe o MU-DCCP, uma extensão do protocolo DCCP capaz de permitir transmissões de dados multimídia de 1 para muitos nós e controle de congestionamento de tráfego não confiável. O MU-DCCP utiliza multicast quando disponível ou um fluxo unicast por cada rede de destino e compartilhamento de dados entre os nós receptores. Com o uso do MU-DCCP, constata-se uma redução considerável de congestionamento na rede.

Abstract

Abstract Here

Agradecimentos

Agradecimentos

Conteúdo

1	Introdução	1
1.1	Panorama Atual: Protocolos de Transporte da Internet	5
1.2	Descrição do Problema	9
1.3	Objetivos da Tese	14
1.3.1	Objetivo Principal	14
1.3.2	Objetivos Específicos	15
1.4	Relevância do Tema e da Tese	16
1.5	Resumo das Contribuições até o Momento	17
1.6	Estrutura do Documento	18
2	Fundamentação	20
2.1	Camada de Transporte TCP/IP	20
2.2	Protocolos de Redes de Computadores	21
2.3	Visão geral do modelo de referência TCP/IP	22
2.4	A Camada de Transporte	24
2.4.1	Transporte de dados não orientado à conexão	28
2.4.2	Transporte de dados orientado à conexão	29
2.5	Princípios de Controle de Congestionamento	29
2.5.1	O que é congestionamento de rede?	29
2.5.2	Duas Abordagens para Controle de Congestionamento	30
2.5.3	ECN - <i>Explicit Congestion Notification</i>	31
2.6	Protocolo DCCP	32
2.6.1	Principais Características do DCCP	34
2.6.2	Estrutura do protocolo DCCP	35

2.6.3	Algoritmos de Controle de Congestionamento (CCIDs)	41
2.6.4	DCCP e o Protocolo IP	45
2.7	Protocolo TCP	47
2.7.1	Transporte Confiável, sem Duplicação, com Ordenação e Verificação de Erro	47
2.7.2	Controle de Fluxo do TCP	48
2.7.3	Controle de Congestionamento do TCP	49
2.7.4	Outros Algoritmos de Controle de Congestionamento do TCP . . .	52
2.8	Protocolo UDP	55
2.9	Comparação do DCCP com o TCP e UDP	56
2.10	Transmissão Multicast	57
2.10.1	Transmissão Multicast no Escopo do Trabalho	57
2.11	Fundamentação X	57
2.11.1	X no Escopo do Trabalho	57
3	Trabalhos Relacionados	58
3.1	Tópicos de Comparação	58
3.2	Descrição dos Trabalhos	59
3.2.1	Trabalho 1	59
3.2.2	Trabalho 2	59
3.2.3	Trabalho 3	60
3.2.4	Outros Trabalhos	60
3.3	Considerações Finais	60
4	Especificação do MU-DCCP	61
4.1	Visão Geral do MU-DCCP	62
4.2	Modos de Transmissão do MU-DCCP	63
4.3	Controle de congestionamento com DCCP em modo multicast	63
4.4	Eleição de Nós DCCP Relays e de DCCP Reporters	64
4.5	Adaptação de Fluxo Multimídia	65
4.6	Compatibilidade com outros protocolos e as recomendação da IETF	66
4.7	Considerações sobre redes de distribuição de conteúdo	66

4.8	Considerações sobre a escolha do DCCP como base para o MU-DCCP . . .	66
4.9	Considerações sobre segurança	66
5	Implementação do MU-DCCP	67
5.1	67
6	Métodos, Simulações e Experimentos	68
6.1	Parâmetros para os Experimentos	69
6.2	Experimentos	71
6.3	Métricas Seleccionadas e Métricas Derivadas	71
6.3.1	Vazão Média, Carga Efetiva Média, Latência Média e Jitter	72
6.4	Metodologia Estatística para o Cálculo Final das Métricas Estudadas	74
6.5	Metodologia para Comparação da Qualidade de Áudio	77
7	Análise de Desempenho do DCCP	79
7.1	79
8	Análise de Desempenho do MU-DCCP	80
8.1	80
9	Conclusão e Planejamento	81
9.1	Contribuições	81
9.2	Limitações do Trabalho	81
9.3	Perspectivas	81
A	A Ferramenta GST-DCCP	84
B	DCCP no NS-3	85

Lista de Símbolos

UE - *Um Exemplo*

OE - *Outro Exemplo*

Lista de Figuras

1.1	TCP × UDP. Após 50 s do início do experimento, o fluxo UDP ocupa toda a largura de banda disponível na rede.	6
1.2	TCP Reno × UDP, sendo o TCP enviando um arquivo de áudio.	7
1.3	TCP × DCCP. Ambos os protocolos conseguem transmitir dados na rede. .	8
1.4	Topologia da rede definida para as simulações realizadas. Cada rede é representada por um roteador e com 10 nós em cada rede.	12
1.5	Gráfico de uma transmissão DCCP com um transmissor enviando dados de áudio VoIP utilizando o protocolo DCCP para X receptores. A vazão média de cada fluxo tende a zero à medida que o número de receptores aumenta. .	13
2.1	Modelo de Referência TCP/IP e Protocolos de Internet	23
2.2	Comunicação lógica entre as camadas de transporte dos sistemas finais . . .	26
2.3	Processo de adição de cabeçalhos aos pacotes à medida que estes passam pelas diversas camadas da rede	27
2.4	Comunicação entre a camada de transporte e a camada de aplicação através de um <i>socket</i>	28
2.5	Ciclo de vida de uma conexão DCCP.	36
2.6	Conexão bi-direcional do protocolo DCCP.	36
2.7	Cabeçalho do protocolo DCCP (extendido).	40
2.8	Cabeçalho do protocolo DCCP (simplificado).	40

Lista de Tabelas

2.1	Tipos de Pacotes do protocolo DCCP.	42
2.2	Tabela comparativa das características do TCP, UDP e DCCP.	57
6.1	Quantidade de fluxos utilizados por confronto de protocolos.	69
6.2	Algoritmos de controle de congestionamento utilizados nos experimentos. .	70

Lista de Códigos Fonte

2.1	Conexão DCCP Através da API de Socket Berkeley	46
-----	--	----

Capítulo 1

Introdução

“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)

A transmissão de conteúdos multimídia em tempo real através da Internet tornou-se uma necessidade em aplicações como Voz sobre IP (VoIP), Videoconferência, Jogos e WebTV. Aplicações deste tipo implementam mecanismos sofisticados para melhorar a qualidade dos fluxos de dados e utilizar de forma eficiente os recursos disponíveis da rede. Na prática, tais mecanismos são implementados em forma de protocolos de rede com o intuito de: (i) reduzir altos níveis de congestionamento na rede; (ii) manter a equidade entre diferentes fluxos transmitidos pelos sistemas finais; e (iii) garantir níveis mínimos de qualidade do conteúdo multimídia sendo transmitido.

Aplicações como as supracitadas estão ganhando cada vez mais espaço, principalmente na Internet. Para se ter uma idéia deste crescimento, segundo previsão da empresa Cisco [11], em 2014 o tráfego de vídeo será maior do que o tráfego P2P em 2009, correspondendo a 39 % do tráfego de dados total na Internet. A empresa prevê também que em 2014, o tráfego de VoIP, vídeo e jogos na Internet alcançará a marca de 40 Exabytes/mês, quase 50 % do tráfego de dados total na Internet previsto para 2014. Embora já se saiba o potencial do modelo de serviço P2P, esta previsão demonstra o nível de aceitação dos usuários em compartilhar

seus arquivos e obter mais dados disponibilizados por outros usuários. Recentemente, a Paramount Pictures publicou uma nota [5] informando que passará a transmitir filmes através de grandes redes P2P, tais como a BitTorrent. A própria empresa BitTorrent anunciou que está trabalhando em uma aplicação P2P para transmissão de conteúdos multimídia em tempo real. Já a Amazon anunciou que também entrará na disputa por uma fatia deste tipo de serviço [1]. A empresa disponibilizará um reprodutor de conteúdo multimídia completamente *online*, onde as pessoas poderão comprar músicas e vídeos *online* e em seguida reproduzi-los em tempo real, com os dados armazenados na infra-estrutura de computação nas nuvens da empresa.

Seguindo a tendência das previsões e dos ~~simples~~ exemplos citados anteriormente, é notório o crescimento de serviços na Internet para distribuição de conteúdos multimídia, sejam serviços onde as empresas disponibilizam os conteúdos totalmente online, sejam os casos em que os próprios usuários disponibilizam tais conteúdos a partir de suas próprias residências. Para este último caso, sites *online*, como o *livestream.tv*, estão se popularizando cada vez mais ao permitirem que os usuários ou empresas transmitam conteúdos multimídia a partir de uma fonte dados multimídia (geralmente um sistema final) para milhares de usuários conectados à Internet seguindo os modelos cliente-servidor ou P2P (entre pares ou Peer-to-Peer).

Atualmente, tecnologias baseadas na arquitetura P2P estão substituindo a arquitetura tradicional cliente-servidor. ~~Arquiteturas baseadas em P2P~~ podem aliviar a carga imposta aos servidores e às redes de computadores. Nesta arquitetura, cada participante do sistema é capaz de obter o serviço de visualização da mídia, e também ~~capaz de~~ contribuir com outros participantes, fornecendo parte do conteúdo da mídia. Assim, a banda de rede necessária em um único ponto no modelo cliente-servidor é compartilhada entre os diversos participantes do sistema de transmissão de conteúdos multimídia em tempo real [?].

Diante deste cenário, desenvolver protocolos de rede para transportar dados desse tipo de aplicação e dar suporte a estas novas formas de fornecer serviços multimídia através da Internet tem se tornado uma tarefa ainda mais complexa. Exige-se uma harmonia entre os requisitos mencionados anteriormente em meio às mudanças no consumo de conteúdos multimídia, nas estratégias adotadas pelos fornecedores de conteúdos para distribuí-los entre os diversos clientes e na disponibilidade de recursos de rede que nem sempre são utilizados

de forma eficiente.

Ao longo dos anos, diversos esforços acadêmicos e da indústria foram feitos para disponibilizar protocolos e sistemas de transmissão para distribuição de conteúdo em tempo real na Internet [?; ?; ?; ?; ?]. Porém, estas soluções são isoladas e assim de baixa utilização na prática devido a falta de padronização, principalmente por serem executadas na camada de aplicação, causando uma pulverização de diferentes soluções e nenhuma realmente efetiva para ser utilizada em larga escala, como na Internet. Estas soluções tentam suprir as limitações existentes dos protocolos da camada de transporte, porém, salvo suas devidas contribuições científicas e reais, tratam-se de soluções complementares e não fundamentais para uma forma efetiva e padronizada de se transportar e distribuir dados multimídia nas redes de computadores. Isto gere que soluções mais efetivas no uso de recursos da rede e principalmente em sua adoção sejam realizadas na camada de transporte e não na camada de aplicação.

Do ponto de vista da camada de transporte da pilha TCP/IP, protocolos tradicionais como o UDP e o TCP não foram projetados para este fim, sendo deixado a cargo dos desenvolvedores das aplicações implementarem seus próprios mecanismos para controle de congestionamento de datagramas sem garantia de entrega, particularmente no caso do uso do UDP, utilizado largamente em aplicações multimídia. Nos casos em que se tenta utilizar TCP em aplicações multimídia em tempo real, o mesmo não apresenta um desempenho satisfatório porque o controle de congestionamento e a garantia de entrega são implementados de uma forma não adequada para aplicações multimídia com transporte de dados em tempo real [?; ?; ?].

Os cenários de aplicação mais críticos do ponto de vista da ineficiência de utilização de recursos de rede são aqueles que apresentam um nó de rede gerador de conteúdo em tempo real e milhares de nós receptor ($1 \rightarrow N$). Tais aplicações compartilham uma característica comum: a existência de muitos usuários interessados por um mesmo conteúdo e pouco ou nenhum dado individualizado, ou seja, que precisa ser transmitido apenas para um usuário ou um grupo restrito deles. Existem diversos exemplos de aplicações que podem ser citados, principalmente quando se utiliza o modelo de serviço P2P:

- Aplicações para transmissão de conteúdo multimídia em tempo real de um nó da rede para um outro, ou para um conjunto de nós. Por exemplo, sites *online* como o *lives-*

tream.tv, *ustream.tv* e *streamtheworld.com* permitem que um usuário transmita conteúdos multimídia do seu computador para milhares de outros usuários conectados à Internet;


- Aplicações de telefonia IP, tais como o Skype, principalmente considerando o modo de conversa em grupo;
- TV *Online*, por exemplo, transmissões através da Internet de jogos da copa do mundo ou do campeonato brasileiro de futebol. A rede Globo de Televisão já possui aplicações experimentais neste sentido; e
- Jogos e rádios *online* e videoconferência 1-para-muitos, típicos na Internet.



Do ponto de vista de pesquisa acadêmica, existem alguns desafios para o desenvolvimento de aplicações e engenharia de rede nesses cenários, tais como: (i) permitir que fluxos multimídia convivam com fluxos de dados de aplicações elásticas [?] sem que estes últimos sejam degradados pelos primeiros – vasta utilização do protocolo TCP; e (ii) evitar perdas excessivas de dados por parte das aplicações multimídia em questão, pois, neste caso, não faz sentido retransmitir dados quando estes são perdidos devido ao comportamento transiente dos fluxos de dados desse tipo de aplicação.

Para viabilizar a utilização de cenários como os apresentados anteriormente, deve-se abordar problemas relacionados ao gerenciamento de conexão multi-ponto, incluindo suporte à conexões partindo de diferentes redes e entre uma fonte transmissora e múltiplos receptores interessados no fluxo multimídia; controle de congestionamento na rede, incluindo aspectos de compartilhamento equinime do canal entre os diversos fluxos de dados e de forma padronizada; adaptação de fluxo multimídia e qualidade de serviço; segurança; dentre outros.

É nesse contexto de desenvolvimento de soluções de software para viabilizar transmissão de conteúdos multimídia da classe de aplicações de rede $1 \rightarrow N$ que se insere esse trabalho, motivado pelo grande interesse de pesquisa, indústria e mercado em evoluir o estado da arte desse recente modo de disponibilização de conteúdos multimídia observado principalmente na Internet.

1.1 Panorama Atual: Protocolos de Transporte da Internet

Atualmente, as principais propostas de protocolos de transporte de dados para Internet são o TCP, o UDP e o recém padronizado, DCCP [9; 10]. Quando se projeta um protocolo de rede para transporte de dados nos cenários discutidos anteriormente, deve-se levar em consideração as características da aplicação para que se possa obter resultados significativos na qualidade do fluxo de dados multimídia sendo transmitido e na eficiente utilização dos recursos da rede. Porém, nota-se que os protocolos da camada de transporte existentes são limitados com vista aos cenários de distribuição de conteúdo 1 → N  envolvendo suporte a controle de congestionamento.

O protocolo UDP tem sido largamente utilizado em aplicações multimídia em tempo real por ser um protocolo en , fazendo uso apenas do serviço de melhor esforço do IP para transmitir dados na Internet. Com o passar dos anos e antes do DCCP, o UDP se tornou a primeira e única opção para transmissão de dados multimídia em tempo real, porém gerando diversos efeitos colaterais nas grandes redes, os quais são vastamente discutidos na literatura [4; 3; 2; ?; ?; .

Para se ter uma idéia dos efeitos colaterais gerados na rede com o uso do UDP, observe o gráfico *vazão × tempo* ilustrado na Figura 1.1. Este gráfico corresponde a um experimento realizado com a transmissão de 1 fluxo TCP competindo com 3 fluxos de áudio UDP em uma rede *Ethernet* de 100 *Mbps*. Observe que o UDP sempre ocupa o máximo da largura de banda disponível na rede ao passo que não oferece chances para outros fluxos utilizarem o canal, como é o caso do TCP. Por este motivo, o UDP sempre se apresenta com altas taxas de perda de pacotes, sobretudo quando há congestionamento na rede. No caso deste experimento, nos primeiros 50 *s*, quando não disputava com nenhum outro fluxo, o fluxo TCP utilizou a rede de forma satisfatória, alcançando uma vazão em torno de 20 *Mbps*. Entretanto, após esta fase, quando os três fluxos UDP foram transmitidos na rede, a vazão do fluxo TCP reduziu praticamente para 0, permanecendo assim até o final do experimento.

O protocolo TCP, por sua vez, atende de forma satisfatória as aplicações que toleram atrasos na entrega de dados e que exigem que estes sejam todos entregues corretamente e em ordem (aplicações elásticas). Porém, em se tratando de transporte de dados multimídia em tempo real, o TCP se torna o protocolo menos apropriado para este fim, pelo menos

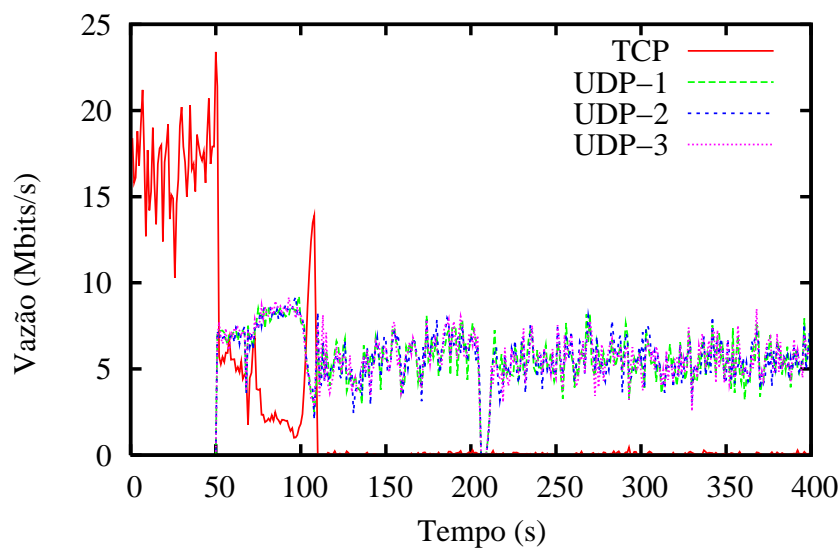


Figura 1.1: TCP \times UDP. Após 50 s do início do experimento, o fluxo UDP ocupa toda a largura de banda disponível na rede.

comparando-o com o UDP e o DCCP. Nas aplicações de fluxo multimídia em tempo real, é preferível manter o fluxo de dados e reproduzir o conteúdo que chega a esperar que a informação perdida seja retransmitida, mesmo diante do fato de que parte dos dados da aplicação tenha sido perdida. Ao utilizar o TCP, isto não é possível. O principal motivo é que o TCP implementa entrega confiável de dados adotando a abordagem de retransmitir qualquer pacote perdido. Esta estratégia acarreta em atrasos indesejáveis quando se trata de transmissão de dados multimídia em tempo real.

Em condições de congestionamento na rede, o atraso fim-a-fim aumenta e conseqüentemente degrada a qualidade do conteúdo multimídia sendo transmitido. Esta situação se agrava com a retransmissão de pacotes perdidos e que podem não fazer mais sentido à aplicação receptora devido ao comportamento transiente dos fluxos multimídia, em particular os transmitidos em tempo real. Neste caso, se os pacotes de dados retransmitidos não alcançarem o receptor até um determinado instante, estes serão descartados ao preço do desperdício no uso dos recursos da rede, pois *buffers* dos roteadores são alocados para processar e repassar pacotes que, nestes casos, terminam sendo inúteis às aplicações.

O comportamento do TCP para situações como a mencionada anteriormente pode ser observado no gráfico ilustrado na Figura 1.2. No experimento realizado, transmitiu-se um áudio com duração de 100 s, armazenado no destino e em seguida comparado com o original.

Neste caso, constatou-se que apenas 32 % do áudio alcançou o destino, fato ocorrido devido ao excesso de retransmissões de pacotes que foram perdidos na rede quando considerados fluxos TCP disputando com fluxos UDP.

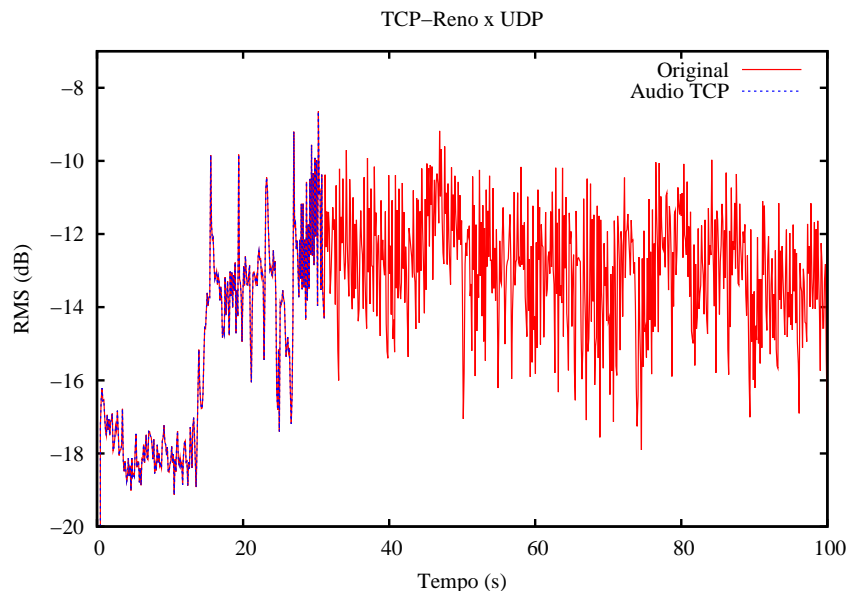


Figura 1.2: TCP Reno \times UDP, sendo o TCP enviando um arquivo de áudio.

Com apenas essas duas opções para transporte de dados na Internet e objetivando promover melhorias nos serviços oferecidos pelas aplicações multimídia, a IETF aprovou a especificação do protocolo DCCP para transporte 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. Todavia, o DCCP implementa controle de congestionamento para transmissão não-confiável de fluxo de dados [3].

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. Além destas características, o DCCP também adiciona dois conceitos novos: a escolha tardia de dados e um arcabouço para gerenciamento dos algoritmos de controle de congestionamento de forma modular. A escolha tardia de dados permite a mudança de dados de um pacote mesmo depois que estes dados já tenham sido

enviados para a camada de transporte, mas ainda não tenham sido enviados através da rede - isto é uma alternativa ao mecanismo de retransmissão do TCP. Já o arcabouço de **gerenciamento** de algoritmos de controle de congestionamento 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 1.3. Neste gráfico, ilustram-se os comportamentos dos protocolos TCP e DCCP quando utilizados para transmissão de um arquivo e de um conteúdo multimídia, respectivamente. A partir do gráfico, é possível constatar que os protocolos TCP e DCCP compartilham entre si a largura de banda disponível, onde cada fluxo consegue transmitir dados na rede. Note que o comportamento do protocolo TCP para os 50 s iniciais foi similar ao confronto TCP × UDP (Figura 1.1). Porém, diferentemente do que ocorreu naquele caso, após os primeiros 50 s dos confrontos TCP × DCCP, a vazão do protocolo TCP continuou sendo satisfatória.

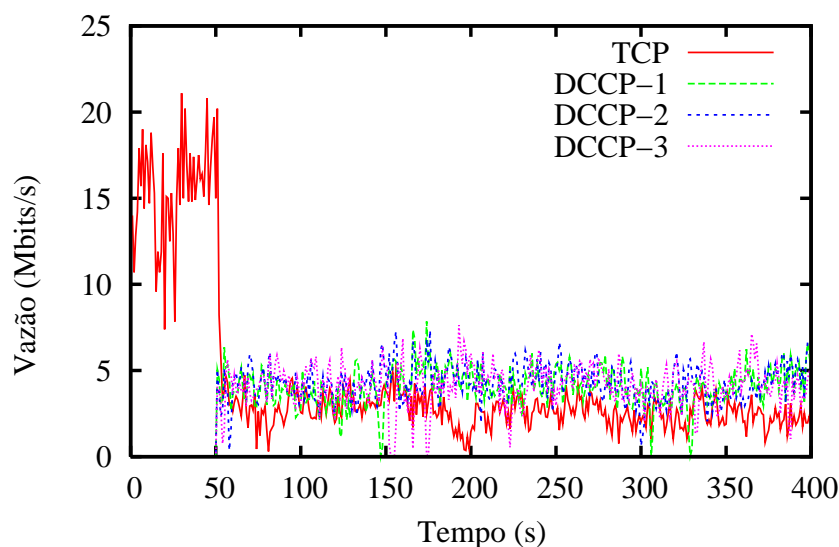


Figura 1.3: TCP × DCCP. Ambos os protocolos conseguem transmitir dados na rede.

Para transmissões de dados multimídia em redes de computadores, onde satisfazer os requisitos de tempo pode definir o nível de qualidade da transmissão multimídia, o DCCP pode melhorar a qualidade do fluxo multimídia e ainda resolver diversos problemas de congestionamento da rede, como os causados por retransmissões desnecessárias de pacotes feitas pelo protocolo TCP ou por problemas de excessiva perda de pacotes quando se utiliza o protocolo

UDP.



Sendo assim, a motivação do DCCP 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 utiliza o UDP. Considerando os problemas e limitações dos protocolos TCP e UDP discutidos até aqui e os cenários de aplicações considerados neste trabalho, o DCCP surgiu como uma das possíveis soluções para o seguinte problema-chave: de um lado apresenta-se o protocolo UDP, adotado por diversas aplicações com restrição de tempo e que não realizam controle de congestionamento. Por outro lado, apresenta-se 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.

Estudos anteriores realizados no contexto deste trabalho [?; ?; ?] e outros publicados por terceiros [?; ?; ?; ?; ?; ?] constataam que a utilização do protocolo DCCP tem trazido diversas vantagens na transmissão de fluxos multimídia e se apresenta como uma opção efetiva a ser adotada para transmissão de dados multimídia, principalmente por ter um comportamento similar ao protocolo TCP e por ser um protocolo padronizado pela IETF.

1.2 Descrição do Problema

Apesar da eficiência do DCCP em alguns cenários de transmissão de dados multimídia na Internet, o mesmo possui falhas críticas quando utilizado em larga escala em cenários $1 \rightarrow N$. Além disso, ao longo dos anos o uso de outros protocolos de transporte para a Internet tem se mostrado pouco efetivo na prática para a distribuição de conteúdos multimídia em tempo real, tal como o uso do UDP ou diversas outras propostas não padronizadas [?; ?; ?; ?; ?], acentuando a motivação em se desenvolver pesquisas nesse contexto. Isto tem ocorrido porque as soluções existentes sempre se apresentam de forma independente uma das outras e sem qualquer preocupação com sua facilidade de implantação em larga escala.

O DCCP é um protocolo orientado à conexão e portanto para cada novo usuário interessado em receber um fluxo multimídia transmitido com DCCP uma nova conexão se faz necessária. As conseqüências desta limitação do DCCP são desastrosas, tornando-o um protocolo paradoxal para o que se propõe: resolver os problemas de congestionamento de rede gerados pelo protocolo UDP em cenários de aplicações multimídia, porém, quando o proto-



colo é utilizado em larga escala, o protocolo não funciona devido aos seguintes motivos:

1. **Excessivo consumo de recurso computacional:** para cada nova conexão, o nó transmissor deve alocar recursos computacionais (memória e processamento) para tratar cada nova conexão. Em cenários de transmissão multimídia $1 \rightarrow N$, se muitos nós estão conectados em um único servidor, então isto elevará sobremaneira o consumo de recurso computacional do nó transmissor proporcionalmente à quantidade de nós receptores interessados pelo fluxo multimídia transmitido por um único nó na rede. Além disso, embora o conteúdo transmitido por um nó seja de interesse de muitos outros nós, os fluxos são enviados independentemente uns dos outros, o que gera duplicações desnecessárias e conseqüentemente desperdício de recursos de rede.
2. **A taxa de transmissão individualmente tenderá a 0:** O protocolo DCCP realiza controle de congestionamento utilizando uma equação matemática para definir a taxa de transmissão de uma conexão. À medida que mais nós se conectam a um nó transmissor, menor será a taxa de transmissão do nó transmissor para cada um dos nós receptores conectados a ele. Para a rede, esta estratégia é justa e evita que a mesma entre em colapso de congestionamento, mas para cada fluxo de dados isto é ruim. Este fenómeno é vastamente descrito na literatura e denominado de *tragédia dos comuns* [6], apresentando-se em diferentes áreas do conhecimento. No caso deste trabalho, a tragédia dos comuns ocorre porque à medida que novos fluxos são transmitidos na rede, menor será a taxa de transmissão individual de cada fluxo, a qual pode se tornar insuficiente para a recepção de um fluxo multimídia, embora todos os fluxos terão direitos iguais sobre o uso do canal, o que é justo.

Sendo assim, apesar dos algoritmos de controle de congestionamento serem corretos visando o caso do melhor global (equidade para com todos os fluxos e assim evitar congestionamento da rede), isto provoca o efeito do caso do pior local (redução da taxa de recepção de cada nó da rede), causando um problema crítico para os sistemas finais: a taxa de transmissão individualmente tende a não ser suficiente em aplicações multimídia e nenhum nó receptor conseguirá reproduzir o fluxo transmitido pelo nó gerador do conteúdo multimídia em questão.

Este fato pode ser explicado matematicamente utilizando como base a Equação 1.1, que define cada taxa de transmissão X_i calculada pelo DCCP¹. Nesta equação, X_i é a taxa de transmissão em bytes/segundo, s é o tamanho do pacote em bytes, R é o R_{TTT} em segundos, p é a taxa de ocorrência de perdas, entre 0 e 1, RTO é o valor do *timeout* de retransmissão do TCP em segundos e b é igual a 1 e representa o número máximo de pacotes confirmados por um único ACK.

~~Considere o problema descrito anteriormente e que o uso total do canal para uma quantidade~~ N de fluxos DCCP é definido por $B = \sum_{i=1}^N X_i$. Em condições severas de congestionamento na rede, o valor de B é equivalente à largura de banda do canal de transmissão. Quando isto ocorre, tem-se que atingiu um valor maior do que a rede suporta, fazendo com que os *buffers* de recepção dos roteadores alcanssem seus limites e portanto os valores de p e R na Equação 1.1 também aumentam, resultando que o $\lim_{N \rightarrow \infty} \frac{B}{N} = 0$, logo, X_i se aproxima de 0 (zero).

$$X_i = \frac{s}{R \times \sqrt{2 \times b \times \frac{p}{3}} + (RTO \times 3 \sqrt{3 \times b \times \frac{p}{8}} \times p \times (1 + 32 \times p^2))} \quad (1.1)$$

Embora esta seja uma discussão teórica sobre o problema tratado neste trabalho, também foram realizadas pesquisas em busca de evidências mais contundentes de que este fato ocorre na prática. Foram executadas simulações de rede no NS-2 cuja topologia foi definida como uma árvore binária, onde cada ramo da árvore representava um roteador e cada roteador tinha 10 nós DCCP conectados a ele (Figura 1.4). Em cada experimento, aumentava-se em 1 o nível da árvore binária que definia a topologia da rede. Por exemplo, o primeiro cenário tinha 10 nós receptores e 1 roteador, no cenário seguinte 30 nós e três roteadores, no seguinte 70 nós e 7 roteadores e assim por diante. A transmissão ocorreu da seguinte forma: um nó localizado na raiz da árvore transmitiu o mesmo conteúdo multimídia para todos os outros nós conectados à rede, simulando uma típica transmissão multimídia $1 \rightarrow N$ e um tráfego de comportamento equivalente a VoIP. Cada um dos 10 experimentos foram repetidos a quantidade de vezes necessária até se alcançar uma média com nível de confiança

¹Essa equação é utilizada pelo algoritmo de controle de congestionamento CCID-3 do DCCP. Para efeito de estudo esta equação foi a escolhida, porém qualquer outra equação para controle de congestionamento poderia ter sido utilizada.

de 95%.

Os resultados obtidos com as simulações descritas anteriormente estão ilustrados no gráfico da Figura 1.5. O eixo X do gráfico representa o número de nós receptores à medida que as simulações eram executadas, ao passo que o eixo Y é a taxa de transmissão média conseguida por cada conexão DCCP.

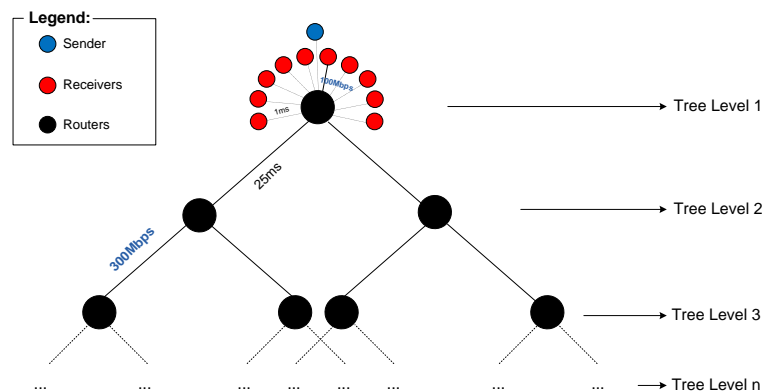


Figura 1.4: Topologia da rede definida para as simulações realizadas. Cada rede é representada por um roteador e com 10 nós em cada rede.

É possível observar no gráfico da Figura 1.5 que a vazão média de cada fluxo DCCP transmitido aos receptores tende a zero à medida que o número de receptores aumenta, sendo possível concluir que o protocolo DCCP não escala quando utilizado para transmissão de dados multimídia em cenários de aplicações com um transmissor transmitindo para vários receptores ($1 \rightarrow N$).

Uma questão intrigante neste aspecto é que o protocolo DCCP funciona perfeitamente em cenários simplórios, mas sofre claramente de um problema de escalabilidade, o que é crítico para aplicações consideradas neste trabalho, as quais não podem continuar utilizando protocolos como o UDP pelos efeitos colaterais causados por este. Isto torna o protocolo DCCP inútil para cenários de distribuição de conteúdo multimídia, fazendo com que os desenvolvedores continuem sem motivações para efetivamente utilizar o protocolo DCCP em suas aplicações.

Note que apenas fluxos DCCP foram transmitidos nesta simulação, os quais já foram necessários para causar o problema em questão. Em situações mais realistas, o problema se torna ainda mais grave, pois o protocolo DCCP disputará o canal não apenas com outros

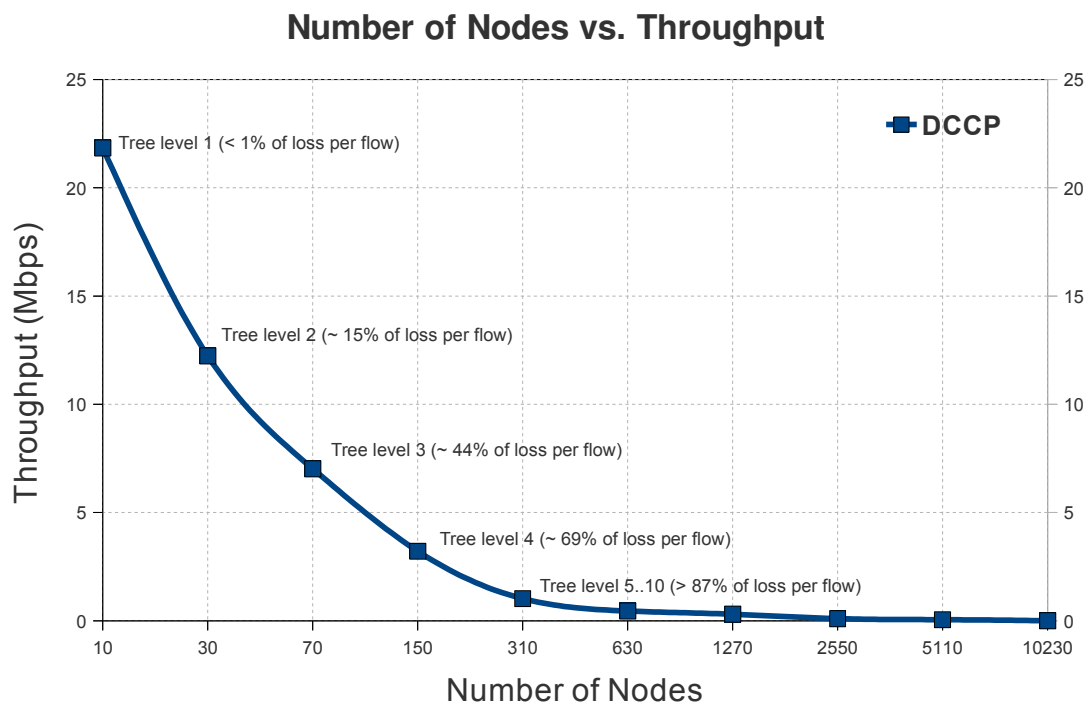



Figura 1.5: Gráfico de uma transmissão DCCP com um transmissor enviando dados de áudio VoIP utilizando o protocolo DCCP para X receptores. A vazão média de cada fluxo tende a zero à medida que o número de receptores aumenta.

fluxos DCCP, mas também com fluxos de protocolos tradicionais, como o TCP e UDP, além de outros protocolos modernos, como o SCTP [?; ?; ?].

Apesar do protocolo DCCP ter sido utilizado para evidenciar o problema trazido à tona neste trabalho e desta forma apresentar resultados concretos, esta discussão pode ser generalizada em direção a qualquer outro protocolo de rede que seja orientado a conexão e que suporte mecanismos para controle de congestionamento de fluxo não confiável de dados. A fim de aumentar a representatividade de protocolos nesse contexto, o leitor pode consultar mais referências nos trabalhos disponíveis em [?; ?; ?; ?; ?; ?; ?; ?] e no Capítulo 3 deste documento.

Neste contexto, os desenvolvedores de aplicações multimídia não tem outra opção a não ser continuar utilizando o protocolo UDP, porém ao custo do que já foi discutido anteriormente além disso, o TCP não é aplicável neste cenário por diversas questões também já discutidas.

Diante do exposto e buscando melhorar a qualidade dos fluxos de dados multimídia transmitidos nas redes de computadores, em particular em larga escala na Internet, pretende-se responder a seguinte questão de pesquisa: 

- Como transportar de forma padronizada dados multimídia para distribuição de conteúdos em topologias de rede $1 \rightarrow N$ com suporte a controle de congestionamento, mas evitando o fenômeno da tragédia dos comuns?


Observando esta questão que deu origem  às pesquisas desenvolvidas neste trabalho, propõe-se a seguinte tese:

Só é possível permitir a distribuição de conteúdos multimídia de forma eficiente e padronizada em cenários $1 \rightarrow N$ através de um protocolo de transporte com controle de congestionamento capaz de combinar o modo de transmissão multicast com o compartilhamento de múltiplos fluxos unicast entre os nós de uma mesma rede.

1.3 Objetivos da Tese

1.3.1 Objetivo Principal

Neste trabalho, tem-se como objetivo principal a concepção e o desenvolvimento de um protocolo de rede de computadores para transporte de fluxos de dados controlados e não confiáveis para a distribuição de conteúdo multimídia em tempo real em cenários $1 \rightarrow N$. Mais especificamente, propõe-se a especificação e o desenvolvimento de um protocolo para a classe de aplicações apresentadas, validando-o através de experimentações e simulações com um conjunto de ferramentas para dar suporte ao desenvolvimento de aplicações nesse contexto, assim como promovendo ferramentas para futuras investigações em pesquisa sobre o protocolo estudado, considerando os seguintes requisitos de sucesso ao fim deste trabalho.

- **Transparência para o desenvolvedor** –  Deve-se prover um protocolo de rede que permita ao desenvolvedor, apenas através da utilização deste protocolo, desenvolver aplicações para envio de dados multimídia em cenários $1 \rightarrow N$ com suporte a controle de congestionamento de fluxo de dados não confiável. Em momento algum, o desenvolvedor deverá implementar mecanismos para controle de congestionamento no

processo de desenvolvimento de sua aplicação. Os mecanismos para controle de congestionamento, assim como o de distribuição de conteúdo, devem ser transparentes para o desenvolvedor.

- **Independência de linguagem e plataforma – O protocolo proposto** deve ser independente de linguagem de programação ou plataforma específica para a sua execução. Sendo assim, espera-se que seja possível utilizar tal protocolo em diferentes contextos de aplicações, considerando diferentes linguagens de programação, incluindo linguagens populares como C, C++, Java e Python; assim como que seja possível implementá-lo em diferentes sistemas operacionais.
- **Comprometimento com práticas já utilizadas em aplicações multimídia – É** imprescindível que o protocolo proposto seja capaz de transmitir dados em modo *multicast* e que suporte algoritmos para controle de congestionamento neste modo, considerando a restrição de que este serviço seja implementado na camada de transporte, e não na camada de aplicação.
- **Monitoramento e notificação de requisitos mínimos para transmissão – O** protocolo deve permitir que as aplicações definam o mínimo de taxa de transmissão necessária para transmissão de dados e notifiá-las caso os limiares definidos sejam atingidos. Requisitos como taxa mínima para transmissão e recepção e atraso máximo figuram como exemplos desses requisitos. Isto permitirá que as aplicações construam soluções para adaptação e transcodificação em tempo real de fluxo de dados multimídia.
- **Especificação do protocolo para uso em larga escala – O protocolo** deve ser descrito e publicado sob domínio público seguindo o formato determinado pela IETF (*Internet Engineering Task Force*) e, desta forma, facilitar a utilização do protocolo em escala global.

1.3.2 Objetivos Específicos

Considerando os requisitos descritos anteriormente, pode-se dividir o objetivo principal deste trabalho nos objetivos específicos descritos a seguir.



1.4 Relevância do Tema e da Tese

Transporte de fluxos de dados multimídia com suporte a controle de congestionamento para distribuição de conteúdos multimídia, em particular, na Internet, é um tema relevante no contexto de redes de computadores devido a melhoria na qualidade de conteúdo sendo transmitido e no uso eficiente dos canais de transmissão. Esta foi a motivação para a concepção do protocolo de rede proposto neste trabalho, principalmente com vista a sua ampla utilização na Internet.

No caso do tema específico tratado neste trabalho, a distribuição de conteúdos multimídia em larga escala é cada vez mais relevante devido às características inerentes à classe de aplicações que têm sido disponibilizadas na Internet, tais como controle de congestionamento e distribuição de conteúdo. Estas características, aliadas à padronização e a transparência da solução no ponto de vista do desenvolvedor da aplicação, tornam o tema ainda mais relevante para o contexto de boas práticas para a distribuição de conteúdos multimídia.

Como a demanda por serviços multimídia em redes de computadores tem aumentado dia após dia, principalmente em cenários de aplicações consideradas neste trabalho, o estudo desenvolvido e os artefatos de software providos por esta tese contribui para o desenvolvimento de aplicações multimídia mais eficientes e de forma padronizada, além de facilitar a tomadas de decisões sobre futuros desenvolvimentos desse tipo de aplicação. Isto é possível porque neste trabalho são apresentados um protocolo para distribuição de conteúdo multimídia em tempo real com suporte a controle de congestionamento e 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. Uma das primeiras publicações realizadas no contexto de apresentar o desempenho do protocolo DCCP foram feitas no contexto deste trabalho e a solução do uso do DCCP para distribuição de conteúdos multimídia aparece como a primeira a tratar do problema trazido à tona neste trabalho.

No que diz respeito à relevância do trabalho, em se tratando de uma tese em Engenharia para Redes de Computadores, o autor considera indispensáveis três principais requisitos que estão sendo contemplados e que reforçam a relevância da tese. Estes requisitos servem como motivação para a realização das atividades desenvolvidas até o presente momento.

O primeiro deles é a consistência teórica. O protocolo de rede proposto foi conce-

bido a partir de evidências sólidas com base em experimentos e simulações de rede, com o problema-chave apresentado e discutido através de fundamentos matemáticos. O protocolo proposto tendo sido descrito de forma rigorosa e não-ambígua, permitindo um melhor entendimento e futuros investimentos no protocolo teórico proposto.

O segundo requisito é a contribuição científica. Diversos trabalhos relacionados foram estudados antes da concepção do protocolo proposto. A partir deste estudo, identificou-se o problema anunciado anteriormente e, a partir de então, foram elencadas as possíveis soluções para o problema, o que culminou com a definição deste trabalho. Até o momento da escrita deste documento e considerando que este é um trabalho ainda em desenvolvimento, não foram encontrados trabalhos com as características aqui propostas, o que reforça o caráter de originalidade e contribuição científica, a qual já vem sendo respaldada pela comunidade através da publicação de vários artigos em veículos relevantes da área.

O terceiro requisito é o potencial prático. A implementação do protocolo de rede, assim como o conjunto de ferramentas que tem sido desenvolvida no contexto deste trabalho, tem como objetivo demonstrar que a abordagem é viável e praticável. Um protocolo de rede simplesmente especificado sem nenhuma implementação real tornaria as reais contribuições deste trabalho apenas suposições. O compromisso com a utilização dos conceitos para construir mecanismos que possam ser aplicados na indústria tornam o trabalho relevante em termos práticos.

1.5 Resumo das Contribuições até o Momento

Com base nos resultados até o momento no contexto deste trabalho, vários trabalhos relacionados foram desenvolvidos, tanto no nível de pesquisa [?; ?; ?; ?; ?] quanto no nível de desenvolvimento [?; ?; ?].

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, sendo essas melhorias já submetidas ao grupo de desenvolvimento de DCCP em Linux durante todo o tempo de desenvolvimento deste trabalho. Para essa última contribuição, 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 tese está acontecendo no contexto de desenvolvimento de aplicações multimídia para sistemas embarcados. Este projeto está sendo desenvolvido nas dependências do Laboratório de Sistemas Embarcados e Computação Pervasiva, localizado na Universidade Federal de Campina Grande (UFCG).

1.6 Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

- No Capítulo 2.1 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 ?? ...
- No Capítulo ?? ...

²O conceito de *socket* será discutido no Capítulo 2.

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

Capítulo 2

Fundamentação

“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)

2.1 Camada de Transporte TCP/IP

Como já discutido no Capítulo ??, os recentes avanços nas tecnologias de hardware 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 consequência 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.

2.2 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 [?].

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 [?].

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.

2.3 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.

Camada de Aplicação	HTTP, HTTPS, SMTP, FTP, UUCP, NNTP, SSH, IRC, SNMP, SIP, RTP, Telnet, ...	Mensagem
Camada de Transporte	TCP, UDP, SCTP, DCCP , ...	Pacote
Camada de Rede	IPv4, IPv6, ICMP, ARP, IGMP, ...	Datagrama
Camada de Enlace	Ethernet, Wi-Fi, Token ring, FDDI, PPP, ..	Dado
Camada Física	RS-232, EIA-422, RS-449, EIA-485...	

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

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 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 1.2.

No caso do protocolo DCCP, isto é resolvido implementando controle de congestionamento

mento 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 sequência 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.

2.4 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 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* [?]. 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

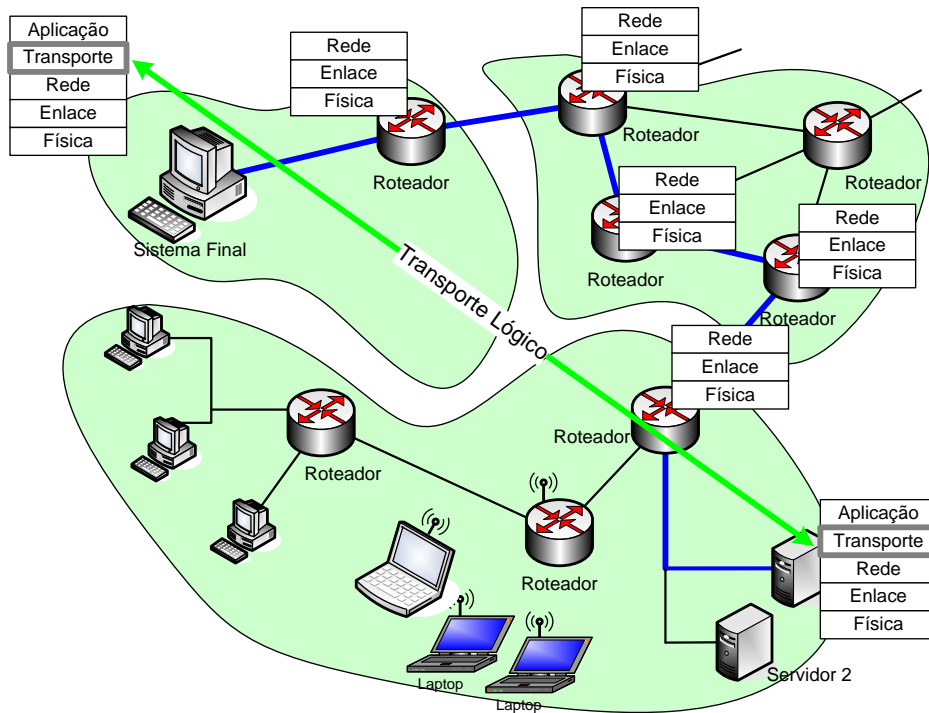


Figura 2.2: Comunicação lógica entre as camadas de transporte dos sistemas finais. Adaptado de [?].

algumas informações de controle (cabeçalho da camada de transporte) e então repassa o 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.

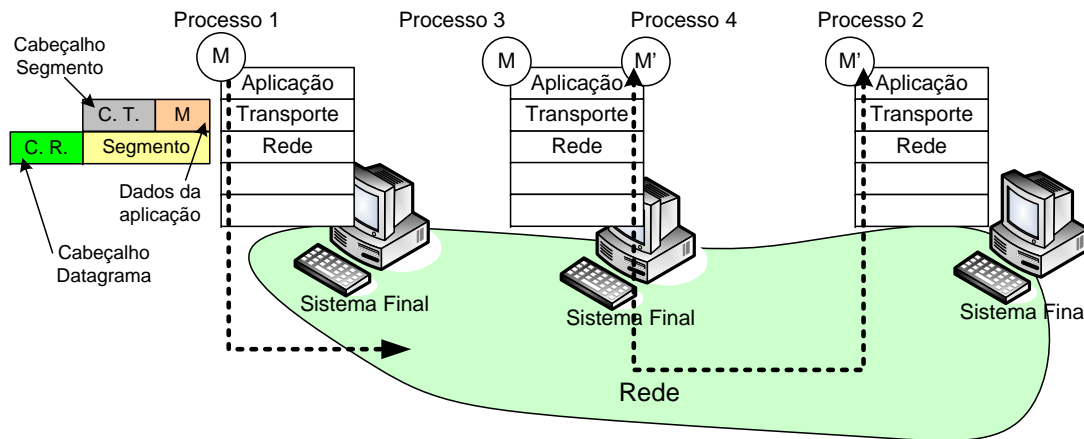


Figura 2.3: Processo de adição de cabeçalhos aos pacotes à medida que estes passam pelas diversas camadas da rede. Adaptado de [?]

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.

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.

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

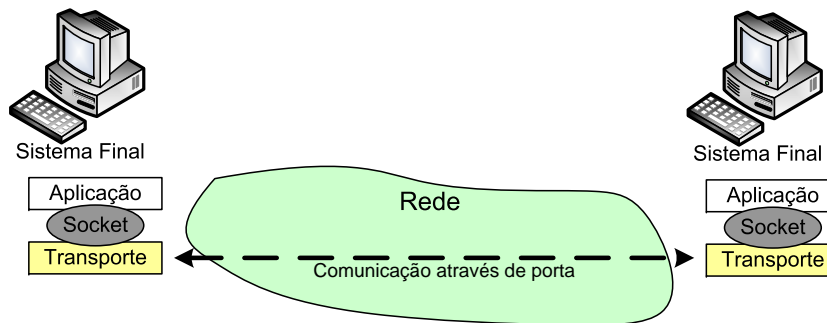


Figura 2.4: Comunicação entre a camada de transporte e a camada de aplicação através de um *socket*. Adaptado de [?]

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 provenientes 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.

2.4.1 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.

2.4.2 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 2.2. 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.

2.5 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 [?] (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.

2.5.1 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 [?]. 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**.

2.5.2 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 receptor [?]. 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 transmis-

sor 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.

2.5.3 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*) [?; ?]. 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 Experienced*). 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 [?]. O mecanismo de ECN para IP está especificado

no RFC 3168 [?] e tanto o TCP quanto o DCCP suportam esse mecanismo.

O ECN é um dos pontos discutidos em [4], 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.

rede, ele passa por uma série de roteadores e até chegar no sistema de destino. Nesse processo, um pacote sofre uma série de atrasos, são eles: acontecem para cada nó da rede, entre o sistema de origem e o de destino. Esse atraso é gerado devido ao tempo necessário para examinar o cabeçalho do pacote e determinar para onde roteá-lo, assim como o tempo necessário para verificação de possíveis erros no pacote; espera para ser transmitido no meio físico da rede. Esse tempo dependerá da quantidade de outros pacotes que chegarem antes dele nos roteadores e que já estiverem na fila esperando para serem roteados; armazenamento e reenvio, é o tempo gasto para transmitir um pacote de tamanho L bits a uma velocidade R bits/s do meio físico. Ou seja, é a razão L/R ; lançado no meio físico para alcançar um outro nó na rede. sistema de origem ao sistema de destino.

2.6 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) [?]. O protocolo UDP não apresenta suporte a essas características e por isso a IETF publicou a RFC 3489 [?], conhecida por STUN (*Simple Traversal of UDP over NAT*). O STUN é uma solução paliativa 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** [?]. 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 2.6.2 e 2.6.3, respectivamente.

O protocolo DCCP é especificado pela IETF e sua especificação é dividida basicamente em 4 principais RFCs, além de diversos *Internet Drafts*¹ não mencionados neste trabalho. A primeira especificação é a RFC 4336 [4], 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 [9] 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 [8] e 4342 [8], 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 2.6.3, 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.

2.6.1 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;
- 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*);

¹Internet 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.

- 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*) [?], o que ajuda a evitar fragmentação na camada IP.

2.6.2 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) [?].

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.

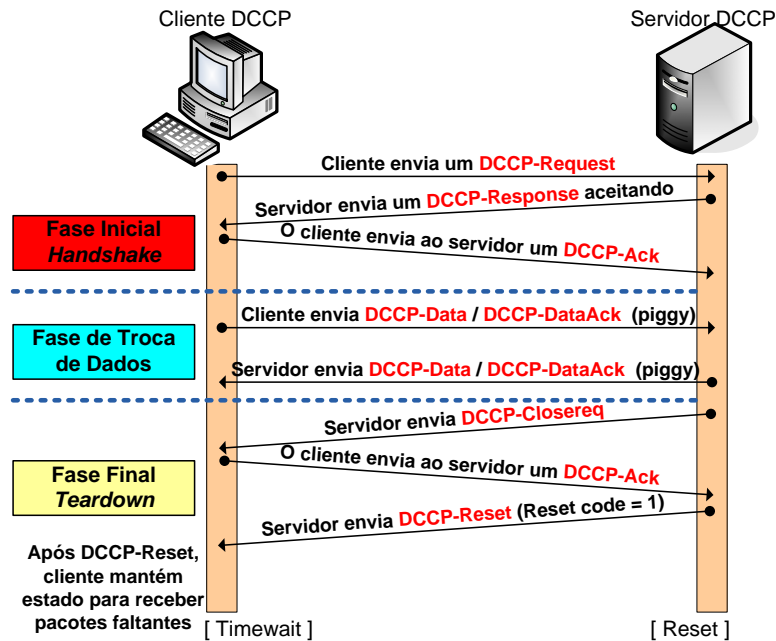


Figura 2.5: Ciclo de vida de uma conexão DCCP.

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*.

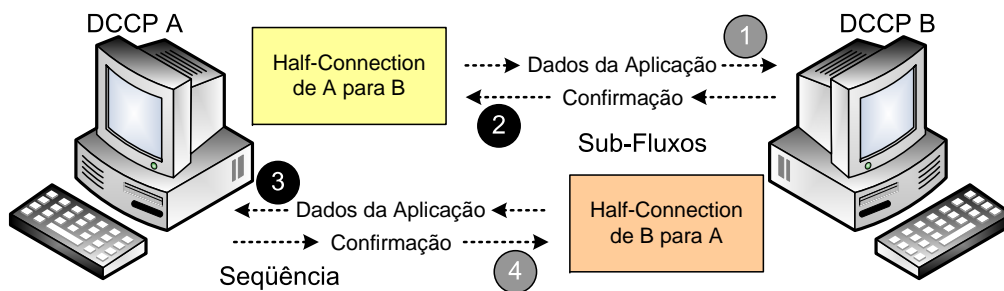


Figura 2.6: Conexão bi-direcional do protocolo DCCP.

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 2.6.2 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 [?]. 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 2.5, 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 [?; ?], 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 [?].

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 sequência estendido (*Extended Sequence Numbers bit*). Se o valor de X for 1, o campo Número de Sequência tem o tamanho de 48 bits e o cabeçalho

genérico fica com o tamanho de 16 bytes.

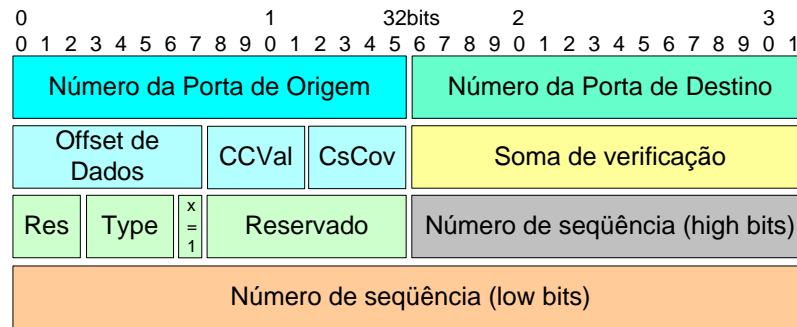


Figura 2.7: Cabeçalho do protocolo DCCP (estendido).

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.

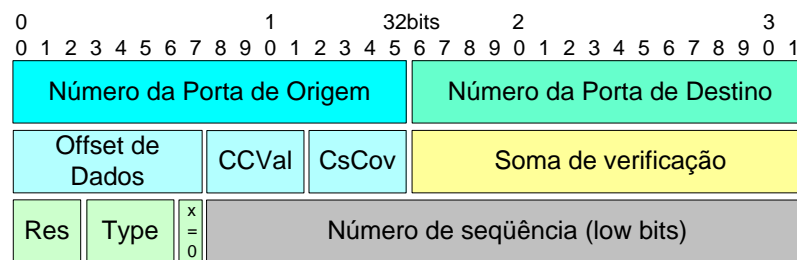


Figura 2.8: Cabeçalho do protocolo DCCP (simplificado).

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 2.6.2;

Número de seqüência estendido (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 exceção 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 [9].

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.

2.6.3 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 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

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.

janela) ou como e com qual frequência 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 2.6.2. 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) [8] e o *TCP Friendly Rate Control* (ou CCID-3) [7]. 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 [8] 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*) [?] 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. (*ack*). O TCP controla apenas pacotes de dados. «REVER

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

O CCID-3 [7] 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*.

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

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 disponível 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 [?], 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) [?]. 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.

2.6.4 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 [?]. Este valor³ 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.

Código Fonte 2.1: Conexão DCCP Através da API de Socket Berkeley

```
1 #include <arpa/inet.h>
2 #include <errno.h>
3 #define IPPROTO_DCCP 33
4
5 (...)
6 int result = 0;
7 if ((sock_fd = socket (AF_INET, SOCK_DCCP, IPPROTO_DCCP)) > 0) {
8     int result = connect(sock_fd, (struct sockaddr *)&serverSockIn,
9                           sizeof(serverSockIn));
10    return result;
11 } else {
12     printf("Conexão Recusada.");
13 }
14 (...)
```

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.

³Outros 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>

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

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 [?], 1122 [?], 1323 [?], 2018 [?], 2581 [?] e 3390 [?].

Como já foi dito, o foco deste trabalho está no protocolo DCCP, e portanto, diferentemente da nossa discussão a respeito do DCCP na Seção 2.6, 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 [?] e [?] 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, conside-

rando os mesmos princípios utilizados pelo TCP e apresentados nesta seção.

2.7.3 Controle de Congestionamento do TCP

Como discutido na Seção 2.5, 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 2.5.1, 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 retransmitindo os segmentos perdidos.

Para evitar este colapso de congestionamento o protocolo TCP reduz a taxa de transmissão quando o congestionamento é detectado, mas continua retransmitindo 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* (início 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 [?]. Isto significa que a taxa de transmissão inicial do TCP é de 1 segmento a cada RTT (*Round Trip Time*) [?]. 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 aumentar linearmente, da mesma forma como explicado anteriormente.

Um dos primeiros algoritmos de controle de congestionamento do TCP (conhecido também por Tahoe), diminui incondicionalmente 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 [?].

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 [?; ?] é baseado no algoritmo TCP BIC [?]. 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 da referência [?].

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

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 alteração. 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 [?], discute-se em mais detalhes o funcionamento do controle de congestionamento TCP Cubic.

TCP Veno

O TCP Veno [?] é um algoritmo para controle de congestionamento baseado no TCP Vegas [?] 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 [?; ?], nas redes sem fio as perdas de pacotes por este tipo de erro ocorrem com mais frequência. 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 [?; ?]. 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 diminui para evitar congestionamento. No Reno, o valor de *sshresh* é igual a 85.3 KB⁵. 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 conseqüentemente 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*

⁵Valor referente a implementação do TCP Reno no Linux

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*.

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 [?], 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 2.4. O protocolo UDP está definido no RFC 768 [?].

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 [?], 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 [?], onde 9 dos 13 servidores DNS (*Domain Name Server*) raiz da Internet entraram em colapso por uma hora, devido ao excessivo número de requisições realizadas 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 [?] e [?].

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.

O primeiro deles o tamanho do cabeçalho de cada pacote, onde como foi explicado na Seção 2.6.2 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 2.6.2. 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 discutidos 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

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

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.

2.10 Transmissão Multicast

2.10.1 Transmissão Multicast no Escopo do Trabalho

2.11 Fundamentação X

2.11.1 X no Escopo do Trabalho

Capítulo 3

Trabalhos Relacionados

3.1 Tópicos de Comparação

A seguir são apresentados os tópicos de comparação dos trabalhos relacionados, os quais se baseiam na motivação para distribuição de conteúdo multimídia em tempo real com suporte a controle de congestionamento de fluxo de dados não confiável, como descrito no capítulo de introdução deste documento.

Tópico 1: descrição tópico 1;

Tópico 2: descrição tópico 2;

Tópico 3: descrição tópico 3;

Tópico 4: descrição tópico 4;

Tópico 5: descrição tópico 5;

Tópico 6: descrição tópico 6;

Tópico 7: descrição tópico 7;

Tópico 8: descrição tópico 8;

3.2 Descrição dos Trabalhos

A seguir são descritos os trabalhos relacionados considerados relevantes para comparação com a infra-estrutura proposta, em ordem alfabética. Tal descrição se baseia em publicações de diversas fontes (revistas, conferências, livros, teses e dissertações etc.) encontradas na literatura. Uma vez que a lista apresentada não é exustiva e considerando a potencial evolução destes trabalhos, sempre que se fizer menção à ausência de uma característica em uma dada abordagem, entenda-se que *não foram encontrados na literatura investimentos por parte dos próprios autores o trabalho ou de outros autores* para prover tal característica.

3.2.1 Trabalho 1

Tópico 1: descrição tópico 1;

Tópico 2: descrição tópico 2;

Tópico 3: descrição tópico 3;

Tópico 4: descrição tópico 4;

Tópico 5: descrição tópico 5;

Tópico 6: descrição tópico 6;

Tópico 7: descrição tópico 7;

Tópico 8: descrição tópico 8;

3.2.2 Trabalho 2

Tópico 1: descrição tópico 1;

Tópico 2: descrição tópico 2;

Tópico 3: descrição tópico 3;

Tópico 4: descrição tópico 4;

Tópico 5: descrição tópico 5;

Tópico 6: descrição tópico 6;

Tópico 7: descrição tópico 7;

Tópico 8: descrição tópico 8;

3.2.3 Trabalho 3

Tópico 1: descrição tópico 1;

Tópico 2: descrição tópico 2;

Tópico 3: descrição tópico 3;

Tópico 4: descrição tópico 4;

Tópico 5: descrição tópico 5;

Tópico 6: descrição tópico 6;

Tópico 7: descrição tópico 7;

Tópico 8: descrição tópico 8;

3.2.4 Outros Trabalhos

Além dos trabalhos citados anteriormente, vários outros foram estudados e analisados [?; ?; ?; ?; ?; ?; ?; ?], com menor nível de similaridade com o trabalho proposto. Por exemplo, vale ressaltar as tecnologias de distribuição de conteúdo conhecidas e utilizadas na indústria, tais como: X, Y, Z. Estas tecnologias têm sido utilizadas com sucesso para a aplicações multimídia, contudo tais soluções não foram concebidas para distribuição de conteúdo multimídia em larga escala e quando o fazem, não dão suporte efetivo à transmissão de conteúdo multimídia em tempo real.

3.3 Considerações Finais

Capítulo 4

Especificação do MU-DCCP

O *Multi(Uni)cast Datagram Congestion Control Protocol* (MU-DCCP) é uma variante do protocolo DCCP para transmissão de fluxos de dados multimídia em cenários com um nó transmissor e muitos nós receptores ($1 \rightarrow N$). O MU-DCCP permite a transmissão de pacotes de dados com suporte ao controle de congestionamento de fluxos não confiáveis. Ao contrário de outras soluções para distribuição de conteúdos multimídia na camada de transporte, o MU-DCCP não necessita explicitamente da instalação de um nó na rede para encaminhar o conteúdo de uma rede externa para uma rede interna (*proxy*). Além disso, o MU-DCCP mantém a *interface* de programação com a camada de aplicação inalterada, apenas adicionando uma extensão na API padrão de *socket* BSD para preservar a compatibilidade com as aplicações multimídia existentes e, ao mesmo tempo, permitir que novas aplicações façam uso dos recursos providos por este protocolo, os quais serão detalhados a seguir.

O MU-DCCP pode operar em dois modos de transmissão: (i) multicast; e (ii) multi-unicast. O modo multicast sempre é utilizado, porém quando este modo não é suportado pela rede, o modo multi-unicast do protocolo é executado. Para cada um dos modos de transmissão providos pelo MU-DCCP, a aplicação primeiramente seleciona o modo desejado e um algoritmo específico é utilizado para realizar o estabelecimento de conexão e transmissão de dados entre os receptores envolvidos. O modo multicast é utilizado geralmente em redes locais, e o modo unicast é utilizado para que um nó, localizado na rede local, estabeleça uma conexão com um nó transmissor e distribua o conteúdo na sua rede local.

4.1 Visão Geral do MU-DCCP

Quando uma aplicação inicia um *socket* DCCP correspondente a conexão desejada, a mesma envia um pacote do tipo DCCP-MREQUEST com o campo TTL da camada de rede igual a 1. Este pacote é enviado em modo multicast para o endereço 239.255.255.251 e a porta 1900 (este *socket* é chamado de canal de controle do MU-DCCP). No cabeçalho do pacote DCCP-MREQUEST existem dois campos, um para especificar o endereço IP (32bits) do nó transmissor e o outro para especificar a porta (16bits) do nó transmissor, do qual o nó receptor deseja receber o conteúdo multimídia. Como o pacote é transmitido na rede local com TTL=1, isto significa que o pacote não será roteado para a rede externa e apenas o nós da sua rede local o receberá. Para facilitar a explicação, considere-se o endereço 200.200.211.5 e porta 8900 como sendo o *socket* do nó transmissor o qual o nó receptor tem interesse de se conectar. Neste caso, o nó receptor envia uma mensagem *multicast* do pacote DCCP-MREQUEST para o endereço *multicast* especificado anteriormente e com os campos IP e porta do pacote DCCP-MREQUEST preenchidos com os dados do *socket* do nó transmissor.

Por outro lado, todos os outros nós DCCP existentes na rede local devem, ao implementarem a extensão MU-DCCP, iniciar um *socket multicast* no endereço IP 239.255.255.251 e na porta 1900. Isto permitirá a recepção de pacotes DCCP-MREQUEST transmitidos por qualquer nó na rede. Sendo assim, caso um nó na rede local já esteja recebendo um fluxo multimídia DCCP vindo de 200.200.211.5 na porta 8900, quando receber um DCCP-MREQUEST responderá para o nó interessado um pacote do tipo DCCP-MRESPONSE, o que deve ocorrer em modo *unicast*. Ao responder com o DCCP-MRESPONSE, o nó receptor do fluxo original, passa a funcionar como um nó *relay* (DCCP *Relay*), retransmitindo na rede local os pacotes que estão recebendo do nó transmissor. No cabeçalho do DCCP-MRESPONSE, o nó especificará em qual modo ele repassará os pacotes para a rede local, se no modo *multicast* ou no modo *unicast*.

Note que, caso não exista nenhum nó do tipo DCCP *Relay* na rede local, o nó interessado em receber o fluxo pode enviar um pacote do tipo DCCP-MREQUEST, porém com TTL=2. Caso o roteador da rede local esteja roteando pacotes *multicast* no endereço IP e porta do canal de controle do MU-DCCP, é possível que um DCCP *Relay* responda com um pacote do tipo DCCP-MRESPONSE da forma que foi explicado anteriormente. Caso o nó que enviar

o DCCP-MREQUEST não receba nenhum pacote do tipo DCCP-MRESPONSE, o mesmo poderá tomar duas decisões, configurável via parâmetro de configuração da aplicação (setado via *setsockopt()*, por exemplo): (1) Enviar outro pacote do tipo DCCP-MREQUEST com o valor de TTL incrementado em 1; ou (2) iniciar uma conexão *unicast* com o nó transmissor. Caso ocorra a segunda opção, o nó MU-DCCP que iniciar uma conexão *unicast* com um nó transmissor qualquer deve se auto eleger para ser um DCCP *Relay* em pedidos de conexão através do pacote DCCP-MREQUEST que este venha a receber via o canal de controle do MU-DCCP.

4.2 Modos de Transmissão do MU-DCCP

A mensagem de resposta de conexão enviada por um nó MU-DCCP, contém três campos: (i) o campo *multicast* (1 bit), se ativado o nó transmitirá os dados em modo *multicast*; (ii) o campo endereço IP, que especificará qual endereço IP (32 bits) o DCCP *Relay* passará a transmitir os dados; e (iii) o campo porta (16 bits), que especificará a porta que o DCCP *Relay* passará a transmitir os dados.

Note que um DCCP *Relay* pode decidir alterar o modo de sua transmissão a qualquer momento, bastando para isso enviar um pacote do tipo DCCP-MSYNC, via *multicast*, para o canal de controle do MU-DCCP, contendo a modificação desejada. Ao receber um DCCP-MSYNC, os nós envolvidos na recepção dos dados devem realizar as devidas modificações em seu modo de recepção. O formato do DCCP-MSYNC é igual ao formato do pacote DCCP-MRESPONSE.

4.3 Controle de congestionamento com DCCP em modo multicast

O MU-DCCP suporta transmissão de dados tanto *unicast* quanto em modo *multicast*. Algoritmos de controle de congestionamento em modo *multicast* são mais complexos porque geralmente estes utilizam o *feedback* dos receptores para tomar decisões na definição da sua taxa de transmissão. Neste caso, nenhum CCID existente para DCCP foi projetado para funcionar neste modo, sendo necessário trabalhar em um novo CCID capaz de tratar todas as

características de transmissão de dados de fluxo não confiável em modo *multicast*. No contexto deste trabalho, definiu-se o CCID-5, um algoritmo para controle de congestionamento de fluxos de dados não confiável em transmissões *multicast* utilizando o protocolo DCCP.

O foco principalmente de funcionamento do CCID-5 é fazê-lo capaz de ser executado em um DCCP *Relay* que esteja transmitindo em modo *multicast*. O CCID-5 determina uma nova taxa de transmissão baseado em relatórios enviados pelos nós receptores da sua rede local. Note que, como estão sendo tratados fluxos de dados não confiáveis, a solução para o CCID-5 pode fazer uso deste tipo de transmissão para reduzir a necessidade de confirmação de todos os pacotes recebidos pelos nós receptores. Como estão sendo considerados cenários onde existem apenas um nó transmissor dos dados e diversos nós receptores, receber confirmação de recepção de todos os nós receptores poderia inundar o nó transmissor de pacotes de controle, com confirmação de recepção de pacotes de dados por parte dos receptores.

Assim, foi possível desenvolver o CCID-5 como um algoritmo de controle de congestionamento que não precisa necessariamente receber um relatório de recepção de todos os nós receptores, mas sim de um sub-grupo de nós receptores, reduzindo de tal forma a quantidade de dados de controle transmitidos na rede em direção ao nó transmissor dos dados. É importante salientar que nem sempre o nó transmissor dos dados será aquele que gera o conteúdo, mas sim um nó transmissor que poderá executar o CCID-5 pode ser um nó do tipo DCCP *Relay*. Dado isto, desenvolveu-se um algoritmo para permitir a eleição de nós DCCP *Relays* e nós DCCP *Reporters*, responsáveis por relatar a um nó transmissor qualquer sua taxa de recepção de dados via *multicast*. Ao receber relatórios enviados pelos DCCP *Reporters*, os nós DCCP *Relays* ajustarão sua taxa de transmissão de acordo com estes relatórios, o que pode ser feito utilizando uma equação de cálculo de taxa de transmissão, como por exemplo a própria Equação 1.1.

4.4 Eleição de Nós DCCP Relays e de DCCP Reporters

Os nós DCCP *Relays* são selecionados de duas formas: (i) serão DCCP *Relays* aqueles que iniciarem a primeira conexão *unicast* com um nó DCCP gerador dos dados, ou seja, o nó transmissor original; (ii) serão DCCP *Relays* aqueles que negociarem com algum outro nó DCCP *Relay* sua promoção para nó DCCP *Relay*. Note-se que este caso o nó que conceder

a promoção de um nó DCCP *Relay* para outro deverá se rebaixar para um nó MU-DCCP normal. Além disso, quando um DCCP *Relay* conceder este *status*, o mesmo poderá se desconectar do nó DCCP gerador dos dados enviando um DCCP-MRESET, que conterá o endereço do novo nó DCCP *Relay*. É possível também que um nó DCCP *Relay* eleja outros nós DCCP *Relays* secundários, localizados na sua própria rede local. Esta funcionalidade é importante porque caso o atual DCCP *Relay* perca sua conexão ou desconecte do nó transmissor gerador dos dados, qualquer DCCP *Relay* secundário poderá assumir o papel de DCCP *Relay* primário. Neste caso, o nó que passar a assumir este papel deverá enviar um pacote do tipo DCCP-MELECT informando que assumirá a transmissão de dados outrora provida pelo nó DCCP *Relay* antigo.

Com relação aos nós DCCP *Reporters*, o processo de eleição funciona de forma similar, porém qualquer nó pode se tornar um DCCP *Reporter*. O processo funciona da seguinte forma: à medida que um DCCP *Relay* receber pacotes do tipo DCCP-MREQUEST, no pacote DCCP-MRESPONSE o nó DCCP *Relay* pode ativar a *flag* desse pacote informando que aquele nó deverá enviar relatórios de confirmação de pacotes. Desta forma, o DCCP *Relay* poderá limitar a quantidade de DCCP *Reporters* e assim receber relatórios apenas de um subconjunto de nós da rede. Como a transmissão é *multicast* e local, é possível ter poucos nós DCCP *Reporters* e continuar sendo efetivo no ponto de vista de determinar a melhor taxa de transmissão. No contexto deste trabalho, não realizou-se muitos experimentos na tentativa de se é possível determinar qual é a quantidade necessária de DCCP *Reporters* para se obter taxas de transmissões justas.

4.5 Adaptação de Fluxo Multimídia

Uma funcionalidade peculiar do MU-DCCP é a capacidade que ele tem de fazer adaptação de fluxos multimídia de forma distribuída. A maioria das soluções para transmissão de dados multimídia, além de realizarem controle de congestionamento no nível de aplicação, realizam adaptação de fluxo multimídia na fonte geradora dos dados. Uma solução para adaptação de fluxo multimídia é transmitir os dados em diferentes canais, sendo que em cada canal transmite-se fluxos multimídia com uma determinada qualidade. Dependendo da qualidade desejada pelo nó receptor, ele solicita a transmissão em um determinado canal. O

problema dessa abordagem é que o nó transmissor, necessariamente deve transmitir os dados em múltiplos canais, o que aumenta a complexidade da aplicação e a quantidade de fluxos de dados sendo transmitidos a partir do servidor. No MU-DCCP, é possível realizar a adaptação de fluxo de dados de forma distribuída, na prática, em cada DCCP *Relay*. Suponhe-se que existem duas redes adjacentes, rede 1 e rede 2. Considere que existe um nó DCCP *Relay* na rede 1 e entre esta e o nó transmissor a largura de banda disponível é de 100 *Mbps*. Caso a largura de banda disponível na rede 2 seja de no máximo 10 *Mbps*, um nó receptor na rede 2 teria que solicitar um fluxo multimídia em um canal diferente, considerando a solução supracitada. No caso do MU-DCCP é possível que um nó na rede 2 obtenha o fluxo multimídia através do DCCP *Relay* presente na rede 1, bastante, neste caso, que o DCCP *Relay* presente na rede 1 adapte o fluxo para o nó que esteja na rede 2. Desta forma, pode-se diminuir o tráfego na rede do nó transmissor e ainda sim permitir que nós em redes com baixa largura de banda consigam obter o fluxo multimídia adaptado.

4.6 Compatibilidade com outros protocolos e as recomendação da IETF

4.7 Considerações sobre redes de distribuição de conteúdo

4.8 Considerações sobre a escolha do DCCP como base para o MU-DCCP

4.9 Considerações sobre segurança

Capítulo 5

Implementação do MU-DCCP

5.1

Capítulo 6

Métodos, Simulações 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 [?], 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 ??.

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.).

6.1 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 \times UDP, TCP \times DCCP e UDP \times DCCP. A quantidade dos fluxos transmitidos durante os experimentos são apresentados na Tabela 6.1.

#	Confrontos	Fluxo TCP	Fluxos UDP	Fluxos DCCP
1	TCP \times UDP	1	2	0
2	TCP \times DCCP	1	0	2
3	UDP \times DCCP	0	2	1

Tabela 6.1: Quantidade de fluxos utilizados por confronto de protocolos.

O objetivo na definição destes confrontos foi analisar a equidade 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 diferentes 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 2.6.3 e ??, 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 2.6.3) 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 6.2, 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 6.2: 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 equidade 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 6.3. É sabido que durante a execução de um *hand-off* ocorrem perdas de pacotes (Seção ??) 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 2.5, 2.6.3, 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*.

6.2 Experimentos

Para todos os cenários de experimentos, primeiramente foi transmitido apenas um fluxo do protocolo TCP, ao passo que sempre¹ 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.

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.

6.3 Métricas Seleccionadas 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

¹Exceto para os confrontos DCCP × UDP, onde foi transmitido apenas 1 fluxo DCCP e 2 fluxos UDP.

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 através 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 6.2. Contudo, para cada métrica definida, é necessário fazer algumas considerações, as quais são feitas a seguir.

6.3.1 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 6.1 e 6.2, onde n é o total de repetições. Assim, temos:

$$\mu_{vazao_{tcp}} = \frac{\sum_{r=1}^n vazao_media_r}{n} \quad (6.1)$$

$$\mu_{carga_{tcp}} = \frac{\sum_{r=1}^n carga_media_r}{n} \quad (6.2)$$

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:

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

sendo que,

$$vazão_media_r = \frac{\sum_{k=1}^F vazão_média_fluxo_k}{F}$$

e a $vazão_media_fluxo_k$ é obtida através da média aritmética das vazões em cada segundo do experimento. Portanto, o valor final para a vazão dos protocolos UDP e DCCP é obtida através da Equação 6.3.

$$\mu_{vazao-final(udp/dccp)} = \mu_{vazao-parcial(udp/dccp)} + S \times \left(\frac{\mu_{vazao-parcial(udp/dccp)}}{T} \right) \quad (6.3)$$

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 ($S = 20\text{ s}$); e
- T , o tempo total do experimento ($T = 100\text{ s}$ ou $T = 300\text{ s}$).

Assim, através da Equação 6.3 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 6.1 e 6.2, 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 6.5. Esta equação foi obtida da seguinte forma:

$$\mu_{jitter-parcial(udp/dccp)} = \frac{\sum_{r=1}^n jitter_medio_r}{n} \quad (6.4)$$

onde,

$$jitter_medio_r = \frac{\sum_{k=1}^F \left(\frac{\sum_{k=1}^{QI} VA_k}{QI} \right)}{F}$$

Logo,

$$\mu_{jitter-final(udp/dccp)} = \mu_{jitter-parcial(udp/dccp)} + S \times \left(\frac{\mu_{jitter-parcial(udp/dccp)}}{T} \right) \quad (6.5)$$

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 $tempo_1 = 10\text{ ms}$ e $tempo_2 = 11\text{ ms}$, $VA = 1\text{ ms}$;
- T , o tempo total do experimento ($T = 100\text{ s}$ ou $T = 300\text{ s}$).

6.4 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 [?], 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 $\{x_1, \dots, x_n\}$. 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:

$$\text{Probabilidade}\{c_1 \leq \mu \leq c_2\} = 1 - \alpha \quad (6.6)$$

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
- $\rho = 100 * (1 - \alpha)$, 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 [?], se um conjunto de amostras $\{x_1, \dots, x_n\}$ 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 σ/\sqrt{n} :

$$\bar{x} \simeq N\left(\mu, \frac{\sigma}{\sqrt{n}}\right) \quad (6.7)$$

Então, tendo como base a relação 6.6 e o *Teorema do Limite Central* (6.7), obtém-se o intervalo de confiança (c_1, c_2) para $\rho = 95\%$ e $\alpha = 0.05$ da seguinte forma:

$$\left(\mu - z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}, \mu + z_{1-\alpha/2} \times \frac{s}{\sqrt{n}}\right) \quad (6.8)$$

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 [?];

²Teorema 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.

- 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 frequentemente 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 [?]. Para determinar este valor, temos:

$$z_{1-\alpha/2} = (1 - 0.05)/2 = 0.975 \quad (6.9)$$

O valor correspondente ao resultado da Equação 6.9, 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 6.3.1, 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 6.8, temos que o intervalo de confiança para uma quantidade n de amostras é definido da seguinte forma:

$$\mu \pm z \times \frac{s}{\sqrt{n}} \quad (6.10)$$

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

$$(\mu(1 - 0.05), \mu(1 + 0.05)) \quad (6.11)$$

Então, igualando os intervalos de confiança 6.11 ao intervalo de confiança 6.10 (geral), obtemos a Equação 6.12.

$$\mu \pm z \times \frac{s}{\sqrt{n}} = \mu(1 \pm 0.05) \quad (6.12)$$

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 6.13, para um nível de confiança $\rho = 95 \%$, o que implica em $z = 1.96$ (a partir da Equação 6.9).

$$n = \left(\frac{1.96 \times s}{0.05 \times \mu} \right)^2 \quad (6.13)$$

6.5 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*) [?] 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 ??.

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:

$$RMS = \sqrt{\frac{(x_1)^2 + (x_2)^2 + (x_3)^2 + \dots + (X_n)^2}{n}} \quad (6.14)$$

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 7

Análise de Desempenho do DCCP

7.1

Capítulo 8

Análise de Desempenho do MU-DCCP

8.1

Capítulo 9

Conclusão e Planejamento

9.1 Contribuições

9.2 Limitações do Trabalho

9.3 Perspectivas

Bibliografia

- [1] Amazon. Introducing Amazon Cloud Player for Web & Android. Online publication in Amazon.com, 4 2011. <http://www.amazon.com/b?ie=UTF8&node=2658409011>.
- [2] Leandro Melo de Sales, Hyggo O. Almeida, Angelo Perkusich, and Marcello Sales Jr. An Experimental Evaluation of DCCP Transport Protocol: A Focus on the Fairness and Hand-off over 802.11g Networks. In *In Proceedings of the 5th IEEE Consumer Communications and Networking Conference*, pages 1149–1153, 1 2008.
- [3] Leandro Melo de 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.
- [4] Sally Floyd, Mark Handley, and Eddie Kohler. Problem Statement for the datagram congestion control protocol (DCCP), 2006. <http://www.ietf.org/rfc/rfc4336.txt>. Ultimo acesso: 12/04/2011.
- [5] Torrent Freak. Paramount Pictures to Release Film on Bittorrent. Online publication in the Torrent Freak Website, 3 2011. <http://torrentfreak.com/paramount-pictures-partner-with-bittorrent-release-movie-110317/>.
- [6] Garrett Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243 – 1248, 3 1968.
- [7] Eddie Kohler and Mark Handley. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), 3 2006. <http://www.ietf.org/rfc/rfc4342.txt>. Ultimo acesso: 12/04/2011.

-
- [8] Eddie Kohler, Mark Handley, and Floyd. Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control. In *IETF Online RFC*, 3 2006. <http://www.ietf.org/rfc/rfc4341.txt>. Ultimo acesso: 12/04/2011.
 - [9] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP), 3 2006. <http://www.ietf.org/rfc/rfc4340.txt>. Ultimo acesso: 12/04/2011.
 - [10] 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.
 - [11] N. Leavitt. Network-Usage Changes Push Internet Traffic to the Edge. *Computer*, 43(10):13–15, 2010.

Apêndice A

A Ferramenta GST-DCCP

Apêndice B

DCCP no NS-3