PAULO ROBERTO DE ALMEIDA

VALIDAÇÃO DE MODELO MATEMÁTICO PARA VERIFICAÇÃO DO COMPORTAMENTO DO PROTOCOLO TCP EM REDES ASSIMÉTRICAS

LAVRAS – MG

2013

PAULO ROBERTO DE ALMEIDA

VALIDAÇÃO DE MODELO MATEMÁTICO PARA VERIFICAÇÃO DO COMPORTAMENTO DO PROTOCOLO TCP EM REDES ASSIMÉTRICAS

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

Orientador
Prof. Ms. Eric Fernandes de Mello Araújo

LAVRAS – MG
2013

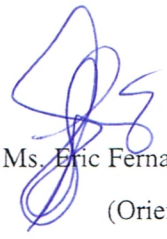
PAULO ROBERTO DE ALMEIDA

VALIDAÇÃO DE MODELO MATEMÁTICO PARA VERIFICAÇÃO DO COMPORTAMENTO DO PROTOCOLO TCP EM REDES ASSIMÉTRICAS

Monografia apresentada ao Colegiado do Curso de Ciência da Computação, para a obtenção do título de Bacharel em Ciência da Computação.

APROVADA em 15 de Abril de 2013.

Prof. Dr. Luiz Henrique Andrade Correia	UFLA
Prof. Dr. Tales Heimfarth	UFLA



Prof. Ms. Eric Fernandes de Mello Araújo
(Orientador)

LAVRAS – MG

2013

Dedico essa monografia a minha avó Maria.

AGRADECIMENTOS

À Deus, meu guia, conforto e proteção durante toda minha vida.

A toda minha família, por ter acreditado em mim e sempre me incentivado.

A Caroline, pelo apoio e compreensão nas horas difíceis durante o desenvolvimento deste projeto.

Ao meu orientador Prof. Eric Fernandes de Mello Araújo pelos conselhos, correções e ajuda imprescindível neste trabalho.

Enfim, a todos que contribuíram para esse importante passo e para a realização desse sonho tão esperado.

RESUMO

Este trabalho apresenta um estudo a respeito do efeito do desempenho do protocolo TCP nas redes assimétricas, em específico sobre as redes ADSL, que estão entre as redes mais utilizadas para acesso à internet devido a seu baixo custo de implantação. O protocolo TCP é um protocolo da camada de transporte mais utilizado na internet. O TCP possui características como confiabilidade de entrega dos dados, controle de congestionamento e controle de fluxo. O trabalho realiza a verificação do modelo matemático proposto por Araújo (2009). Foram realizadas simulações através do simulador NS2 e os resultados obtidos mostraram que quanto maior o grau de assimetria entre os canais maior será o desempenho da rede.

Palavras-Chave: Protocolo TCP; Redes Assimétricas; ADSL.

ABSTRACT

This work presents a study about the effect of the protocol performance TCP on asymmetric networks, in particular on ADSL networks, which are among more networks used for internet access due to its low cost of implantation. The TCP is a transport layer protocol more used in internet. The TCP has characteristics such as reliability of data delivery, congestion control and flow control. The work performs verification the mathematical model proposed by Araújo (2009). Were performed Simulations through the NS2 simulator and the results showed that as higher the degree of asymmetry between channels higher is network performance.

Keywords: Protocol TCP; Asymmetric Networks; ADSL.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A internet conquistou o mundo e passou a ser um dos meios mais acessados e usados na sociedade para troca de informações. Segundo KUROSE (2010) a Internet teve seu início nos Estados Unidos na década de 60, recebendo o nome de ARPANET. Posteriormente, na década de 80, teve um grande crescimento no seu uso através da integração das redes de diversas universidades americanas. Na década de 90 com o fim da ARPAnet e com o surgimento da *World Wide Web* (WWW) a internet pôde ser acessada em todos os lares e empresas mundialmente, permitindo grandes trocas de informações entre computadores em pontos diferentes do globo, aumentando assim o poder de comunicação.

Entre as razões que levaram a internet a ter um crescimento nessas proporções estão o aumento contínuo da velocidade em que os dados são enviados, sua facilidade de uso, as aplicações criadas para lazer, comunicação, trabalho e outras ferramentas oferecidas aos usuários.

Novas tecnologias e ferramentas foram e têm sido criadas com o objetivo de apoiar e permitir o crescimento da rede mundial de computadores. Entre essas tecnologias, destacam-se as que envolvem a estrutura física de transmissão dos dados. As primeiras redes que surgiram utilizavam a rede telefônica para transmitir os dados, e devido ao baixo custo de implantação dessas redes, foi permitido um crescimento maciço da internet e uma adesão por milhões de usuários.

As redes domésticas foram criadas refletindo as aplicações da época, bem como o comportamento de seus usuários, que eram mais passivos do que ativos, ou seja, recebiam mais informações da rede do que enviavam. Aplicações para navegação em *sites* e leitura de *e-mails* eram predominantes nesse momento. Para atender à essa demanda, os sistemas para as chamadas redes domésticas tinha uma

capacidade de recebimento de dados maior do que a capacidade de envio destes por parte do usuário da rede, configurando uma rede assimétrica.

Com a evolução da internet e o surgimento de novas aplicações, como por exemplo a *Web* 2.0 e as redes *peer-to-peer* (P2P), observou-se uma mudança de comportamento por parte do usuário, sendo que este agora demanda uma capacidade de envio de dados igual ou superior à capacidade de recebimento.

Em outros contextos observamos assimetrias entre as bandas de *download* e *upload*, podendo essa ser na capacidade de transmissão ou até mesmo no atraso para envio dos dados. Alguns exemplos de redes assimétricas são as redes de Satélites e as redes *Assymmetric Digital Subscriber Line* (ADSL) (BALAKRISHNAN; PADMANABHAN; KATZ, 1999).

1.1 Contextualização e Motivação

Atualmente, no Brasil, as redes ADSL estão entre as redes mais utilizadas para acesso à internet. Não ocupa a linha telefônica, permite a transferência digital de dados em alta velocidade, possui baixo custo de implantação, o que tem feito com que as empresas provedoras de internet forneçam, em sua grande maioria, canais de transmissão assimétricos, ainda que o usuário venha demandando uma taxa de transmissão equivalente para envio e recebimento de dados.

O *Transmission Control Protocol* (TCP) é o protocolo mais utilizado atualmente na internet, e tem como principais características a garantia de entrega dos dados, controle de fluxo e controle de congestionamento.

Em uma rede assimétrica temos problemas para o protocolo TCP, pois este necessita de reconhecimentos para aumentar sua taxa de transmissão ou estimar seu *Round-trip time* (RTT). Nas próximas seções estudaremos o problema quando

há congestionamento entre dados e ACKs no canal reverso (*upload*) provocando perdas de desempenho no outro canal (*download*).

1.2 Objetivos Gerais

A finalidade deste trabalho é realizar a verificação e estudo do modelo matemático proposto por Araújo (2009).

1.3 Objetivos Específicos

Os objetivos específicos são:

- Avaliar o protocolo TCP utilizando o simulador NS2.
- Relacionar o comportamento do protocolo TCP com o modelo matemático proposto pelo autor Araújo (2009).
- Constatar a proximidade da simulação com o modelo matemático proposto por Araújo (2009).

1.4 Organização do Trabalho

Este trabalho está organizado da seguinte forma: O capítulo 2 apresenta o Referencial Teórico, tendo um estudo a respeito dos principais conceitos que foram utilizados neste trabalho e sobre os trabalhos relacionados já desenvolvidos. Em seguida, o capítulo 3 descreve a metodologia que foi usada para o desenvolvimento dessa pesquisa. O capítulo 4 traz os resultados que foram obtidos com este trabalho. As discussões e conclusões são apresentadas no capítulo 5.

2 REFERENCIAL TEÓRICO

Para compreensão e entendimento dos resultados desse trabalho, é importante conhecer o protocolo TCP, bem como suas características. Também é relevante estudar a escolha do simulador a ser utilizado, bem como estudos mais aprofundados na modelagem matemática de Araújo (2009). Dessa forma, iremos tratar aqui desses tópicos, de forma a contextualizar o leitor em relação ao trabalho produzido.

2.1 Protocolo TCP

O protocolo TCP da arquitetura TCP/IP é o objeto de estudo deste trabalho, sendo que suas principais características são a garantia de entrega dos dados, controle de fluxo e controle de congestionamento. Este protocolo é muito utilizado na internet.

Para que se tenha uma garantia de entrega em ordem, é alocado para cada segmento TCP um número de sequência e um contador que representa o número de *bytes* de dados que foram enviados. Com estas características tem-se a garantia que os dados chegaram em ordem ao receptor, e pelo número de sequência pode ser verificado se os dados foram recebidos com sucesso (KUROSE, 2010).

KUROSE (2010) cita as demais características do protocolo TCP:

- *Unicast*: suportando assim a comunicação apenas entre dois hospedeiros.
- *Full duplex*: ambas as partes envolvidas na comunicação podem enviar dados. Se tivermos dois processos A e B, os dados poderão fluir tanto do processo A quanto do processo B.
- Mantém um estado de conexão que representa o estabelecimento da conexão entre as partes envolvidas: Este estabelecimento é chamado de *handshake* entre cliente e o servidor.

- Confiabilidade no transporte de dados: garantindo que os dados serão enviados em ordem do processo transmissor para o receptor. Quando ocorrem perdas de alguns destes dados na rede, haverá retransmissão para que assim se possa ter uma coerência de ordem dos dados que estão sendo armazenados no receptor. O transmissor espera a confirmação de recebimento dos dados por parte do servidor, após o recebimento desta, novos dados são transferidos entre os processos em questão.

- Protocolo *Streaming*: não há uma regra determinada para a emissão e recepção dos segmentos. Se tivermos um remetente enviando três pacotes de dados consecutivos, estes poderão ser recebidos (verificados) pelo receptor em uma única operação de leitura.

- Possui taxa de adaptação: o TCP tem sua taxa adaptada a capacidade de transmissão que a rede suporta ou em relação à capacidade de processamento do receptor. Sendo que assim, não há uma taxa de transmissão de dados pré definida. Se houver disponibilidade de capacidade o transmissor enviará uma quantidade de dados maior. Ao perceber um congestionamento o transmissor diminui sua taxa de transmissão a fim de fazer com que haja uma recuperação por parte da rede.

O TCP possui também dois mecanismos importantes: o primeiro é o chamado *Maximum Segment Size* (MSS), que corresponde ao tamanho do maior segmento que o transmissor pode enviar ao receptor. O MSS poderá depender do tamanho do *buffer* disponível do receptor. Pois, se o transmissor enviar um fluxo de dados maior que a capacidade de *buffer* do receptor, haverá problemas relacionados ao controle de fluxo, o qual será explicado na seção 2.1.2. O segundo mecanismo é *Maximum Transmission Unit* (MTU) que representa a quantidade máxima de transmissão que pode ser enviada na rede (KUROSE, 2010).

A janela de transmissão é outra característica importante para se ter um melhor aproveitamento da largura de banda disponível na rede. Ela nos mostra o tempo esperado por um transmissor para que este receba um reconhecimento referente aos dados entregues com sucesso. Um exemplo deste tempo durante a transmissão é o *Round-trip time* (RTT) que corresponde ao tempo de ida e volta da comunicação. Este tempo é calculado em relação ao tempo que um segmento gasta para chegar a seu destino e ter seu reconhecimento, ACK recebido.

O protocolo TCP possui uma Janela Deslizante. Seu principal objetivo é dar apoio à transferência de uma determinada quantidade de dados. Seu uso poderá ser verificado na seção 2.1.2, a qual trata do controle de fluxo durante a transmissão. Durante a transmissão dos dados, quando se tem um reconhecimento de um respectivo dado enviado, a janela deslizante desliza para que novos segmentos possam ser enviados à rede, respeitado a capacidade de *buffer* do receptor (TANENBAUM, 2003).

O TCP conta com um mecanismo chamado *Piggybacking*, usado em transmissões bidirecionais. Pacotes de dados e de reconhecimentos (ACKs) são enviados nos dois sentidos: transmissor-receptor e receptor-transmissor. Tem-se o uso de pacotes de dados para confirmar a chegada dos pacotes recebidos da transmissão no outro sentido.

O protocolo TCP possui algumas fragilidades. Por exemplo, quando um pacote é perdido tem-se a necessidade de reenviar este dado, consumindo assim parte da banda que poderia ser usada para envio de outros dados. Assim, outros dados devem ficar na fila do roteador para serem enviados a seu destino (TANENBAUM, 2003).

2.1.1 Estrutura do cabeçalho TCP

Segundo KUROSE (2010), as funcionalidades dos campos do cabeçalho TCP são de fundamental importância para seu desempenho, contendo informações importantes relacionadas a controle de congestionamento e janela deslizante. A figura 2.1 mostra a estrutura do respectivo cabeçalho:

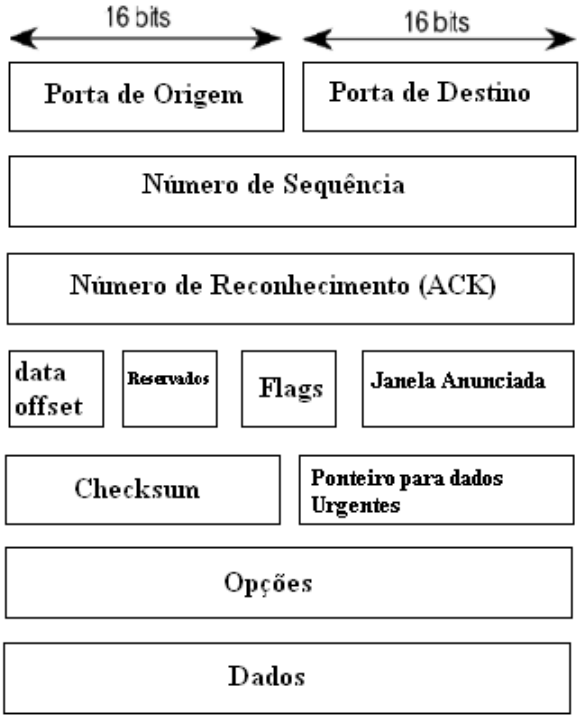


Figura 2.1: Estrutura do Cabeçalho TCP (KUROSE, 2010).

O cabeçalho TCP contém informações referentes às portas de destino e origem. Quando se tem inicialização da conexão TCP, há alocação de portas para os hospedeiros que irão transmitir e receber os dados.

O campo número de sequência informa qual a posição ocupada do segmento na sequência de dados que estão sendo enviados.

O campo número de reconhecimento (ACK) informa o último segmento recebido com sucesso pelo receptor. O receptor atribui um número de confirmação que corresponde ao número de sequência do próximo *byte* que ele espera receber do transmissor.

O campo Janela Anunciada refere-se à janela de recepção do protocolo TCP. Sendo informada a disponibilidade de espaço na janela de transmissão que será usada pelo controle de fluxo.

O campo *flags* é composto por *bits* que serão usados para representar determinadas condições. O campo URG é usado juntamente com o ponteiro para dados urgentes, sua função é indicar que um determinado dado tem certa urgência para ser entregue a seu destinatário. O campo ACK é utilizado para informações de recebimentos dos dados pelo segmento. O campo PSH é utilizado para indicar que os dados que foram enviados sejam entregue o mais rápido possível. O campo RST é usado para redefinir a conexão. O campo SYN é utilizado na etapa de inicialização da conexão. O campo FIN é usado na etapa de encerramento da conexão.

O campo ponteiro para dados urgentes, é usado como forma de indicação da localização do último *byte* dos dados que foram marcados como urgentes.

O campo *checksum*, tem como funcionalidade garantia da integridade dos pacotes de dados, ou seja, nenhum dos *bits* do pacote foi entregues corrompidos.

2.1.2 Controle de Fluxo

Nesta seção, iremos tratar de como é feito o controle de fluxo pelo protocolo TCP, bem como quais problemas ocorrem quando se tem uma saturação do *buffer* entre as partes envolvidas na comunicação.

No processo de transmissão de dados, o protocolo TCP faz uso da técnica de controle de fluxo para buscar o ponto de equilíbrio nesta transmissão. A importân-

cia de buscar este ponto de equilíbrio pode ser verificada em dois casos. O primeiro deles é se o protocolo TCP enviar uma alta quantidade de dados na rede, pois isto poderá ocasionar um congestionamento, e posteriormente perda de dados. O outro caso refere-se ao TCP enviar os dados com uma baixa taxa de transmissão, isto poderá levar ao canal ficar ocioso. Para que se tenha este controle da quantidade de dados que são enviados entre os hospedeiros, tem-se a necessidade de verificar a capacidade do transmissor, receptor e da rede (KUROSE, 2010).

Segundo TANENBAUM (2003), para que se tenha êxito no controle de fluxo é necessário usar uma janela deslizante em cada conexão. A função desta é impedir que um determinado transmissor rápido sobrecarregue um receptor lento.

O uso do protocolo de janela deslizante é um fator que auxilia para que se tenha informação do ponto de equilíbrio na transferência de dados. A cada reconhecimento de um dado, o receptor informa ao transmissor o espaço em *buffer* disponível. O transmissor verifica a capacidade da janela anunciada pelo receptor para que se evitem problemas durante o processo de transmissão. A rede possui uma janela de congestionamento, que refere-se a quantidade de dados que podem ser enviados na rede sem que haja congestionamento. O transmissor irá fazer uma adaptação de sua taxa de transmissão respeitando os limites que foram estabelecidos pelo receptor e pela rede (TANENBAUM, 2003).

Segundo KUROSE (2010), a função do controle de fluxo é fazer com que o transmissor não esgote a capacidade de *buffer* do receptor envolvido na comunicação. Os hospedeiros de cada lado mantêm um respectivo *buffer* e tem-se uma janela de recepção no lado transmissor, esta se referindo à capacidade de *buffer* disponível no lado receptor.

Segundo KUROSE (2010), o tamanho do *buffer* de recepção disponível para a comunicação pode ser calculado pela diferença entre o número de sequência

do maior segmento que pode ser recebido pela rede e em relação ao número de sequência do último segmento que chegou à rede. Conforme equação 2.1:

$$MaximoSeg - NumeroSeqUltimo \leqslant buferReceptor \tag{2.1}$$

A verificação pelo transmissor da janela de envio pode ser obtida pela diferença entre o último segmento enviado e a última confirmação recebida. Se esta diferença for menor que a janela de envio, o transmissor terá certeza que não está saturando o *buffer* do receptor. Conforme equação 2.2:

$$SegEnviado - UltimoAckRec \leqslant TamJanelaEnvio \tag{2.2}$$

Com o uso da janela deslizante poderá ter um maior controle de fluxo entre as partes envolvidas na transmissão dos dados. Esta janela irá informar a quantidade de dados que poderão ser transmitidos, sendo que ela irá aumentar à medida que novos reconhecimentos (ACK) atualizem a capacidade de *buffer* no receptor.

2.1.3 Princípios do Controle de Congestionamento

Nesta seção, iremos tratar de como é feito o controle de congestionamento pelo protocolo TCP, e quais são os problemas que ocorrem quando se detecta um congestionamento na rede.

Segundo TANENBAUM (2003), ocorre um congestionamento na rede quando a carga de dados que está sendo transportada é maior que sua capacidade. Ou seja, este congestionamento é gerado por transmissores enviando dados com uma taxa maior do que a suportada pela rede. O controle de congestionamento é de fundamental importância para que se tenha uma estabilidade da internet e refere-se à capacidade da rede.

A busca por um ponto de equilíbrio entre a taxa de envio de dados e capacidade da rede é muito importante para que não ocorra perda de pacotes ou atrasos na rede. Se esta busca não for bem definida, pode ocasionar um mal uso da capacidade da rede, sendo que esta ficará ociosa durante certo período de tempo, provocando problemas na transmissão dos dados.

Assim, o uso deste mecanismo pelo protocolo TCP é de fundamental importância, pois ele irá controlar as ações do transmissor e receptor e toda a capacidade da rede para que se possa ter uma transferência confiável dos dados.

Segundo KUROSE (2010), os mecanismos para controle de congestionamento são assistidos por rede ou fim a fim. O controle assistido por rede tem como característica que os componentes da camada de rede (exemplo roteadores) forneceram informações de possíveis congestionamentos aos hospedeiros. O controle de congestionamento assistido por rede não é a melhor forma para detecção do congestionamento, pois a camada de rede IP não fornece um tipo de serviço confiável para detecção deste.

No controle fim a fim, a camada de rede não irá fornecer nenhum suporte sobre os possíveis congestionamentos que esta ocorrendo na rede. Os sistemas finais que irão verificar que está havendo congestionamento na rede.

Uma das causas do congestionamento é a formação de filas entre os pacotes de dados e reconhecimentos (ACKs). Quando isso ocorre, tem-se perdas de desempenho no canal onde estão trafegando os dados ou até mesmo perdas de informações.

Para um melhor controle de congestionamento, têm-se algoritmos que tentam fazer com que não haja ociosidade na rede ou problemas relacionados à perda de informações.

2.1.4 Algoritmo Partida Lenta

Este algoritmo foi desenvolvido por Jacobson (1988), com a finalidade de que se tenha um aumento da taxa de transmissão, sem que ocorra uma sobrecarga na capacidade da rede. Tem-se o uso de uma variável chamada janela de congestionamento (cwnd), sendo que esta é o tamanho da janela deslizante usada pelo transmissor e menor que a capacidade da janela do receptor (definida no momento de estabelecimento da conexão).

KUROSE (2010) afirma que inicialmente o valor da janela de congestionamento (cwnd) é 1MSS, sendo que o objetivo dessa cwnd é começar a transmissão com uma taxa que tem baixa probabilidade de gerar perda de dados na rede. A cada reconhecimento de um segmento que foi enviado, o valor de cwnd é aumentado em 1 MSS. Ou seja, a taxa e envio tem o início lento, mas cresce exponencialmente durante o processo o qual chamamos de partida lenta.

Por exemplo, considerando que inicialmente tem-se um envio de 1 segmento. Após o recebimento do reconhecimento de entrega deste segmento, a cwnd é aumentada para dois segmentos, podendo assim ser enviado mais 2 segmentos para a rede. Quando esses segmentos são reconhecidos, o remetente aumenta a janela de congestionamento para 1 MSS para cada segmento reconhecido, fornecendo assim uma janela de congestionamento de 4 MSS. Este processo resulta na multiplicação da taxa de envio a cada RTT.

Este crescimento exponencial termina em duas situações. A primeira é quando ocorre um evento de perda, sendo que o remetente TCP irá restabelecer o valor de cwnd para 1 e fazer a inicialização do processo de partida lenta novamente. Na segunda situação, é definida uma região chamada de *threshold*. Inicialmente essa região de *threshod* é igual ao tamanho máximo da janela do receptor. Sendo que a partida lenta continua até que o valor da cwnd não supere esta região. Quando o

congestionamento for detectado, a região de *threshold* será igual à metade do valor quando houve perda dos dados.

2.1.5 Perda de pacotes no TCP

Um dos motivos das perdas de pacotes é o congestionamento no canal de transmissão. Os pacotes de dados ficarão enfileirados no roteador, sendo que este poderá descartar os pacotes por falta de espaço no *buffer*.

Segundo Huston (2000), o transmissor poderá detectar a perda de pacotes de duas maneiras: através de ACK duplicados, ou quando há uma expiração do RTT da rede.

Quando se tem uma transmissão de vários pacotes na rede, se ocorrer uma perda de um algum destes pacotes, esse evento de perda irá fazer com que o receptor enviem ACKs duplicados para cada pacote com número de sequência maior que o pacote perdido. O fato do recebimento de ACKs duplicados é entendido pelo transmissor ocorreu perda durante a transmissão.

O outro evento de perda é detectado quando se tem uma expiração do RTT. Como vimos nas seções anteriores, isso ocorre pois o transmissor não recebe o reconhecimento dos seus dados enviados em um certo período de tempo.

Quando for detectado o evento de perda de pacotes, os dados devem ser retransmitidos, respeitando a taxa de envio da rede. A retransmissão dos dados é usada para que se tenha uma garantia de entrega e em ordem.

Deve-se buscar o ponto de equilíbrio da taxa de envio dos dados, para que esta não sature a capacidade da rede.

2.2 Redes Assimétricas

Segundo Toledo (2001) as redes assimétricas possuem diferentes velocidades nos sentidos da conexão, sendo que é reservado uma maior largura de banda para o canal de *download* do que para o *upload*.

As redes assimétricas são redes na qual temos diferença entre os canais de recepção e transmissão de dados. Um exemplo deste tipo de redes são as redes ADSL e as redes via satélite (BALAKRISHNAN; PADMANABHAN; KATZ, 1999).

Nas redes via satélite, apenas o canal de *download* era utilizado, sendo que os pacotes de confirmação eram retornados por uma rede discada, gerando assim uma grande assimetria entre as bandas dos canais que são utilizados para enviar os dados e também havia grande diferença nos tempos de transmissão. Atualmente, nas redes via satélite *download* e *upload* são feitos utilizando o mesmo canal, não havendo assim problemas relacionados à assimetria.

As redes ADSL que são usadas atualmente possuem grande diferença entre a capacidade de *download* (receber dados da rede) e capacidade de *upload* (enviar dados à rede). As empresas oferecem ao cliente uma banda maior para receber dados do que para enviar dados à rede.

Segundo Balakrishnan, Padmanabhan e Katz (1999), quando temos uma transferência bidirecional, dados e ACKs poderão ser enviados nos dois sentidos. Este tipo de transferência poderá gerar problemas, como por exemplo, conflito entre pacotes de dados e reconhecimentos (ACKs). Ocasionando assim um enfileiramento de ACKs e dados no canal reverso. Como vimos nas seções anteriores, ocorrerá perda de desempenho no canal de *download*, pois este depende do recebimento de reconhecimentos (ACKs) para aumentar sua taxa de transmissão.

Uma pesquisa a respeito das taxas de *download* (D) e *upload* (U) que são oferecidas atualmente por empresas brasileiras nas capitais, resultou nos dados da tabela 2.1:

Tabela 2.1: Tabela de taxas de *Download* (D) e *Upload* (U) banda larga brasileiras.

Empresa	Velocidade	Taxa D	Taxa U	Valor (α)
GVT [1]	15Mbps	15Mbps	1Mbps	0,06
Speedy [2]	8Mbps	8Mbps	600Kbps	0,07
Net Virtua [3]	20 Mbps	20 Mbps	1 Mbps	0,05
Oi Velox [4]	20 Mbps	20 Mbps	1 Mbps	0,05
TVA Telefonica AJATO [5]	100 Mbps	100 Mbps	3 Mbps	0,03

Assim, temos que a rede com maior assimetria é TVA Telefonica, pois esta tem um grau de assimetria (α) menor que as demais redes que foram pesquisadas.

2.3 Simuladores de Rede

Os simuladores de rede são de fundamental importância para verificar o desempenho de uma determinada rede. São usados para reduzir gastos e tempo, fornecer aos usuários uma visão do comportamento da rede ou dos protocolos que estão sendo simulados.

[1] http://www.gvt.com.br/portal/profissionaisliberais/servicosinternet/power/index.jsp acessado em 02 de fevereiro de 2013

[2] http:/http://www.telefonica.com.br/landing/telefonica.html acessado em 02 de fevereiro de 2013

[3] http://www.netcombo.com.br/netPortalWEB/appmanager/portal/desktop acessado em 02 de fevereiro de 2013

[4] Informações obtidas via atendimento telefônico direto com a operadora

[5] Informações obtidas via atendimento telefônico direto com a operadora

A simulação é um processo que visa o entendimento de um modelo real, realizando experimentos com este modelo com a finalidade de verificar e analisar o comportamento deste (SHANNON, 1998).

De acordo com Ingalls (2002) a simulação consiste dos seguintes elementos estruturais:

- Entidades: são os objetos que interagem entre si para revelar o comportamento do sistema. Exemplo: nós de computadores, fluxos de pacotes.
- Recursos: oferecem serviços para as entidades. Exemplo: largura de banda, número de servidores.
- Atividades e Eventos: a partir do envolvimento das entidades em atividades, têm-se a criação de eventos que causam alguma modificação no sistema. Exemplo: atrasos e filas.
- Escalonador: tem como função manter uma lista de eventos e seu respectivo tempo de execução durante a simulação.
- Variáveis Globais: tem seu acesso comum por qualquer função ou entidade envolvida no sistema, mantendo assim valores comuns da simulação. Exemplo: número de pacotes que estão sendo transmitido, comprimento da fila de pacotes.

A partir de resultados gerados pela simulação pode-se fazer um estudo estatístico para verificar o comportamento da rede em questão.

Segundo Portnoi e Araújo (2002), a partir de uma variação do *REAL Network Simulator* (KESHAV, 1997), surgiu o NS2 em 1989. Ele foi projetado nos Estados Unidos pela *Cornell University*, que posteriormente recebeu o apoio de outras universidades, como por exemplo a universidade de *Berkeley* e empresas

como a *Xerox PARC* (*Palo Alto Research Center*). Seu objetivo principal é fazer a simulação de eventos discretos relacionados à área de redes de computadores.

De acordo com Issariyakul e Hossain (2011), as características do simulador NS2 são: *open source*, orientados a eventos, projetado internamente na linguagem orientada a objetos C++. Sua interface com o usuário para definição dos parâmetros a serem usados é feita através da linguagem interpretada OTcl.

A linguagem OTcl faz a configuração dos cenários de toda a simulação, como por exemplo, configuração dos objetos que serão usados na rede a qual está sendo analisada. Ao final da simulação tem-se um arquivo *trace* que irá representar os eventos que ocorreram durante a respectiva simulação.

Portnoi e Araújo (2002) explicam que o motivo pelo qual tem-se o uso de duas linguagens neste simulador é que a linguagem C++ é usada internamente para manipulação de *bytes*, e para o usuário tem-se a linguagem interpretada OTcl. Essa linguagem interpretada tem um maior desempenho para mudanças de variáveis durante a simulação, como por exemplo, tamanho do enlace, sendo que neste caso se fosse usada a linguagem C++ teria que ser feito compilações de programas para chegar à mudança desejada.

As vantagens do simulador NS2 são:

- Seu núcleo interno poderá ser modificado de acordo com as necessidades do usuário. Pois seu código é aberto, realizando compilações nas respectivas classes que foram modificadas.
- É gratuito, fornece também suporte a muitas tecnologias de rede, podendo ser simulados diferentes cenários relacionados a protocolos de rede.

O NS2 apresenta, em contrapartida, uma alta curva de aprendizagem devido à complexidade na manipulação de configurações mais refinadas no código fonte.

2.4 Trabalhos Relacionados

Este capítulo irá tratar das técnicas que já foram propostas por outros autores. Estas técnicas fazem alguma alteração no protocolo TCP ou incluem algum mecanismo à transmissão com a finalidade de que o uso indevido da banda de canal de *upload* não interfira no desempenho no canal de *download*.

A técnica Redução de ACKs proposta por Balakrishnan, Padmanabhan e Katz (1999), busca diminuir a quantidade de ACKs que são enviados no canal reverso para tentar melhorar o desempenho de utilização do canal de *upload*. Se houver uma saturação no canal reverso, isso irá gerar perda de desempenho no canal de *download*.

O trabalho apresentado por Balakrishnan, Padmanabhan e Katz (1999), tenta reduzir a quantidade dos pacotes de confirmação (ACKs) no canal reverso, buscando assim amenizar o problema relacionado à saturação deste canal. Para que se tenha uma diminuição dos pacotes de confirmação que trafegam no canal reverso, o autor propõe que um ACK reconheça vários pacotes de dados.

A Figura 2.4 representa a topologia que foi proposta por (BALAKRISHNAN; PADMANABHAN; KATZ, 1999).

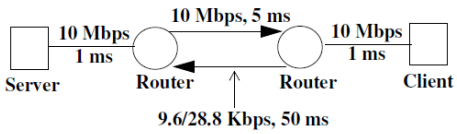


Figura 2.2: Topologia de simulação de uma rede Assimétrica (BALAKRISHNAN; PADMANABHAN; KATZ, 1999).

Balakrishnan, Padmanabhan e Katz (1999), propuseram duas soluções que buscam tratar os problemas do protocolo TCP em redes assimétricas: Controle de Congestionamento de ACKs (ACK *Congestion Control* –ACC) e Filtragem de ACK's (ACK *Filtering* –AF).

Controle de Congestionamento ACK- ACC: Essa técnica propõe que se tenha um mecanismo de controle de congestionamento no lado receptor (cliente), com a finalidade de controlar o envio dos reconhecimentos (ACKs). Para ter um gerenciamento de quantos dados serão reconhecido com apenas um ACK, tem-se definido uma variável denominada d, que corresponde à quantidade de dados que o receptor (cliente) irá receber, sendo que essa quantidade de dados será reconhecida com apenas um ACK, esta técnica considera que estes reconhecimentos(ACKs) são cumulativos.

Filtragem de ACKs – AF: Esta técnica é usada para que se tenha uma diminuição na quantidade de reconhecimentos (ACKs) que são enviados do receptor para o transmissor. A filtragem pode ser feita fazendo modificações no roteador do receptor. Ao verificar que um reconhecimento (ACK) está prestes a ser enfileirado no roteador, busca-se na fila ACKs anteriores a ele que pertencem à mesma conexão. Pressupondo-se que os ACKs são cumulativos, tem-se a eliminação no *buffer* dos ACKs mais antigos, liberando assim espaço no *buffer*.

Adaptação do remetente TCP- Sender Aptation (SA): As técnicas ACC e AF têm como objetivo controlar o número de reconhecimentos (ACKs) durante a transmissão dos dados. Elas fazem com que ocorra uma diminuição na quantidade de ACKs que são enviados durante a transmissão. Considerando do fato dos reconhecimentos serem cumulativos, tem-se que um determinado ACK poderá reconhecer uma quantidade de dados. Como foi visto anteriormente, essas técnicas podem gerar sérios problemas para o desempenho do protocolo TCP, como por exemplo, problemas relacionados à janela de transmissão de dados e ao mecanismo de transmissão rápida.

Com a técnica SA proposta por Balakrishnan, Padmanabhan e Katz (1999), tem-se estipulado um número de pacotes de dados que o remetente poderá transmitir ao receptor. Havendo assim um limite para transferência dos dados, se a janela

suportar mais dados que este limite, deve-se respeitar a quantidade de dados que foi definida neste.

O objetivo principal desta técnica é que o protocolo TCP opere com sucesso mesmo quando se tenha uma diminuição de reconhecimentos (ACK). Pode ser enviadas rajadas de dados à rede, sendo que estas rajadas deverão respeitar uma determinada taxa para envio de dados. A taxa é calculada por cwnd/srtt, onde cwnd corresponde ao tamanho da janela de congestionamento e srtt é a estimativa de RTT. Assim, uma grande rajada de dados é quebrada em partes menores, e cada uma destas partes é enviada em um determinado período de tempo.

Com essa técnica irá ocorrer uma diminuição do problema da retransmissão rápida. Pois o remetente não irá precisar contar o número de ACK recebidos para aumentar sua janela de transmissão, mas sim a quantidade de dados que foram reconhecidos por um determinado ACK.

2.5 Modelo Matemático

Araújo (2009) propôs um modelo matemático para analisar o desempenho do canal de *download* em relação ao canal reverso. Este modelo é relacionado a uma rede com bandas assimétricas transmitindo dados nos dois sentidos. No desenvolvimento do modelo desconsideraram-se atrasos relacionados às transmissões, efeitos do crescimento das janelas de transmissão e o uso de compressões de pacotes.

Em uma transmissão sendo realizada nos dois sentidos, têm-se pacotes de dados e de reconhecimentos (ACKs) disputando o mesmo canal. Na ocorrência disto, Araújo (2009) considerou que os dados têm preferência de serem enviados em detrimento aos pacotes de reconhecimentos. A ocupação do respectivo canal poderá ser obtida pela quantidade total de dados que está sendo transmitidos em relação à capacidade total do canal.

No modelo escolhido considera-se que cada ACK irá gerar um pacote de reconhecimento como resposta ao recebimento dos dados com sucesso. Poderá também ser feito um controle da fração do canal reverso a qual será usada para enviar os dados.

Os cenários para representação do modelo proposto por Araújo (2009) é composto por duas transmissões unidirecionais em sentidos opostos e uma transmissão bidirecional.

As simulações para coleta e análise dos dados desta pesquisa, será realizada em relação ao cenário da Figura 2.3.

Duas transmissões unidirecionais em sentidos opostos: Como mostra a figura 2.3, temos a transmissão de dados pelos clientes em dois pontos distintos da rede.

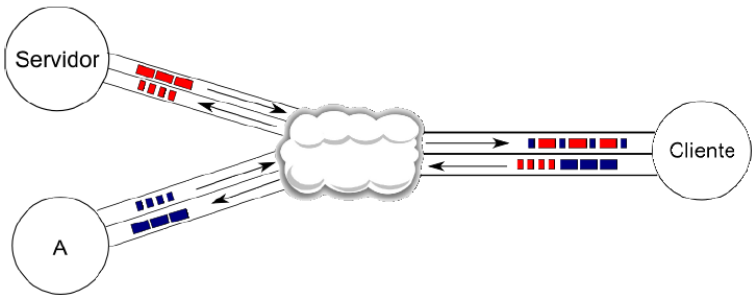


Figura 2.3: Transmissão do cliente com dois pontos distintos da rede (Araújo, 2009).

Araújo (2009), propôs a equação 2.3, a qual representa o cálculo do valor de β onde a taxa de *download* começará a ser afetada em função da variável k proposta por Balakrishnan, Padmanabhan e Katz (1999).

$$\beta = 1 - k \tag{2.3}$$

Como exemplo considere uma rede com taxa de *download* de 1,5Mbps e 300 Kbps de *upload*, os pacotes de dados com tamanho de 1500 *bytes* e os pacotes de confirmação de 60 *bytes*.

Considerando a equação proposta por Balakrishnan, Padmanabhan e Katz (1999), que é definida em relação k= (LBD/LBU) / (TPD/TPA). Sendo LBD: largura de banda de *download*, LBU: largura de banda de *upload*, TPD: tamanho do pacote de dados, TPA: tamanho do pacote de ACKs. Teremos como resultado, k=0,2. Assim, β=0,8 , quando tem-se 80 % do canal reverso sendo utilizado, teremos perdas na taxa de *download*.

Uma Transmissão bidirecional: Este cenário é ilustrado na figura 2.6, sendo que os dados são transmitidos do emissor para o receptor e vice-versa, estes dois pontos da rede enviam e recebem dados pelo mesmo canal. Considerou-se o uso do mecanismo *piggybacking* como forma de confirmação dos recebimentos dos pacotes de dados pelos respectivos canais.

Com o uso deste mecanismo de *piggybacking*, os pacotes de dados que são enviados pelo canal de *upload* serão utilizados como forma de reconhecimento (ACK) dos pacotes de dados que chegaram pelo canal de *download*.

Figura 2.4: Transmissão bidirecional entre cliente e servidor (Araújo, 2009).

Neste cenário considerou-se que todo o canal de *upload* será utilizado. Araújo (2009), verificou que uma redução nos tamanhos dos pacotes no canal reverso aumenta a taxa de *download* devido à maior quantidade de reconhecimentos (ACKs) que poderão ser suportadas no canal reverso.

Os resultados obtidos por Araújo (2009) para este cenário foram:

Através do valor α (fração entre as bandas de *upload* e *download*), tem-se que a fração de uso da banda de *upload* β, será obtida de acordo com a equação 2.4. Considera-se pA: tamanho dos pacotes de reconhecimentos, Pd: tamanho dos pacotes de dados no canal de *download*.

$$\beta = \frac{1}{\alpha}\left(\frac{p_A - \alpha p_d}{p_A - p_d}\right) \tag{2.4}$$

Um exemplo é quando se tem pacotes de reconhecimentos de 60 *bytes* e 1500 *bytes* de pacotes de dados, sendo considerando bandas de 8Mbps de *download* e 1Mbps de *upload*, ocorrerá uma saturação no canal de *upload* quando β =0,407.

Quando se tem variações nos tamanhos dos pacotes do canal reverso, o valor de saturação é dado pela equação 2.5. Considera-se pu: tamanho dos pacotes de dados no canal de *upload*.

$$\beta = \left(\frac{p_u p_A - \alpha p_u p_d}{\alpha p_d (p_A - p_u)}\right) \tag{2.5}$$

Verificou-se que a redução dos pacotes do canal reverso irá fazer com que o valor de β tenha melhor resultados para ambientes com maior assimetria.

A partir de resultados obtidos por Araújo (2009), teremos que:

- A taxa de utilização do canal de *upload* irá influenciar diretamente no desempenho do canal de *download* nos cenários analisados acima. Pois o canal reverso precisa enviar pacotes de confirmação, que irão ocupar parte da banda, e que podem sofrer restrições caso o canal de *upload* esteja sendo usado para envio de dados.

- O mecanismo de *piggybacking* irá agravar a perda de desempenho na medida que os pacotes de dados, quando estivem preenchendo o canal, não conseguirem confirmar todos os pacotes de dados de *download*.

3 METODOLOGIA

Nesta seção iremos tratar da forma como essa pesquisa foi desenvolvida, buscando explorar todos os pontos que foram seguidos até que o objetivo seja alcançado.

3.1 Tipo de Pesquisa

A forma na qual foi desenvolvida essa pesquisa pode ser classificada seguindo as características propostas por Jung (2004):

Em relação à natureza ela será uma pesquisa básica, pois será buscado a partir de conhecimentos sobre o protocolo TCP relacionados às redes assimétricas, validar o modelo matemático proposto por Araújo (2009). Quanto aos objetivos, essa pesquisa será descritiva, pois serão usadas simulações como forma de análise do comportamento do protocolo TCP em redes assimétricas. Quanto aos procedimentos, essa pesquisa é experimental, pois serão realizadas simulações de cenários. Será também uma pesquisa de laboratório, pois poderemos escolher a melhor taxa de transmissão de dados que estão sendo usadas pelas empresas prestadoras de serviço atualmente para realizar as devidas simulações.

3.2 Procedimentos Metodológicos

Durante o desenvolvimento deste trabalho, foi realizado um estudo a respeito das características do protocolo TCP. Posteriormente verificando seu comportamento em relação as redes assimétricas, com foco voltado para a área de Redes de Computadores.

Foi realizado também um estudo a respeito de trabalhos relacionados que propuseram uma solução para o problema em questão. Sendo realizado também um

estudo a respeito do simulador NS2, verificando suas características e como será realizado o desenvolvimento das arquiteturas a serem simuladas.

Para verificação do comportamento da rede, foram realizadas simulações de 4 situações para um cenário - Tabela 3.1. As situações serão chamadas de Rede1, Rede 2, Rede 3 e Rede 4. As respectivas taxas de *upload* e *download* correspondem a tabela 3.1:

Tabela 3.1: Tabela dos cenários/taxas de transmissão.

Situação	Velocidade de *Download*(D)	Velocidade de *Upload*(U)
Rede 1	8Mbps	1Mbps
Rede 2	4Mbps	256Kbps
Rede 3	1.5Mbps	300 Kbps
Rede 4	256Kbps	64Kbps

A simulação da rede assimétrica através do Network Simulator foi totalmente implementada na liguagem OTcl, sendo denominada monografia.tcl (Anexo A). A rede assimétrica foi projetada através de *links*, nós e aplicações para transferência dos dados. A partir de simulações dos cenários, foi obtido a *Network Animator* (NAM) - Figura 3.1, que corresponde a uma animação gráfica do respectivo cenário simulado.

A arquitetura simulada é composta por 4 nós. O nó 2 simula o roteador da rede. Os dados são enviados do nó 0 para o nó 3, do nó 1 para o nó 3, do nó 3 para o nó 0. As taxas de transmissão dos enlaces são correspondentes a cada cenário a ser simulado.

Ao executar o *script* da simulação, é gerado o arquivo *Trace*. Este arquivo descreve toda a troca de informação realizada pelos *hosts*. Abaixo na Figura 3.2, está um trecho do *trace* de um pacote enviado do nó 2 para o nó 3:

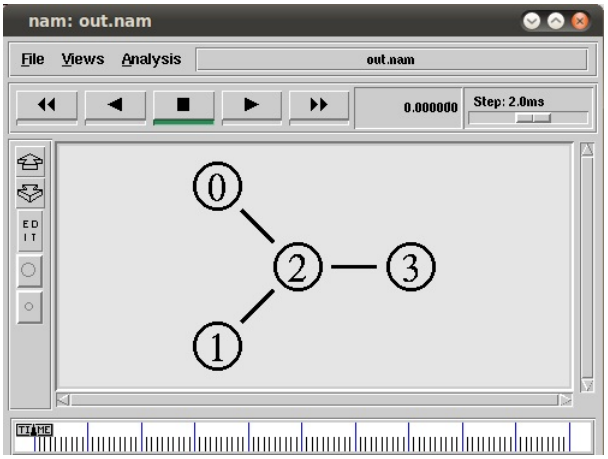


Figura 3.1: Tela de execução da Simulação.

```
+ 0.1 2 3 tcp 1540 ------- 2 1.0 3.1 1 6
- 0.12 2 3 tcp 1540 ------- 2 1.0 3.1 1 6
r 0.5 2 3 tcp 1540 ------- 2 1.0 3.1 1 6
```

Figura 3.2: Saída do arquivo tracer.

Cada linha do *trace* irá corresponder a um determinado evento. O primento campo define o tipo de evento ocorrido, o segundo é o tempo na qual o evento ocorreu, o terceiro corresponde ao nó de origem e nó de destino do pacote, o quarto corresponde ao tipo de pacotes, o quinto ao tamanho do pacote.

Por exemplo, a segunda linha da Figura 3.2, mostra que o pacote enviado do nó 2 para o nó 3 saiu da fila do roteador no tempo de 0.12 segundos.

Em cada cenário que foi simulado, foram realizadas cem variações aleatórias do tempo de *upload*. A diferença entre esses tempos corresponde a uma janela de transmissão de 5 segundos.

Como forma de analisar os dados obtidos, foi calculada a vazão entre os pacotes enviados do nó 2 para o nó 3. A vazão corresponde a quantidade de pacotes recebidos pelo nó 3 e o tamanho da janela de transmissão considerada durante a

execução da simulação, conforme equação 3.1 . Para análise dos resultados, foram consideradas cem variações de tempos aleatórios do início do *upload*. Posteriormente foi calculada a utilização do *link*, correspondente a quantos *bits per second* (bps) foi usada em relação a capacidade total da rede.

$$Vazão = \frac{Número\ de\ Pacotes\ Recebidos}{Tamanho\ da\ Janela\ de\ Transmissão} \tag{3.1}$$

Foi calculado também o valor de α (grau de assimetria entre os canais), este é obtido através da razão entre a capacidade de *Upload* (U) e *Download* (D), conforme equação 3.2 . O objetivo deste cálculo é verificar a influência da assimetria em relação ao desempenho que é obtido pela respectiva rede analisada.

$$\alpha = \frac{U}{D} \tag{3.2}$$

3.3 Arquitetura do Sistema Simulado

A topologia a ser simulada visando avaliação e comparação do modelo matemático proposto por Araújo (2009), é composta por duas transmissões unidirecionais em sentidos opostos.

Este cenário é representada na Figura 3.3. Os pacotes de dados são transmitidos pelos clientes que estão em dois pontos distintos da rede.

A implementação desta arquitetura no NS2 foi realizada conforme a Figura 3.4. Os links que representam os canais de *upload* e *download* foram configurados como simplex, ou seja os dados são carregados em apenas um sentido. As taxas de transmissões foram configuradas conforme a característica da rede a ser simulada. Foram implementadas aplicações *File Transfer Protocol* (FTP) para transferências de dados entre os nós da rede. Os dados são transmitidos do nó 3 para o nó 0, do

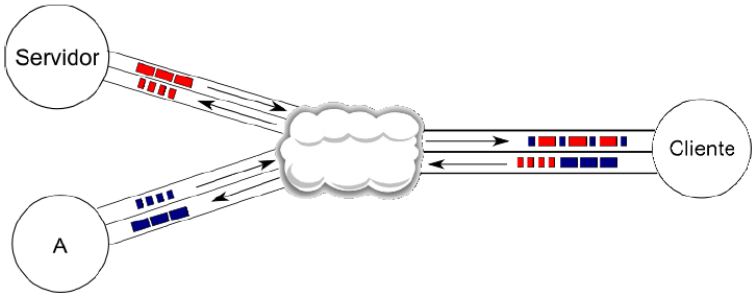


Figura 3.3: Transmissão do cliente com dois pontos distintos da rede (Araújo, 2009).

nó 0 para o nó 3 e do nó 1 para o nó 3. Foi implementada um função para gerar tempos aleatórios para o início da aplicação FTP 1, a qual representa o *upload* de dados realizados pelo nó 3. Foram realizadas modificações na semente do NS2 para que fosse geradas soluções diferentes a cada execução dos cenários.

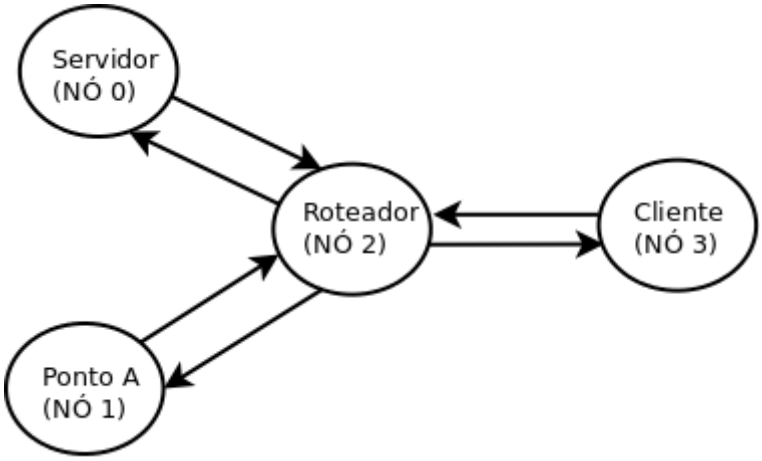


Figura 3.4: Projeto da Arquitetura a ser simulada no NS2.

4 RESULTADOS E DISCUSSÃO

O objetivo deste capítulo é apresentar os resultados obtidos nas simulações do comportamento do protocolo TCP em redes assimétricas. Após as simulações das quatro configurações para o cenário escolhido, foi realizada uma análise para verificar a influência da saturação do canal de *upload* em relação ao *download*.

Os resultados alcançados durante as simulações dos cenários corresponde são apresentados na Tabela 4.1. A primeira coluna representa as análises que foram realizadas para cada cenário, as respectivas colunas correspondem a média entre as vazões, o desvio padrão e o intervalo de confiança entre os dados.

Tabela 4.1: Tabela de Médias e Desvio Padrão do Cenário

Análise dos Cenários	Rede 1	Rede 2	Rede 3	Rede 4
Média	649,11	323,68	121,636	20,722
Desvio Padrão	1,01	1,43	0,66	0,52
Intervalo de Confiança:	LI: 648,91	LI: 323,39	LI: 121,50	LI: 20,61
	LS: 649,30	LS: 323,96	LS: 121,76	LS: 20,82

O cálculo do desvio padrão foi realizado para verificar o quanto os valores da vazão estão sofrendo variações em relação à média obtida.

A Tabela 4.2, corresponde ao cálculo do valor de α (grau de assimetria entre os canais) e a utilização do *link* de *Download* (D) para cada cenário, estes valores das utilizações foram retirados a partir dos resultados das vazões em cada cenário em relação à capacidade de cada *link* simulado.

A partir do resultados obtidos pelo cálculo do valor α, Tabela 4.2, verificou-se que quanto maior o valor de α maior será o desempenho da rede, pois esta terá uma menor assimetria entre os canais de *upload* e *download*. Este resultado foi

Tabela 4.2: Tabela dos valores de α e utilização do link de D em cada Situação.

Situação	Valor de α	Percentual da Utilização do Link D
Rede 1	0,125	46%
Rede 2	0,0625	39%
Rede 3	0,195	77%
Rede 4	0,25	98%

verificado através da análise da utilização do *link* de *download* em cada situação que foi simulada.

A comparação dos resultados dessa pesquisa com os resultados obtidos por Araújo (2009) é representado pela Figura 4.1, que mostra a variação da taxa de download para algumas bandas mais comuns. Araújo (2009) realizou o cálculo do valor da constante de assimetria (K) para cada uma das redes, posteriormente foi calculado o valor de β onde a taxa de download começará a ser afetada em função de k.

Como exemplo, calculando o valor da constante k para a rede 4, teremos que k= 0,16. Assim, pelo modelo matemático proposto por Araújo (2009) equação 2.3, teremos que β = 84 %. Ou seja, quando tivermos 84% do canal reverso sendo utilizado, teremos perdas na taxa de *download*. A tabela 4.3 contém os valores de β e as respectivas utilizações do link de upload em relação a quantidade de pacotes que foram trafegados neste *link*.

Comparando os dois resultados, é verificado que os resultados não são condizentes, as utilizações dos *links* de *upload* obtidas com este trabalho, diferem do modelo matemático proposto por Araújo (2009) . O motivo dessa divergência pode ser o fato de não ter realizado a implementação no NS2 da prioridade dos ACKs em relação aos pacotes de dados.

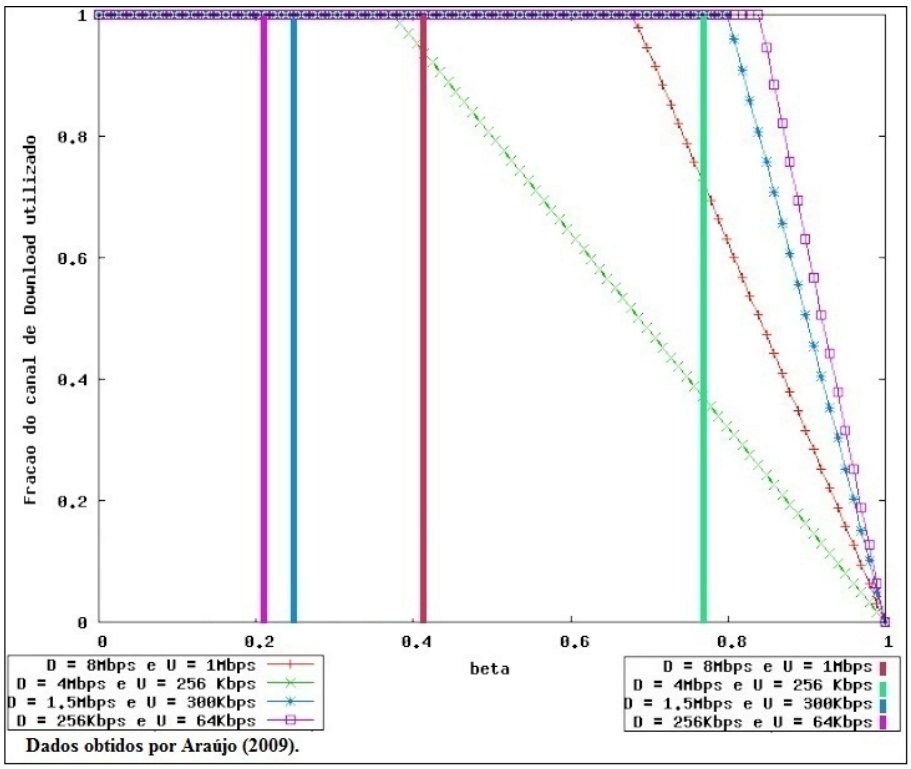


Figura 4.1: Transmissão do cliente com dois pontos da rede (Araújo, 2009).

Tabela 4.3: Tabela dos valores de β e utilização do link de U em cada Situação.

Situação	Valor de β	Percentual da Utilização do link de U
Rede 1	0,80	40,8
Rede 2	0,59	78,9
Rede 3	0,87	25,3
Rede 4	0,90	20,2

5 CONCLUSÕES E TRABALHOS FUTUROS

Foi possível através deste projeto realizar um grande estudo e por consequência um enorme aprendizado a respeito das redes ADSL, protocolo TCP e simulador NS2.

Durante o desenvolvimento deste trabalho, foi buscado realizar a implementação do cenário em outros simuladores, porém selecionamos o simulador NS2. Através do NS2 foi desenvolvida a criação das situações para o respectivo cenário e posteriormente realizada a verificação e análise do modelo matemático.

Foi analisado o arquivo de saída do NS2, e com isso obteve-se valores como a vazão em cada cenário, utilização do *link* em cada rede, cálculo das médias da vazão e utilização do *link*, desvio padrão e intervalo de confiança. Através da análise de todos esses resultados obtidos, foi possível constatar que quanto maior o valor de α (grau de assimetria entre os canais), maior será o desempenho da rede.

Um dos problemas encontrados para verificação da vazão na rede foi o escalonador de pacotes do simulador NS2, que gerou sérios problemas como descarte de pacotes e atrasos durante a transmissão.

Os resultados obtidos não condiz de forma suficiente com o modelo matemático que foi proposto pelo autor Araújo (2009), pois não foi implementado no NS2 a priorização dos ACKs em relação aos pacotes de dados. Foi possível verificar que o canal de *upload* influencia diretamente no desempenho do canal de *download*. As redes ADSL não são adequadas quando tem-se grande assimetria entre seus links, pois haverá perdas de pacotes de dados, atrasos durante a transmissão e problemas para o canal de *download*.

Com os resultados obtidos com este trabalho, surge a oportunidade de buscar a validação do modelo matemático proposto por Araújo (2009) através de outros cenários, realizar o refinamento da simulação para considerar outras variavéis, testar outros simuladores de redes para se chegar a um resultado desejado.

REFERÊNCIAS BIBLIOGRÁFICAS

Araújo, E. F. M. **Uma Análise dos Problemas do Protocolo TCP em Redes Assimétricas e soluções existentes**. Dissertação (Mestrado em Ciência da Computação) - Universiade Federal de Minas Gerais, Belo Horizonte. 2009.

BALAKRISHNAN, H.; PADMANABHAN, V.; KATZ, R. **The effects of asymmetry on TCP performance**. *Mobile Networks and Applications*, Springer, v. 4, n. 3, p. 219–241, 1999.

HUSTON, G. **TCP Performance**. *The Internet Protocol Journal*, Citeseer, v. 3, n. 2, p. 2–24, 2000.

INGALLS, R. **Introduction to simulation**. In: IEEE. *Winter Simulation Conference*. [S.l.], 2002. v. 7, p. 7–16.

ISSARIYAKUL, T.; HOSSAIN, E. ***Introduction to Network Simulator NS2***. [S.l.]: Springer Verlag, 2011.

JACOBSON, V. **Congestion avoidance and control**. In: ACM. *ACM SIGCOMM Computer Communication Review*. [S.l.], 1988. v. 18, n. 4, p. 314–329.

JUNG, C. **Metodologia para pesquisa & desenvolvimento aplicada a novas tecnologias, produtos e processos**. *Rio de Janeiro : Axcel Books*, 2004.

KESHAV, S. **REAL 5.0 Overview**. *Cornell University. Disponível em: http://www.cs.cornell.edu/skeshav/real/index.html. Acessado em: 22 out 2012*, 1997.

KUROSE, J. **Redes de Computadores e a Internet:** uma abordagem top-down. 5. ed. São Paulo : Pearson. 2010.

PORTNOI, M.; ARAÚJO, R. **Network Simulator–visão geral da ferramenta de simulação de redes**. *SEPA-Seminário Estudantil de Produção Acadêmica. Salvador*, p. 173–181, 2002.

SHANNON, R. E. **Introduction to the art and science of simulation**. In: IEEE. *Simulation Conference Proceedings on Winter*. [S.l.], 1998. v. 1, p. 7–14.

TANENBAUM, A. **Redes de Computadores**. 4 ed. Rio de Janeiro: Campus. 2003.

TOLEDO, A. P. **Redes de acesso em telecomunicações**. São Paulo, Ed. Makron Books, 2001.

A ANEXO A - ARQUIVO MONOGRAFIA.TCL

```
# Criação de um objeto simulador.
set ns [new Simulator]

    #Criando um arquivo para o trace.
set mytrace [open out.tr w]
$ns trace-all $mytrace

    # Criando um arquivo para nam.
set myNAM [open out.nam w]
$ns namtrace-all $myNAM

    # Arquivo para armazenar os dados coletados sobre a vazão
set throughput [open gsimple.plt w]

    # Define o procedimento fim
proc fim
global ns mytrace throughput myNAM
$ns flush-trace
close $mytrace
close $myNAM
close $throughput
puts »»»»»> «««««««««"
puts "****DADOS ANALISADOS COM SUCESSO...****"
puts »»»»»> «««««««««<"
exec nam out.nam &
exit 0
}
```

```
# Criação dos nós.
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Definindo as cores dos fluxos:
$ns color 1 blue
$ns color 2 red
$ns color 3 orange

# Declaração dos enlaces para o Cenário 1 - Rede 1.
$ns simplex-link $n0 $n2 8Mb 10ms DRR
$ns simplex-link $n2 $n0 8Mb 10ms DRR
$ns simplex-link $n1 $n2 8Mb 10ms DRR
$ns simplex-link $n2 $n1 8Mb 10ms DRR
$ns simplex-link $n2 $n3 8Mb 10ms DRR
$ns simplex-link $n3 $n2 1Mb 10ms DRR

# Agrupamento (posição) dos nós na Nam.
$ns simplex-link-op $n0 $n2 orient right-down
$ns simplex-link-op $n1 $n2 orient right-up
$ns simplex-link-op $n2 $n3 orient right-center

#Monitorar a fila do canal (n2-n3) para a Nam.
$ns simplex-link-op $n2 $n3 queuePos 0.5

#Definição do limite da fila no roteador entre os dois nós.
$ns queue-limit $n2 $n3 500

# Criação da conexão TCP, na qual o no 3 envia dados para o no 0;
set tcp1 [new Agent/TCP]
```

```
$tcp1 set class_ 1
$tcp1 set packetSize_ 1500
$ns attach-agent $n3 $tcp1
set tcpSink1 [new Agent/TCPSink]
$ns attach-agent $n0 $tcpSink1
$ns connect $tcp1 $tcpSink1

    # Criação da conexão TCP, na qual o no 1 envia dados para o no 3;
set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
$tcp2 set packetSize_ 1500
$ns attach-agent $n1 $tcp2
set tcpSink2 [new Agent/TCPSink]
$ns attach-agent $n3 $tcpSink2
$ns connect $tcp2 $tcpSink2

    # Criação da conexão TCP, na qual o no 0 envia dados para o no 3;
set tcp3 [new Agent/TCP]
$tcp3 set class_ 3
$tcp3 set packetSize_ 1500
$ns attach-agent $n0 $tcp3 set tcpSink3 [new Agent/TCPSink]
$ns attach-agent $n3 $tcpSink3 $ns connect $tcp3 $tcpSink3

    # Configurando os fluxos da aplicação FTP
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
```

```
    # Criação de um procedimento que irá calcular a taxa
de download em relação ao nó 3 (neste caso tcpsink3 e tcpsink2)
proc record {} {
global throughput tcpSink2 tcpSink3
# Obtem uma instância do objeto simulador.
set ns [Simulator instance]
# Configura o intervalo de chamada recursiva.
set time 0.1
# Calcula o número de bytes recebidos pelo Sink em determinado instante.
set bwSinkA [$tcpSink3 set bytes_]
set bwSinkB [$tcpSink2 set bytes_]
# Mostra a quantidade de bytes recebidos pelo no 3.
# puts "uantidades de bytes recebidos pelo no 3 é : [$tcpSink3 set bytes_]"
# Obtem o tempo corrente.
set now [$ns now]
# Calcula a largura de banda (em Mb/s) e as escreve no arquivo
puts $throughput "$now [expr ($bwSinkA+$bwSinkB)/$time*8/1000000]"
# inicializa as variaveis.
$tcpSink3 set bytes_ 0
$tcpSink2 set bytes_ 0
# na linha abaixo é a chamada recursiva.
$ns at [expr $now+$time] "record"
}

    $ns at 0.0 "record"

    # Criação da Semente no NS2.
global defaultRNG
$defaultRNG seed 9999
```

```
    # Criação da semente RNG.
set arrivalRNG [new RNG]
set sizeRNG [new RNG]
for set j 1 $j < 10 incr j
$arrivalRNG next-substream
$sizeRNG next-substream
}

    # Invervalo para geração de números aleatórios.
set size_ [new RandomVariable/Uniform]
$size_ set min_ 0
$size_ set max_ 5
$size_ use-rng $sizeRNG

    # Impressão dos 6 números aleatórios gerados.
for set j 0 $j < 6 incr j
#puts [format "%-8.1f"[$size_ value]]
}

    #Configuração para Inicio e Fim do upload.
set t 0
for set i 0 $i<100 set i [expr $i+1]
set n($i) [$size_ value]
#gerando 100 numeros aleatórios entre 0 a 5
puts [format "%-8.1f"[$size_ value]]
set t [expr $t + $n($i)]
#Controle Inicio e Fim do Upload. $ns at t "$ftp1 start";
$ns at t+5 "$ftp1 stop";
set t [expr $t + 5]

    }
```

```
    # Períodos de Transmissão e execução das aplicações ftps.
$ns at 0.1 "$ftp2 start";
$ns at 0.1 "$ftp3 start";
$ns at 1500.0 "$ftp2 stop";
$ns at 1500.0 "$ftp3 stop";

    # Executa o procedimento fim, finalisando a respectiva simulação.
$ns at 1500.0 "fim"

    # Execução da simulação
$ns run
```