COPPE/UFRJ

APLICAÇÃO DE ALGORITMOS DE CONTROLE PARA O TRATAMENTO DE CONGESTIONAMENTO EM REDES DE COMPUTADORES

Diana Beatriz Benítez Cáceres

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica

Orientadores:

Prof. Eugenius Kaszkurewicz

Amit Bhaya

Rio de Janeiro

Maio 2010

APLICAÇÃO DE ALGORITMOS DE CONTROLE PARA O TRATAMENTO DE CONGESTIONAMENTO EM REDES DE COMPUTADORES

Diana Beatriz Benítez Cáceres

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS–GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIENCIAS EM ENGEHARIA ELÉTRICA.

Examinada por:

_____

Prof. Eugenius Kaszkurewicz, D.Sc.

_____

Prof. Amit Bhaya, Ph.D.

_____

Prof. Fernando Cesar Lizarralde, D.Sc.

_____

Prof. José Ferreira de Rezende, Dr.

_____

Prof. Virgílio Augusto Fernandes de Almeida, Ph.D.

_____

Prof. José Roberto Boisson de Marca, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2010

Benítez Cáceres, Diana Beatriz

Aplicação de Algoritmos de Controle para o tratamento de congestionamento em Redes de Computadores / Diana Beatriz Benítez Cáceres. – Rio de Janeiro: UFRJ/COPPE, 2010.

XIX, 140 p.; 29,7 cm.

Orientadores: Eugenius Kaszkurewicz, Amit Bhaya

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Elétrica, 2010.

Referências Bibliográficas: p. 124-131

1. Internet. 2. Controle de congestionamento. 3. Otimização. 4. Protocolos TCP. 5. I. Kaszkurewicz, Eugenius, et.al. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À esperança na vida, nas pessoas e na felicidade
fontes de inspiração e motivo da minha força, insistência e persistência.

Assim mesmo

Muitas vezes as pessoas são egocêntricas, ilógicas e insensatas.
Perdoe-as assim mesmo.
Se você é gentil, as pessoas podem acusá-la de egoísta, interesseira.
Seja gentil assim mesmo.
Se você é vencedora, terá alguns falsos amigos e alguns inimigos verdadeiros.
Vença assim mesmo.
Se você é honesta e franca, as pessoas podem enganá-la.
Seja honesta e franca assim mesmo.
O que você levou anos para construir,
alguém pode destruir de uma hora para outra.
Construa assim mesmo.
O bem que você faz hoje pode ser esquecido amanhã.
Faça o bem assim mesmo.
Dê ao mundo o melhor de você, mas isso pode nunca ser o bastante.
Dê o melhor de você assim mesmo.

MADRE TERESA DE CALCUTÁ

Agradecimentos

Ao terminar esta tese de doutorado gostaria registrar meus nossos sinceros agradecimentos às pessoas que de várias formas contribuíram para que se tornasse numa realidade.

Para maior percepção desse sentido devo contar que esta não foi uma caminhada breve, mas uma travessia que parecia sem fim, principalmente pelas intercorrências pessoais de toda ordem, que me atropelaram. Esses percalços, longe de obscurecerem o trajeto, aumentaram-lhe o brilho. E, ao invés de me deterem, em certa forma, impulsionaram-me com mais força.

É assim que gostaria agradecer:

Ao amor da minha vida, pela paciência nestes últimos anos.

Ao meu pai, que sempre foi um exemplo de honestidade e bondade. À minha mãe, que sempre incentivou com grande entusiasmo os meus projetos acadêmicos.

A minha família e amigos de Paraguai que sempre confiaram no meu esforço e na minha capacidade de superação.

Ao meu orientador, prof. Eugenius Kaszkurewicz, pelo inestimável apoio, pelos conselhos, e pelo seu tempo, que foram determinantes para o sucesso deste trabalho.

Ao meu co orientador, o prof. Amit Bhaya, pelo grande apoio, pela confiança, e pelas suas orientações no meu passo pela UFRJ e no desenvolvimento do deste trabalho.

As pessoas da secretaria da elétrica da COPPE especialmente Daniele Oliveira da Silva; as pessoas do NACAD especialmente para Mara Prata e finalmente mais não menos importante à Sra. Marcia Valerio e as pessoas da FINEP que sempre me receberam com cordialidade.

Aos meus professores, amigos e colegas da COPPE/UFRJ, que sempre foram apoio nos anos de estudo e na realização deste trabalho, especialmente aos amigos Judith, Carina, Reinaldo, Gisela, Lucia, Magna e Christian que sempre estiveram à disposição quando precisei.

Ao CNPQ pelo suporte financeiro concedido através da bolsa de estudo, o qual permitiu a implementação do presente trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

APLICAÇÃO DE ALGORITMOS DE CONTROLE PARA O TRATAMENTO DE CONGESTIONAMENTO EM REDES DE COMPUTADORES

Diana Beatriz Benítez Cáceres

Maio/2010

Orientadores: Prof. Eugenius Kaszkurewicz

Prof. Amit Bhaya

Programa: Engenharia Elétrica

O crescimento acelerado da Internet nas ultimas décadas traz como conseqüência o congestionamento da mesma. Na atualidade o controle de congestionamento na Internet baseia-se em algoritmos de ajuste binário.

Este trabalho apresenta e analisa novas propostas de controle de congestionamento. O objetivo deste trabalho é a utilização de algoritmos de controle para modelar e solucionar o problema de controle de congestionamento. Primeiramente, são introduzidas duas técnicas baseadas numa seleção mais adequada dos parâmetros de incremento e decremento das dinâmicas do algoritmo AIMD, tradicionalmente utilizado, segundo equilíbrios virtuais. Neste contexto são propostos dois novos protocolos do estilo TCP que controlam o congestionamento de forma mais eficiente e equitativa em demérito das retransmissões utilizando para isto uma estimativa da largura de banda e do número de usuários que compartilham a rede.

Seguidamente, este trabalho trata do problema de controle de congestionamento sob a perspectiva de controle, propondo-se dos possíveis esquemas baseados em otimização obtendo-se uma melhor convergência considerando equidade e completa utilização da capacidade do canal. O preço a ser pago, pela melhora obtida, é que, além da informação de realimentação binária, cada usuário deve conhecer as taxas de envio dos vizinhos. Resultados de simulações mostram que o tempo de resposta e a amplitude das oscilações são melhorados, inclusive na presença de atrasos e capacidades variáveis.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

APPLICATION OF CONTROL ALGORITHMS FOR TREATMENT OF CONGESTION IN COMPUTER NETWORKS

Diana Beatriz Benítez Cáceres

May/2010

Advisors: Prof. Eugenius Kaszkurewicz

Prof. Amit Bhaya

Department: Electrical Engineering

The speed up growth of the Internet in the last few decades brings as consequence congestion problems. Current Internet congestion control relies on binary adjustment algorithms.

This work presents and analyzes new proposals of congestion control. The objective of this work is the use of control algorithms to model and to solve the problem of congestion control. First, two techniques based on a more appropriated selection of the parameters of increment and decrement of the dynamic of the AIMD algorithm are introduced according to virtual equilibria. In this context, this work proposes two new TCP-like protocols which control the congestion in a more efficient and fairness way, trading off with a larger number retransmissions. The new protocols use an estimation of the bandwidth and an estimate of the number of users sharing the network.

This work also studies the congestion control problem from a control perspective, starting from variants of this classical algorithm in order to propose two optimization-based schemes, which obtains faster convergence to a small neighborhood of the optimal fair full utilization state. The price to be paid for this improvement is that, in addition to the binary feedback information, each source must have knowledge of the sending rates of its neighbors. Simulation results are presented showing that responsiveness and smoothness are improved, even in the presence of delays and time-varying link capacity.

Sumário

Nomenclatura

A_AIMD_G: Formulação assíncrona (ou com retardos heterogêneos) de Gorinsky para *Additive Increase Multiplicative Decrease*.

AIMD: *Additive Increase Multiplicative Decrease*

Alg_OptAbs: Algoritmo Otimização Absoluta

Alg_OptQua: Algoritmo Otimização Quadrática

Alg_VE: Algoritmo Equilíbrios Virtuais

Alg_WVE: Algoritmo Westwood Equilíbrios Virtuais

AQM: *Active Queue Management*

S_AIMD_CJ: Formulação original síncrona (ou com retardos homogêneos) de Chiu e Jain para *Additive Increase Multiplicative Decrease*.

TCP: Protocolo de Controle de Transmissão

TCP_NR: Protocolo de Controle de Transmissão New Reno

TCP_VE: Protocolo de Controle de Transmissão Equilíbrios Virtuais

TCP_W: Protocolo de Controle de Transmissão Westwood

TCP_WVE: Protocolo de Controle de Transmissão Westwood Equilíbrios Virtuais

VE: Equilíbrios Virtuais

Índice de Figuras

Índice de Tabelas

1. Introdução.

Os usuários que utilizam uma rede de comunicação compartilham a largura de banda disponível. Quando a carga enviada pelos usuários é maior que a capacidade da rede ocorre o congestionamento e a Internet não é uma exceção à regra. Assim, os usuários devem reagir ao congestionamento adaptando suas taxas de transmissão, para evitar o colapso causado pelo congestionamento e manter uma eficiente utilização da rede.

Quando ocorre o congestionamento e pacotes são descartados, tem-se a diminuição da vazão e o aumento do atraso fim-a-fim dos tráfegos. O usuário, ao identificar que a emissão de dados está causando congestionamento, não deve enviar novos pacotes à rede, até que os pacotes enviados anteriormente sejam entregues.

O controle de congestionamento pode ser realizado no nível da camada de rede. No entanto, mesmo que esta camada tente controlá-lo, o trabalho mais pesado é realizado pela camada de transporte, já que a verdadeira solução para o problema é diminuir a taxa de transmissão de dados.

Embora, vários algoritmos para controle de congestionamento e suas correspondentes implementações em protocolos tenham sido apresentados nos últimos anos (LOW et.al., 2002, RYU et.al, 2003), atualmente, o controle de congestionamento na Internet baseia-se em algoritmos de ajuste binário (BANSAL e BALAKRISHNAN, 2001), que se encontram no Protocolo de Controle de Transmissão – TCP (acrônimo para o inglês *Transmission Control Protocol*) da camada de transporte. O algoritmo de prevenção do congestionamento do TCP se comporta de forma similar ao algoritmo *Additive-Increase Multiplicative-Decrease* (AIMD) proposto por CHIU e JAIN em (1989). O modelo AIMD é um modelo simples para alocar recursos de forma equitativa entre dois usuários que compartilham um único canal. Portanto, este modelo pode ser considerado um paradigma. Por outro lado KELLY et.al. (1998), formulam o modelo de controle de fluxo na forma de uma otimização estática e um problema de estabilização dinâmica, demonstrando que o problema de controle de congestionamento pode ser resolvido como um problema de otimização. Neste contexto o problema de controle de congestionamento pode ser proposto pelo ponto de vista de um sistema de controle.

Este trabalho, cujo objetivo é a implementação de algoritmos de controle para o problema de congestionamento em rede de computadores, é estruturado como segue. No Capítulo 1, alem da introdução, são apresentados os objetivos, as definições e as métricas a serem utilizadas ao longo do trabalho, assim como as abordagens do problema do controle de congestionamento da Internet. No Capítulo 2 são introduzidos os modelos matemáticos existentes que serviram de inspiração para os novos algoritmos apresentados no Capítulo 3. O Capitulo 4 detalha os novos protocolos propostos neste trabalho e o capítulo 5 apresenta os resultados das simulações lineais dos modelos propostos e dos protocolos implementados no simulador NS2 em topologias tipo *dumbbell*. Finalmente, no Capítulo 6, são apresentadas as conclusões e as perspectivas de trabalhos futuros. O apêndice A contém uma breve introdução dos conceitos básicos do simulador NS2. No apêndice B encontram-se algumas tabelas resultantes das simulações que foram úteis para o estudo dos protocolos.

1.1 Objetivos, definições, e critérios de avaliacao.

A Figura 1.1 apresenta uma rede composta por um único canal com capacidade *cap* bps, na qual vários usuários, em seus respectivos *hosts*, estão baixando arquivos de servidores Web, uma situação típica na Internet.

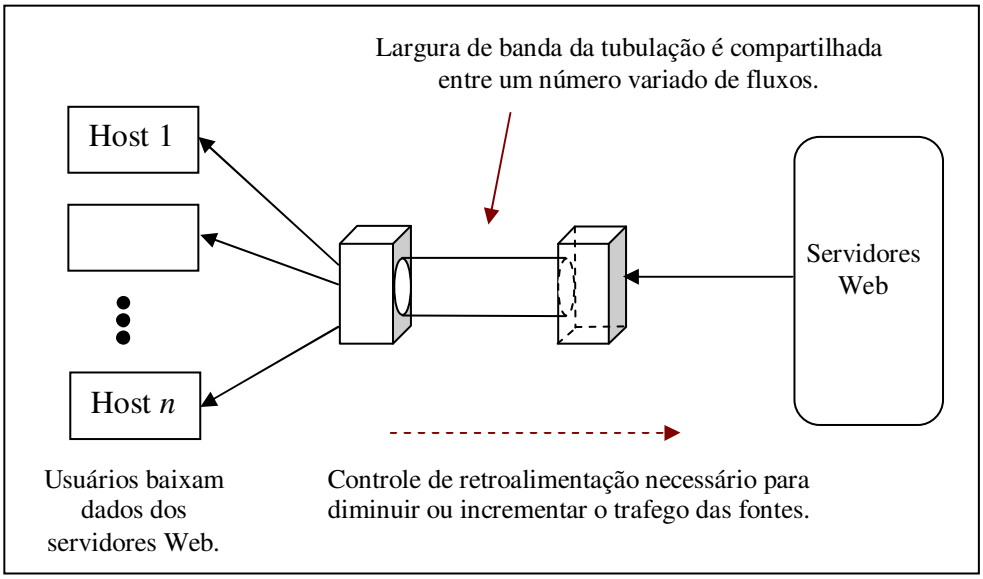


Figura 1.1: Usuários compartilhando um canal para baixar arquivos de servidores Web.

Supondo que, um único usuário inicia um *download* do servidor Web sobre o canal, é razoável esperar que um protocolo de transferência de dados ideal forneça a

transferência deste arquivo com uma vazão ou taxa de transferência efetiva de *cap* bps (KUMAR et.al., 2004). Se esta largura de banda for totalmente disponibilizada para transferência, a sessão estará fora do sistema assim que possível. Supondo que agora outro usuário comece uma sessão, quando o primeiro usuário ainda está transmitindo, quando o correspondente servidor Web dá início à transferência de dados para o usuário, a taxa total de entrada no canal (em direção ao servidor Web) excederá *cap* bps. Se a transferência do primeiro arquivo está sendo processada a *cap* bps, a taxa de serviço do canal será excedida sem considerar o quão lentamente o segundo usuário está emitindo seus dados. Como conseqüência, ocorrerá congestionamento do canal. O dispositivo da rede que interconecta o servidor LAN ao canal *backbone* terá *buffers* neste canal. Estes *buffers* podem absorver o excesso de dados acumulados durante esta sobrecarga, desde que esta situação não persista por muito tempo. Além disso, esta situação é claramente injusta, já que um usuário obtém a taxa total do canal e o outro usuário quase não obtém uma taxa de transferência efetiva.

Estes dois temas, congestionamento e injustiça, para serem estudados, requerem uma espécie de realimentação das fontes de dados de modo que a taxa da primeira transferência seja reduzida e a taxa da segunda seja incrementada de tal forma que cada transferência receba uma taxa de $cap/2$ bps. Supondo agora que a transferência do primeiro arquivo termine e a segunda transferência continue a $cap/2$ bps, a largura de banda do canal será desperdiçada e a segunda sessão será prolongada desnecessariamente. Então, quando a primeira sessão acaba, a fonte da segunda sessão poderia incrementar sua taxa de transferência e obter uma taxa de transferência efetiva *cap* bps.

Em resumo, pode-se concluir que um controle de realimentação explícito ou implícito é oportuno na medida que o número de sessões varia a taxa de transferência fornecida a cada sessão varia conformemente. A realimentação explícita significa que os pacotes de controle fluem entre as fontes de tráfego, dissipadores e a rede, e estes pacotes carregam informações (e.g., uma taxa explícita ou um sinal de redução de taxa) usadas pelas fontes para adaptar suas taxas de envio. Por outro lado, a realimentação implícita pode ser fornecida por pacotes perdidos ou um incremento no atraso da rede, isto é, uma fonte pode reduzir sua taxa quando detecta que um dos pacotes que emitiu não chegou ao destino, por exemplo, TCP usa um protocolo de transmissão baseado em janelas (TANENBAUM, 1996). Uma fonte TCP detecta os pacotes perdidos, toma isto

como indicador de má combinação de taxas e congestionamento da rede e, voluntariamente, reduz sua taxa de transmissão, reduzindo a sua janela de transmissão.

O controle de congestionamento pode ser classificado da seguinte forma: o controle de ciclo aberto, usado em redes de comutação de circuitos (*circuit switched*), e controle de ciclo-fechado, usado principalmente em redes de comutação de pacotes (*packet switched*) e usa informações de retorno (informação de realimentação) global e local. O controle de ciclo fechado é classificado em realimentação explícita e realimentação implícita.

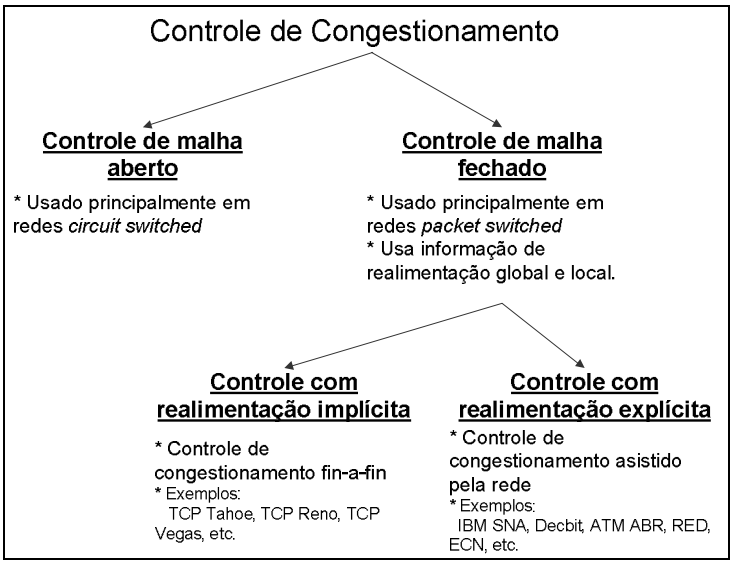


Figura 1.2: Classificação do controle de congestionamento

Como modo de comparação pode-se utilizar um sistema de águas, onde os usuários abrem ou fecham as torneiras para ajustar o fluxo de água contido no sistema. O objetivo é não ultrapassar a capacidade do canal. Nas Figuras 1.3 a 1.9 pode-se observar o processo de envio de pacotes (representado por águas) com realimentação em casos onde existe ou não congestionamento. As Figuras 1.3 a 1.6 mostram que os remetentes TCP aumentam suas taxas de envio respondendo aos sinais de não congestionamento. Na Figura 1.7 as taxas de envio superam a capacidade do canal (a água ultrapassa a capacidade) e o roteador descarta os pacotes que não podem ser tratados por ele. Então, um sinal de congestionamento é enviado à fonte, que reduz a taxa segundo um determinado algoritmo. O sinal de congestionamento pode ser explícito ou inferido por ausência do sinal de realimentação ou ACK correspondente (ou ACK duplicados, como no caso do protocolo TCP).

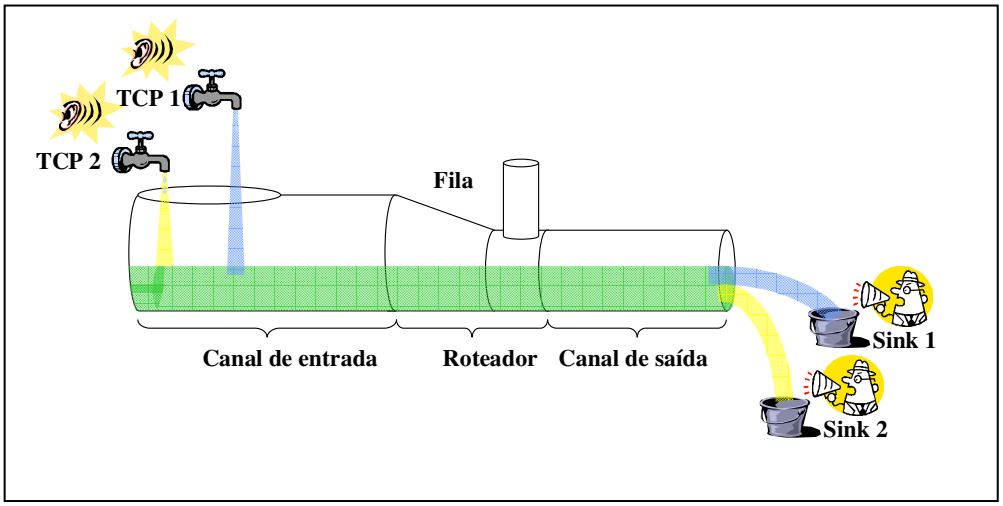


Figura 1.3: Dois remetentes TCP enviam fluxos para dois receptores TCP.

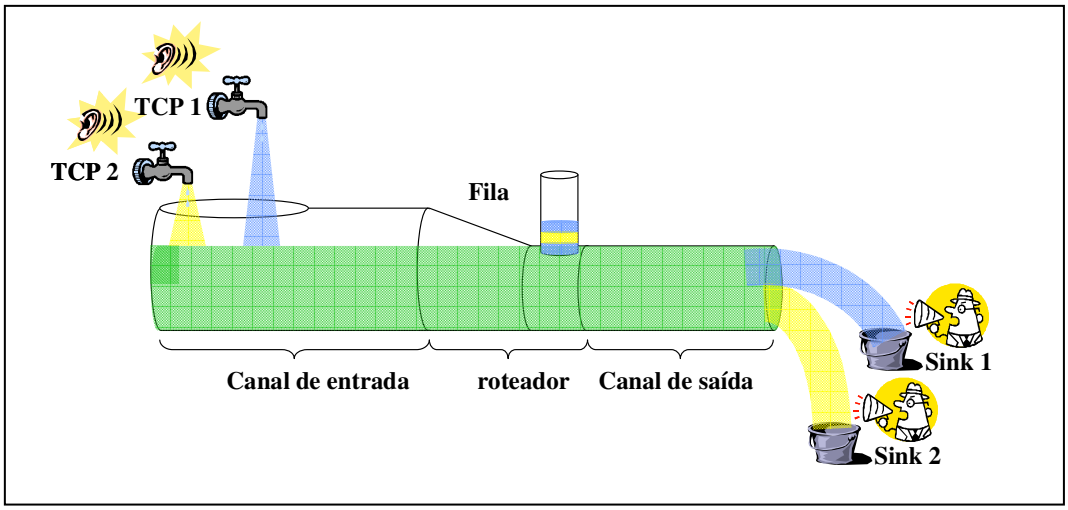


Figura 1.4: Receptores TCP enviam ACKs aos remetentes indicando que não existem perdas.

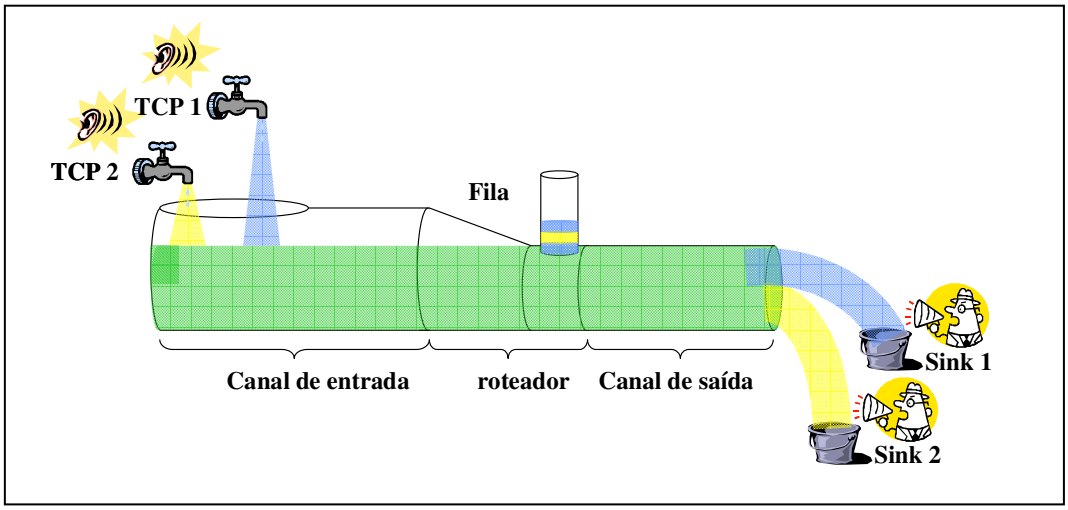


Figura 1.5: Remetentes TCP aumentam seus fluxos de envio

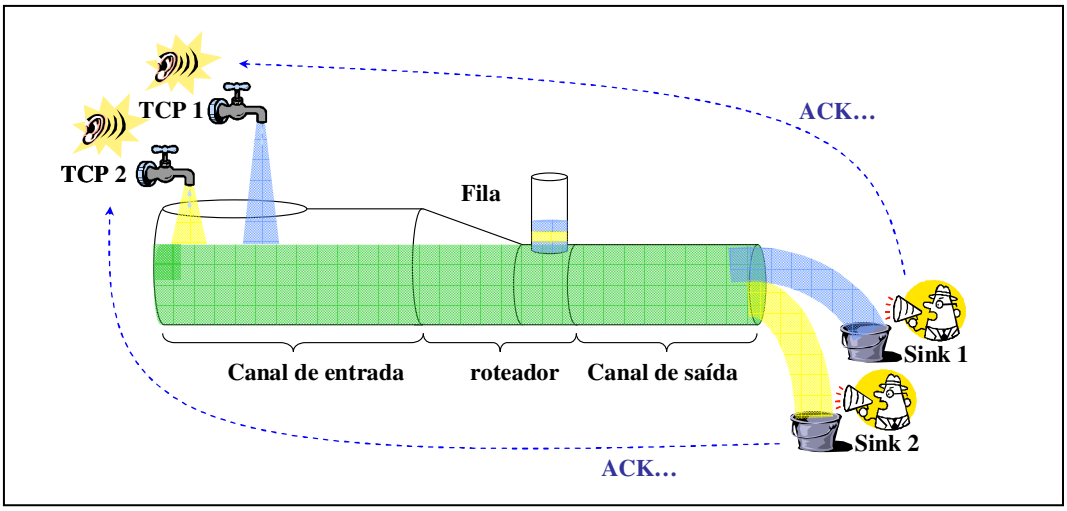


Figura 1.6: Receptores TCP enviam ACKs aos remetentes TCP.

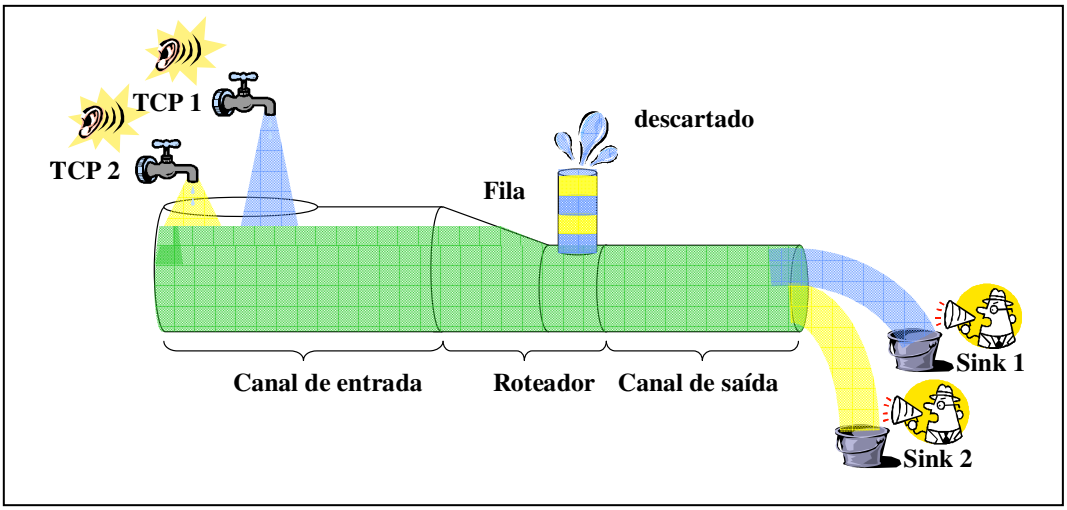


Figura 1.7: Remetentes TCP aumentam seus fluxos, mas existe perda na fila do roteador.

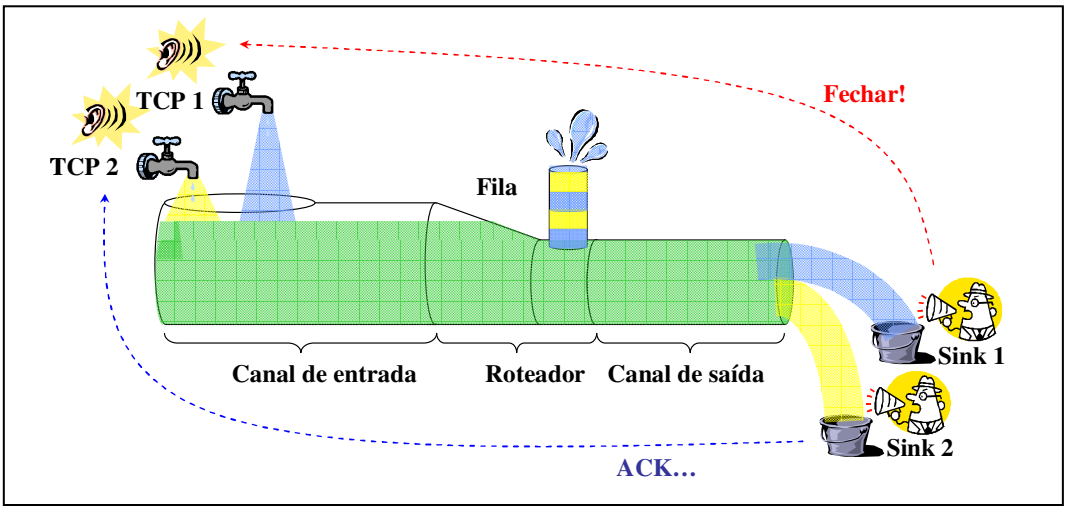


Figura 1.8: O fluxo 1 teve perda e o receptor correspondente avisa ao remetente.

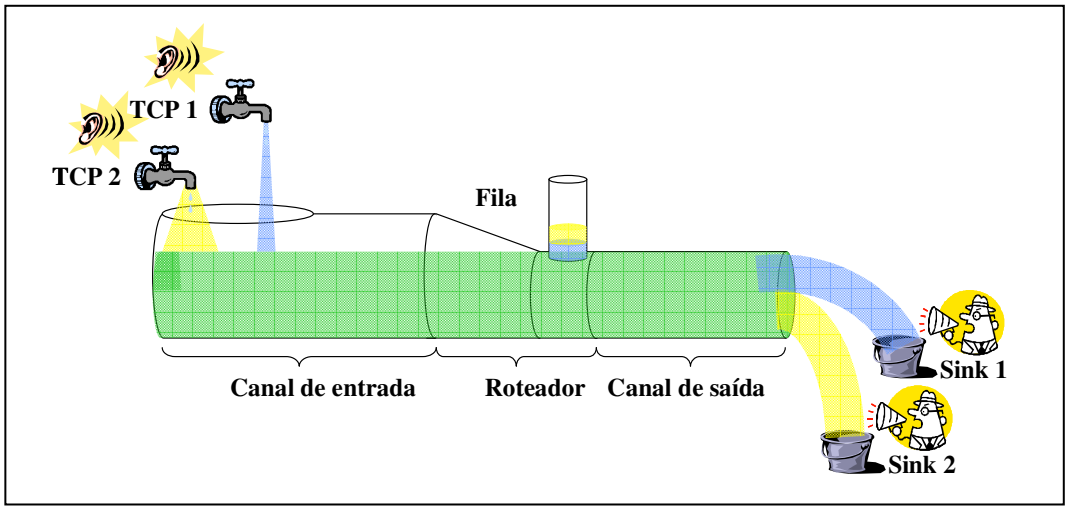


Figura 1.9: O remetente TCP que recebeu o sinal de congestionamento reduz seu fluxo e não existe mais perda.

Pode-se observar através das figuras precedentes (Figuras 1.3 a 1.9) que o roteador não aplica nenhuma política para prever a perda de pacotes. O roteador somente se limita a descartar os pacotes (fluxo) quando a fila fica cheia (isto é conhecido como *Drop-Tail*).

Em outros cenários, o roteador intervém no controle. Por exemplo, no algoritmo *Active Queue Manager* ou AQM (RYU et.al., 2003) o roteador tem um nível na fila e um observador. Se a quantidade de pacotes ultrapassa um determinado limiar, o roteador envia um sinal de congestionamento ao receptor TCP correspondente, que indica ao emissor que reduza a sua taxa para evitar, assim, a perda de pacotes e o congestionamento do canal. A intervenção do roteador no controle de congestionamento tem suas vantagens, tais como, a redução da perda de pacotes, devido à sobrecarga da fila, e a redução de atrasos na mesma. No entanto, este trabalho tem como foco os métodos de controle de congestionamento nos quais somente a camada de transporte interfere e o principal algoritmo proposto se executa nessa camada. Porém, os protocolos apresentados obtêm ajuda do roteador.

A modelagem do controle de congestionamento da Internet é essencial. Ao contrário dos intensivos esforços de desenho, a modelagem tem sido substancialmente menos abordada (GORINSKY, 2004). A convergência aos modelos realísticos bem compreendidos é uma chave para superar a "ossificação" da Internet. Os modelos impróprios existentes não estão surpreendendo devido à complexidade de modelagem do controle do congestionamento. Em detalhe, parece impossível construir um único

modelo que represente exatamente todos os aspectos do controle do congestionamento da Internet e que seja simples o bastante para ser útil (GORINSKY, 2004).

A Figura 1.10 mostra o modelo suposto de uma rede com *n* usuários compartilhando a mesma rede (CHIU e JAIN, 1989). Neste modelo, x_i denota a carga ou taxa gerada pelo usuário *i*. O modelo supõe que todos os usuários têm o mesmo Tempo Total de Percurso (ou RTT pelo acrônimo do inglês *Round Trip Time*) ajustando suas cargas simultaneamente, isto é, um modelo síncrono sem atrasos. Assim, o modelo utiliza uma escala de tempo discreto na qual cada instante *t* corresponde ao momento que cada usuário *i* ajusta a sua carga $x_i(t)$. O estado do congestionamento é determinado pelo número de pacotes no sistema ($\sum x_i(t)$). A rede envia aos usuários um sinal binário $I(t)$ (realimentação binária) que indica se a carga total ultrapassa ou não um valor ótimo de carga ou capacidade da rede cap. Se a rede envia $I(t) = 1$ aos usuários quando está congestionada, isto indica que os mesmos devem diminuir suas taxas. Caso contrário, se envia $I(t) = 0$, a rede indica que não existe congestionamento. Neste caso, os usuários incrementam suas taxas de envio.

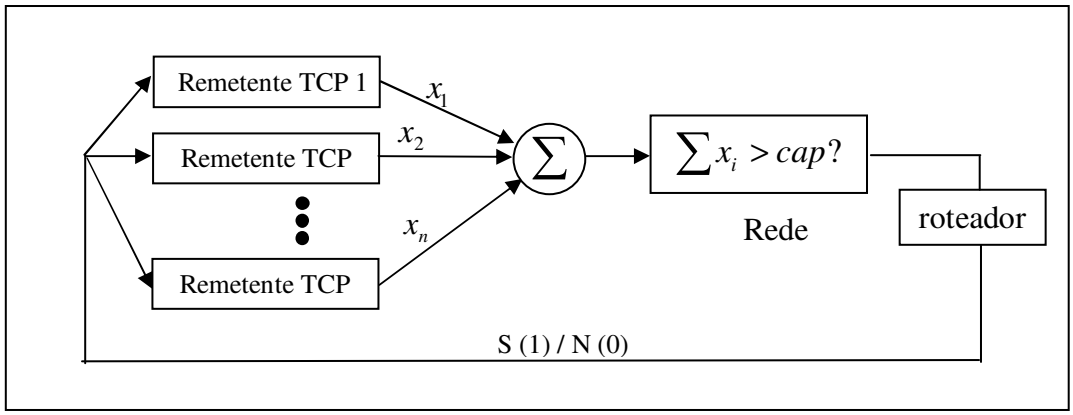


Figura 1.10: Modelo de controle síncrono do sistema de n usuários compartilhando uma rede.

A Figura 1.11 mostra a correspondência do modelo com o diagrama de água apresentado anteriormente, no qual os remetentes TCP variam as taxas dependendo da resposta da rede. O roteador descarta o fluxo que não cabe na rede, isto é, quando a somatória das taxas excede a capacidade do canal.

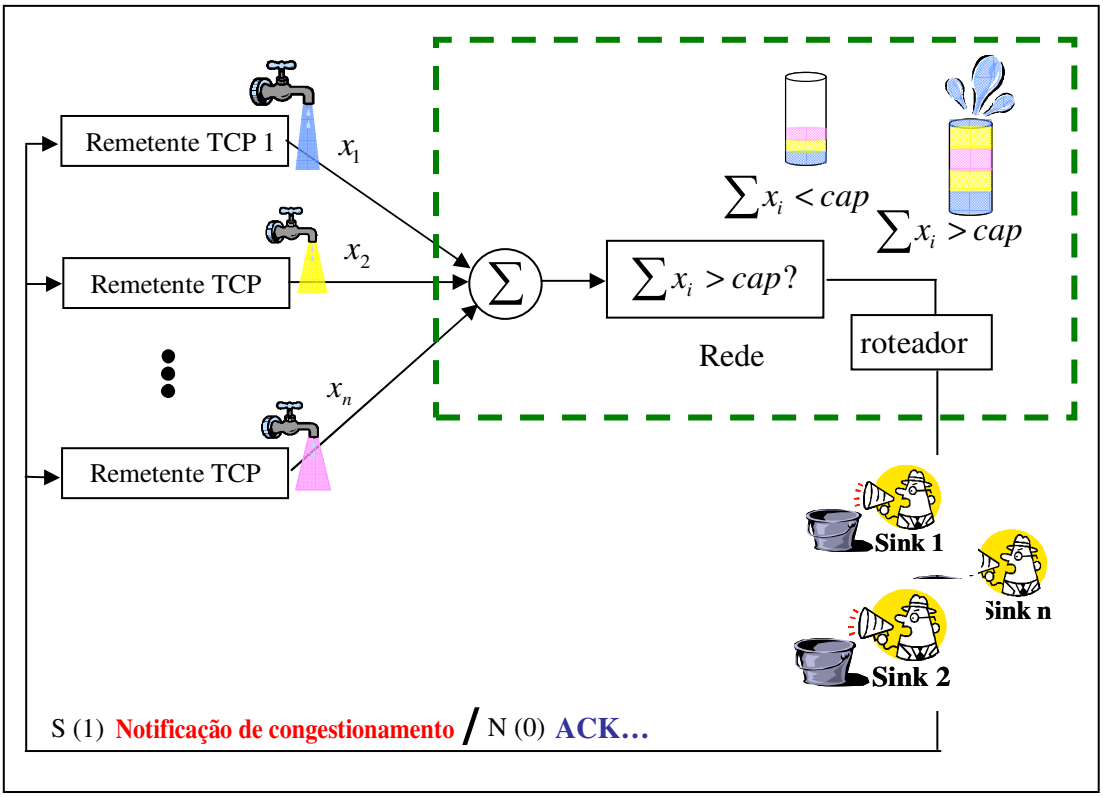


Figura 1.11: Correspondência do modelo de controle síncrono com a representação anterior.

Para o estudo dos algoritmos serão necessários critérios para avaliar o desempenho dos mesmos. Algumas dos critérios usados para avaliar uma técnica que evite o congestionamento são: eficiência, equidade, distributividade e convergência (CHIU e JAIN, 1989, LAHAMAS , 2003, LAHAMAS e TSAUSSIRIS, 2003, MAMATAS et.al, 1999, POSTEL, 1981).

Eficiência:

Eficiência do uso de um recurso é definida como a aproximação da carga total ao valor desejado cap, logo o recurso está operando eficientemente quando a alocação $X(t) = \sum x_i(t)$ está próxima ao cap. Sobrecarga, ($X(t) > cap$), ou subcarga, ($X(t) < cap$), são ambas indesejáveis e consideradas ineficientes. Nota-se que a eficiência se relaciona somente com as alocações totais e, assim, duas alocações diferentes podem ambas ser eficientes sempre e quando a alocação total estiver perto do objetivo. A distribuição da alocação total entre os usuários individuais é medida pelo critério de equidade.

Equidade:

O critério de equidade foi estudado extensamente na literatura (CHIU e JAIN, 1989, GORINSKY e VIN, 2002). Quando múltiplos usuários compartilham múltiplos recursos o critério de equidade *max-min* tem sido bastante adotado (CHIU e JAIN, 1989). Essencialmente, o conjunto de usuários é dividido em grupos de classes equivalentes sujeitos a qual recurso é seu primeiro gargalo. O critério *max-min* afirma que usuários na mesma classe equivalente devem ter partes iguais na largura de banda. Assim, o sistema no qual $x_i(t) = x_j(t) \ \ \forall i, j$ compartilha o mesmo gargalo está operando em forma justa. Se os usuários não recebem exatamente as mesmas alocações, o sistema é menos justo. Por isso, faz-se necessário um índice ou função que qualifique a equidade. Uma métrica bem conhecida é:

$$\text{Índice de equidade} = F(x) = \frac{\left(\sum x_i\right)^2}{n\left(\sum x_i^2\right)} \tag{1.1}$$

O índice de equidade tem as seguintes propriedades:

a) Este índice está limitado entre 0 e 1 (ou 0% e 100%). Uma alocação totalmente equitativa (com todos os $x_i's$ iguais) tem o índice de equidade 1 e uma alocação totalmente injusta (com todos os recursos outorgados a um único usuário) tem um índice de equidade de $1/n$ que é 0 no limite quando n tende para ∞. Como este índice está limitado entre 0 e 1, a equidade nesse contexto caracteriza a distribuição justa dos recursos entre os fluxos em um canal de gargalo compartilhado.
b) O índice de equidade é independente à escala, i.e., a unidade de medição não interessa.
c) O índice de equidade é uma função continua. Qualquer cambio ligeiro na alocação manifesta se no índice de equidade.
d) Se somente um número k de n usuários compartilham o recurso igualmente com os restantes $n-k$ usuários que não recebem nenhum recurso, logo o índice é k/n.

Distributividade:

Outro critério que deve ser considerada em um esquema de controle é que este seja distribuído. Um esquema centralizado requer completo conhecimento do estado do sistema. Por exemplo, quando se deseja conhecer a demanda individual de cada usuário,

esta informação pode estar disponível no recurso. No entanto, disponibilizar esta informação para cada usuário y cada vez causa uma considerável sobrecarga, especialmente se um usuário estiver usando vários recursos ao mesmo tempo. O controle deve ser de tal forma que possa ser implementado em redes reais, o que se deve assumir que o sistema deve retornar uma realimentação mínima. Devendo-se indicar somente si ele está sobrecarregado ou subcargado mediante bits binários de realimentação.

Convergência:

Finalmente, deseja-se que o esquema de controle seja convergente. A convergência é, geralmente, medida pela velocidade (ou tempo) que o sistema se aproxima do estado objetivo a partir de qualquer estado inicial. No entanto, devido à natureza binária da realimentação, o sistema não converge, em geral, a um único estado estável. Pelo contrário, o sistema alcança um equilíbrio no qual ele oscila em torno de um estado ótimo. O tempo que leva para alcançar este equilíbrio e a medida das oscilações determinam a convergência. O tempo determina o grau de resposta e o tamanho das oscilações determina a suavidade do controle (Figura 1.12). De fato, se deseja-se que as oscilações sejam pequenas. Com isso, controles com pequenos tempos e pequenas amplitudes de oscilações são chamados de mais rápida resposta e mais suaves.

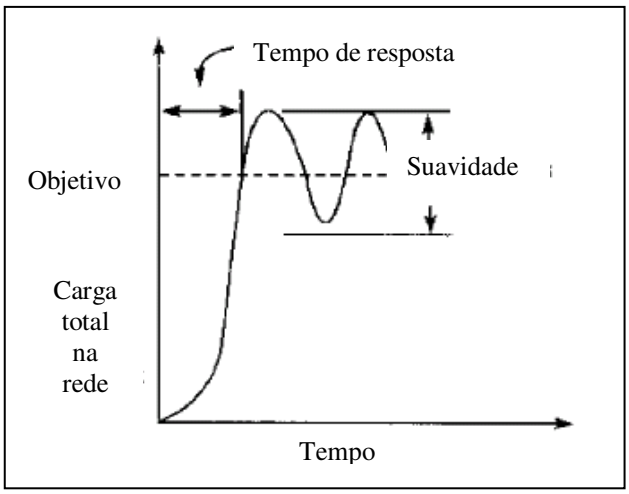


Figura 1.12: Resposta e suavidade

Qualquer que seja o algoritmo de congestionamento, este deve cumprir certos objetivos (MITRA e SEERY, 1993):

- Alcançar uma utilização elevada da largura de banda.
- Convergir mais rapidamente à equidade.

- Minimizar o comprimento das oscilações.
- Manter o tempo de resposta rápido.
- Coexistir razoavelmente e ser compatível com protocolos (baseado em *AIMD*) tradicionais extensamente usados.

Tendo em vista os objetivos mencionados, são apresentadas diversas técnicas que podem ser classificada seguindo diferentes abordagens, como será apresentado mais adiante.

1.2 Abordagens do problema de controle do congestionamento em redes de comunicação.

Como todo desenho de controle de congestionamento, o TCP consiste em dois componentes: retroalimentação, que notifica à entidade transmissora o estado do congestionamento na rede; e o algoritmo de ajuste, que regula a transmissão em resposta à retroalimentação e está baseado no algoritmo AIMD.

O paradigma prevalecente usado para o desenho e operação do Internet é "torná-lo simples". O principio do desenho do TCP é "não pergunte à rede o que você pode fazer por você mesmo" (WALRAND, 1998). Não obstante, na medida em que o número de usuários e o tamanho da Internet aumentam, mais pacotes são perdidos e acontecem outras degradações de performance (RYU et.al., 2003). Além disso, a Internet está evoluindo de um único serviço do melhor-esforço para uma variedade de serviços, incluindo serviços de tempo real. Consequentemente, muitos algoritmos foram propostos recentemente para controlar o congestionamento eficientemente e para lidar com a evolução dos serviços da Internet. Têm-se assim abordagens de arquitetura e abordagens econômicas.

1.2.1 Abordagens baseadas em arquitetura

As abordagens de arquitetura se referem às abordagens que modificam os algoritmos da fonte e/ou da rede para fornecer um melhor controle do congestionamento ao manter o paradigma do Internet atual e os princípios do projeto do TCP. A modificação do algoritmo da fonte usa somente a informação implícita do congestionamento na fonte, sem nenhum auxílio da rede. Por outro lado, a modificação

do algoritmo da rede usa a informação explícita do congestionamento e entrega a melhor indicação do congestionamento à fonte (realimentação explícita). Cada uma destas modificações pode ser executada independentemente. Entretanto, para a cooperação entre algoritmos da fonte e os componentes da rede, é muito importante conseguir a alocação de recurso eficiente da rede, assim como o controle eficaz do congestionamento.

No caso dos algoritmos de fonte, um remetente TCP aumenta o fluxo da carga após a recepção de uma confirmação *ACK* (acrônimo do inglês *acknowledgment*). Porém, o transmissor decrementa o fluxo da carga depois de um *timeout* ou depois de receber um dado número de *ACK* duplicados (JACOBSON, 1988). Até a recepção do primeiro indicador de congestionamento, cada conexão TCP incrementa a sua carga de maneira semelhante ao Algoritmo de *Multiplicative Increase* (CHIU e JAIN, 1989). A utilização deste mecanismo supõe uma convergência rápida aos estados eficientes. Uma vez que a eficiência é atingida, a conexão TCP muda para o modo Prevenção do Congestionamento e ajusta a carga de forma similar ao AIMD (CHIU e JAIN, 1989). Supõe-se que o uso do *AIMD* fornece estabilidade, isto é, converge a estados eficientes e balanceados.

Os protocolos de camada de transporte mais utilizados hoje em dia são o TCP com as variações TCP Tahoe (JACOBSON, 1988, POSTEL, 1981, RYU et.al., 2003, TCP Reno (RYU et.al., 2003), TCP Vegas (BRAKMO et.al., 1994), TCP SACK (MATHIS et.al., 1996, 2008) e TCP New Reno (FLOYD e HENDERSON, 1999, RYU et.al., 2003) .

Para melhorar o controle de forma mais efetiva, mecanismos de controle de congestionamento para gerenciamento de filas FIFO-baseadas, gerenciamento de filas por-fluxo e mecanismo de planejamento são requeridos nos roteadores (RYU et.al., 2003). A técnica tradicional de gerenciamento (*FIFO*-baseada) no roteador é o *Drop Tail*, a qual ajusta um comprimento máximo da fila nos termos do número dos pacotes para cada fila.

Esta técnica tem dois inconvenientes: bloqueio, que acontece quando o *Drop Tail* permite que certas conexões monopolizem o espaço da fila, e filas cheias que podem persistir por muito tempo até *Drop Tail*. Uma possível solução é descartar pacotes antes que a fila fique cheia de forma tal que as fontes possam responder ao congestionamento antes que os *buffers* se excedam (aconteça a sobrecarga). Esta técnica é conhecida como *Active Queue Management* (*AQM*) (RYU, 2003) e o *Random Early Detection* (*RED*)

(FLOYD e JACOBSON, 1993, RYU et.al., 2003, LIN e MORRIS) é um exemplo desta técnica. O *DECbit* (RAMAKRISHNAN e JAIN, 1988) e o *Explicit Congestion Notification* (*ECN*) (RAMAKRISHNAN e FLOYD, 1999) também permitem que um roteador congestionado notifique a fonte do congestionamento sem necessidade de descartar pacotes. O *Available Bit Rate* (*ABR*) (KALYANARAMAN et.al., 2000) e o *eXplicit Control Protocol* (*XCP*) (KATABI et.al., 2002, PACHECO et.al., 2008), que substituem a realimentação binária por informe de múltiplos bits são outros mecanismos que sofrem intervenção do roteador.

1.2.2 Abordagens baseadas em princípios econômicos

As abordagens econômicas foram propostas introduzindo conceitos da economia da rede usando modelagem e análise matemática. Estas estudam o problema de controle de congestionamento como um problema de alocação de recursos (a rede). As abordagens baseadas em preço usam o preço (por pacote ou por largura de banda) como um dispositivo para o controle do congestionamento e para diferenciação do serviço. As abordagens baseadas em otimização usam a programação matemática ou técnicas de teoria de jogos para analisar e otimizar o comportamento da rede ou dos usuários.

a) Abordagens Baseadas em Preços

Os economistas desenvolveram maneiras de modelar o problema dos indivíduos que competem por recursos limitados (por exemplo, largura de banda) por preços de tratamento como um mecanismo para dirigir o consumo (VARIAN, 2007). A diferença da teoria econômica padrão é que a infra-estrutura tecnológica de Internet pode permitir uma descoberta na área de contabilidade distribuída em linha. A descoberta é a capacidade de cobrar dos usuários de maneira que reflita precisamente suas ações usando somente um mecanismo fixando o preço simples em cada pacote (RYU et.al., 2003).

MACKIE-MASON e VARIAN (1995) descrevem uma abordagem inteligente do mercado para alocar recursos em uma rede, onde um preço (ou uma oferta) é ajustado para cada pacote dependendo do nível da demanda para a largura de banda. A rede admite pacotes com preços de oferta que excedem a quantidade atual da interrupção. Esta interrupção é determinada pelos custos marginais do congestionamento imposto

pelo pacote adicional seguinte. Os pacotes rejeitados são retornados aos usuários para tentar novamente o envio ou podem ser distribuídos por um canal mais lento.

SHENKER (1995), SHENKER et. al. (1996) são contra fixar o preço das estruturas baseadas unicamente no custo do congestionamento marginal. Os autores argumentam que os custos marginais podem não ser suficientes para recuperar o rendimento e que os custos do congestionamento são difíceis de calcular. Esta proposta está mais focalizada em edições estruturais e arquiteturais. Neste contexto, é proposto um esquema de preço de fronteira, que descreve o lugar em que as cargas são avaliadas. Estas cargas podem ser por pacote ou por byte, para comprar uma determinada quantidade de capacidade da rede.

ODLYZKO (1999) sugeriu o esquema *Paris Metro Pricing (PMP)* para fornecer serviços diferenciados na Internet. Nesta abordagem, a rede é dividida em diversos canais logicamente separados, diferenciando–se somente pelos preços pagos por seu uso. A idéia é apresentar menos tráfego nos canais com preço mais elevado e melhor qualidade de serviço fornecida.

b) Abordagens Baseadas em Otimização

Métodos baseados em otimização empregam a programação matemática ou teoria de jogos para resolver o problema do controle de congestionamento. Estes métodos podem formular o controle de congestionamento como um modelo analítico e, assim, obter estados estacionários (RYU et.al., 2003). Estes métodos podem ser usados também para analisar a dinâmica da rede em um ambiente de controle descentralizado. Podem-se encontrar igualmente guias para a modificação (ou melhoria) de métodos atuais de controle de congestionamento, assim como, no desenho e operação de algoritmos de controle.

O problema de alocação de recursos de rede (e.g., de controle de congestionamento) tem duas componentes: os usuários e a rede. Os usuários podem ser diferenciados por seus tráfegos característicos, representados por uma função de utilidade (STOICA et.al., 1998). Um usuário deseja maximizar seu beneficio, utilidade menos o custo de largura de banda, durante o controle de congestionamento. A rede consiste em um conjunto de canais com restrições de capacidade. Os recursos da rede são compartilhados por um conjunto de usuários. Com isso, o problema da alocação de recursos da rede se torna um problema de programação não-linear (NLP) com constantes lineais (BERTSEKAS e GALLAGER, 1992, KELLY et.al., 1998, MITRA e

SEERY, 1993). O objetivo é escolher taxas de envio das fontes de forma a maximizar a utilidade agregada sujeita à restrição de que a taxa total da fonte em qualquer canal não exceda a capacidade do canal. Através desta formulação, pode-se alcançar a *Eficiência Pareto*[1] ou um conjunto ótimo global de taxas para usuários.

Existem vários trabalhos que focam este método: GOLESTANI e BHATTACHARYYA (1998) formularam o controle de congestionamento fim-a-fim como um problema de otimização global e propõem uma classe de algoritmos *minimum cost flow control* (*MCFC*) para ajustar taxas de envio aos tamanhos de janelas. Kelly et. al. (1998, 2003) propuseram um algoritmo de controle de congestionamento que divide o problema em um subproblema de usuário e um subproblema da rede. FAN et.al. (2005) estenderam o modelo de Kelly para usuários não cooperativos e desenvolveram um algoritmo baseado em nós de fronteira de ajuste de preços, usando ferramentas da teoria de controle, que é aplicável aos diversos tipos de redes. LOW e RADHA (1999) propõem um procedimento diferente do problema de otimização, em que os canais da rede e as fontes são consideradas como processadores de um sistema distribuído da computação, e resolve o problema usando o método da projeção do gradiente.

[1] Uma alocação dos bens e dos preços chama-se *Pareto eficiente* se não houver nenhuma carga que beneficie, simultaneamente, alguém e prejudique a outra, medido por suas funções de utilidade (Ryu et.al., 2003)

2. Modelos matemáticos

A modelagem do problema do controle de congestionamento da Internet ao contrario dos intensivos esforços de desenho, tem sido substancialmente menos tratada (GORINSKY, 2004). Parece impossível construir um modelo único que represente todos os aspectos do controle do congestionamento da Internet e que seja simples ou bastante para ser útil (GORINSK, 2004)

Este capítulo apresenta três enfoques: a modelagem tradicional AIMD, modelos baseados em otimização e modelos inspirados em aspectos biológicos.

2.1 Modelo AIMD de Chiu e Jain

O algoritmo prevalecente de controle de congestionamento da Internet é o AIMD. É um algoritmo executado na camada de transporte de cada nó terminal, com assistência mínima da camada de rede. O AIMD foi introduzido por JACOBSON (1988) e logo em seguida estudado por CHIU e JAIN (1989), resultando em um dos trabalhos mais conhecidos dos últimos anos.

A justificação teórica para favorecer ao AIMD aparece no artigo de CHIU e JAIN. (1989). Nesse trabalho é descrito o primeiro modelo matemático que reconhece a relação entre o controle de congestionamento e a alocação de recursos, também estudada por SRIKANT (2003). O objetivo de todo usuário que envia dados é operar independentemente, ajustando sua taxa de envio de dados (ou janela) de forma que a largura de banda total da rede seja utilizada eficientemente e efetivamente. Segundo a análise de algoritmos lineares de ajuste (CHIU e JAIN, 1989), o AIMD fornece uma convergência mais rápida ao estado de equilíbrio. Porém, trabalhos mais recentes (GORINKSKY e VIN., 2002, LAHAMAS e TSAUSSIRIS, 2003, MITRA e SEERY, 1993, YANG et.al., 2000) demonstram que o AIMD nem sempre garante a convergência mais rápida ao estado equitativo ou nem sempre garanta a convergência.

Os mecanismos de prevenção e de controle de congestionamento são problemas de gerenciamento de recursos dinâmicos que podem ser formulados como problemas do controle de sistema nos quais o sistema detecta seu estado e realimenta esta informação a seus usuários que ajustam seus controles (CHIU e JAIN, 1989).

2.1.1 Modelo de Chiu e Jain para atrasos homogêneos

A Figura 2.4 apresenta o modelo usado por CHIU e JAIN (1989, GORINKSKY et.al., 2003) para a análise de um sistema de controle de congestionamento. Neste modelo, x_i denota a carga ou taxa gerada pelo usuário *i*. No modelo se supõe que todos os usuários têm o mesmo Tempo Total de Percurso (RTT), ajustando suas cargas simultaneamente, isto é, um modelo síncrono no qual, não existem atrasos ou os atrasos são homogêneos. Assim, o modelo utiliza uma escala de tempo discreto, onde cada instante *t* corresponde ao momento em que cada usuário *i* ajusta a sua carga $x_i(t)$.

$$\forall i \quad x_i(t) = \begin{cases} a_I + b_I x_i(t-1) & \text{si} \quad y(t) = 0, \quad \text{(a)} \\ a_D + b_D x_i(t-1) & \text{si} \quad y(\text{t}) = 1 \quad \text{(b)} \end{cases} \tag{2.1}$$

Onde a_I, b_I, a_D e b_D [2] são constantes reais.

O sinal binário $y(t)$ que indica se a carga total $X(t-1)$, após o ajuste prévio, ultrapassa ou não um valor ótimo de carga ou capacidade do canal $cap = c$ para que os usuários incrementem ou diminuam suas taxas de envio.

$$y(t) = \begin{cases} 1 & \text{si} \quad X(t-1) > c \\ 0 & \text{si} \quad X(t-1) \le c \end{cases} \tag{2.2}$$

onde $X(t) = \sum_{j=1}^{n} x_j(t)$ é a carga combinada de todos os *n* usuários no tempo *t*.

O algoritmo pertence à família de algoritmos de ajuste binários que usam funções lineares para o incremento e o decremento. A família inclui:

- Algoritmos *Multiplicative-Increase Multiplicative-Decrease (MIMD)* com
  $a_I = 0$, $b_I > 1$, $a_D = 0$, e $0 < b_D < 1$
- Algoritmos *Additive-Increase Additive-Decrease (AIAD)* com
  $a_I > 0$, $b_I = 1$, $a_D < 0$, e $b_D = 1$,
- Algoritmos *Additive-Increase Multiplicative-Decrease (AIMD)* com
  $a_I > 0$, $b_I = 1$, $a_D = 0$, e $0 \le b_D < 1$, (2.3)
- Algoritmos Multiplicative-*Additive-Increase Multiplicative-Decrease (MAIMD)* com
  $a_I > 0$, $b_I > 1$, $a_D = 0$, e $0 \le b_D < 1$.

[2] (Utilizando-se a letra I para Incremento e D para decremento)

Destes algoritmos o mais utilizado no protocolo TCP e variações é o AIMD. Reescrevendo a equação (2.2) em términos de $h\,\mathrm{sgn}$ e $uh\,\mathrm{sgn}$, temos:

$$x_i(t) = (a_I + b_I x_i(t-1)) \left(-h\,\mathrm{sgn}\left(\sum_{j=1}^{n} x_j(t-1) - c \right) \right) + (a_D + b_D x_i(t-1)) \left(uh\,\mathrm{sgn}\left(\sum_{j=1}^{n} x_j(t-1) - c \right) \right) \tag{2.4}$$

Denotamos o modelo da equação (2.4) como **S_AIMD_CJ**[3] (com $a_I > 0$, $b_I = 1$, $a_D = 0$, e $0 \le b_D < 1$).

O critério para a seleção de um algoritmo apropriado inclui sua estabilidade. Para qualquer carga inicial dos usuários, a carga $x_i(t)$ de cada usuário *i* deve convergir para uma quantidade balanceada c/n. Para quantificar esse balanceamento justo, da carga Chiu e Jain usam o índice de equidade $F(x)$ (1.1). Este índice de equidade é um tipo de função de Lyapunov (CHEN et.al., 2007). Esta função de Lyapunov é sempre não crescente ao largo de toda a sequencia de *switching* demonstrando assim a convergência do método. O método de Chiu e Jain está baseado no uso desta função como função de Lyapunov para garantir assim a convergência do mesmo.

Segundo o analise de Chiu e Jain (GORINSKY e VIN, 2002), o algoritmo linear para ajuste binário deve pertencer à classe *LIMD* (*Linear-Increase Multiplicative-Decrease)* para convergir a estados eficientes equânimes:

Proposição 1: Para satisfazer os requerimentos de convergência distribuída à eficiência e equidade sem truncamento, a política de decremento deve ser multiplicativa e a política de incremento deve ter uma componente aditiva e pode ter uma componente multiplicativa: $a_I > 0$, $b_I \ge 1$, $a_D = 0$, e $0 \le b_D < 1$.

Depois convergir à eficiência, a carga total sob LIMD oscila dentro dos limites estritos: $b_D\, c < X(t) \le n\, a_I + b_I\, c$.

A Proposição 1 tem uma explicação intuitiva (GORINSKY e VIN, 2002): O ajuste multiplicativo não afeta o índice de balanceamento, porém os incrementos aditivos o melhoram. Portanto, os algoritmos de *Linear-Increase Multiplicative-Decrease* ou *LIMD* (com $b_I \ge 1$) conduzem monotonamente ao índice de balanceamento para o seu valor ótimo.

---

[3] S: Síncrono

GORINSKY e VIN. (2004) afirmam logo, que para convergir à equidade, um algoritmo de ajuste linear deve ser *MAIMD* ou *AIMD*.

CHIU e JAIN (1989) mostram que existe um conflito entre a suavidade e o tempo de resposta: cada intento de trocar a_I, b_I, ou b_D para melhorar a suavidade piora o tempo de resposta e vice versa. Considerando esta negociação entre tempo de resposta e suavidade, CHIU e JAIN (1989) examinam a velocidade de convergência a estados equânimes. Depois de provar que um único aumento sob LIMD impulsiona o índice de equidade mais quando $b_I = 1$, Assim, CHIU e JAIN (1989) concluem:

Proposição 2: Para obter a mais rápida convergência ao estado equitativo, o algoritmo de ajuste linear deve ser com $a_I > 0$, $b_I \geq 1$, $a_D = 0$, e $0 \leq b_D < 1$.

Contradizendo a proposição 2, GORINSKY e VIN (2002, 2004) fazem uma análise estendida do algoritmo e provam que *AIMD* nem sempre garante a mais rápida convergência aos estados justos (ver Figura 2.1) . Mostram que um algoritmo *LIMD* com um componente de incremento multiplicativo pode fornecer uma convergência mais rápida à equidade. Demonstram, ao mesmo tempo, que uma classe mais vasta de algoritmos (que incluam aqueles com componentes de incremento aditivo com uma política de decremento diferente ao multiplicativo) podem garantir convergência.

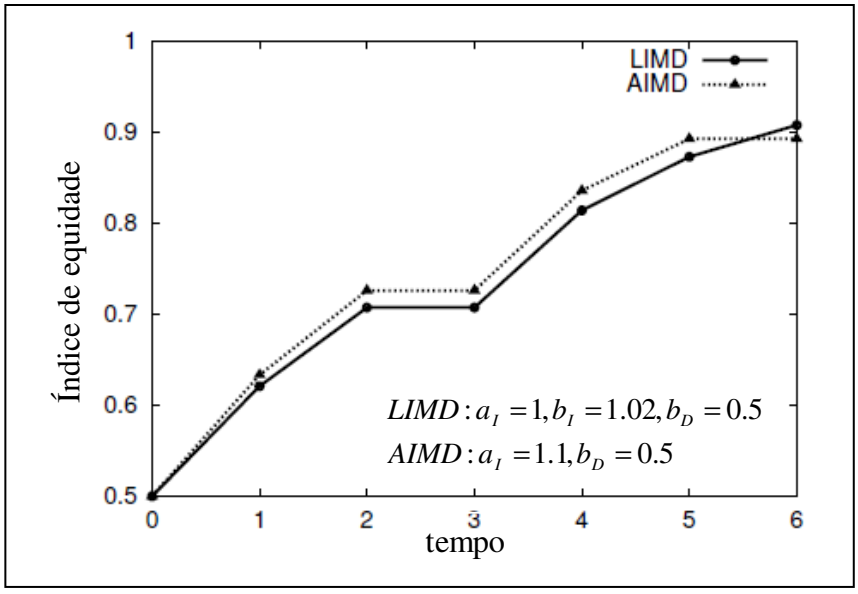


Figura 2.1: Velocidade de convergencia a estados equitativos sob LIMD em modelo de Chui e Jain para, $n = 2$, $c = 10$, $x_1(0) = 7$ e $x_2(0) = 0$.

GORINSKY e VIN (2004) mostram também que a seleção do modelo tem um impacto direto nos resultados. De acordo com os autores, o *Multiplicative-Increase*

Multiplicative-Decrease ou *MIMD* ($a_I = 0$ e $a_D = 0$), não é estável no modelo de Chiu e Jain e converge a estados equitativos baixo suposições mais realistas de realimentação negativa. Um usuário com maior taxa recebe indicadores de congestionamento mais frequentemente. Assim, realimentação negativa reflete equidade e, por conseguinte facilita a convergência aos estados justos.

YANG et al.(2000), também, demonstraram que *MAIMD* ($b_I > 1$) pode ter maior velocidade de convergência em forma global. Seus resultados experimentais evidenciam que as velocidades de convergência do *AIMD* e *MAIMD* são muito próximas umas de outras. Ainda afirmam que *MAIMD* pode incrementar sua carga com velocidade exponencial e, por conseguinte, ser muito mais rápido que *AIMD*. Além disso, afirmam que escolher *AIMD* ao invés de *LIMD* baseado na velocidade de convergência à equidade pode não ser o melhor critério de decisão.

2.1.2 Modelo de Chiu e Jain para atrasos heterogêneos

Embora o modelo síncrono seja adotado extensamente, está associado a um número de suposições e/ou de simplificações, que realmente podem não acontecer em Internet. O diagrama do sistema é mostrado na Figura 2.1 e apresenta o algoritmo de forma mais realista, permitindo a diferentes usuários ter diferentes RTTs. A Figura 2.2, também, representa o modelo assíncrono estudado por SRIKANT (2003) e por GORINSKY e VIN (2002, 2004), onde d_i^f indica o atraso de x_i da fonte i ao canal e d_i^b o atraso do sinal de realimentação.

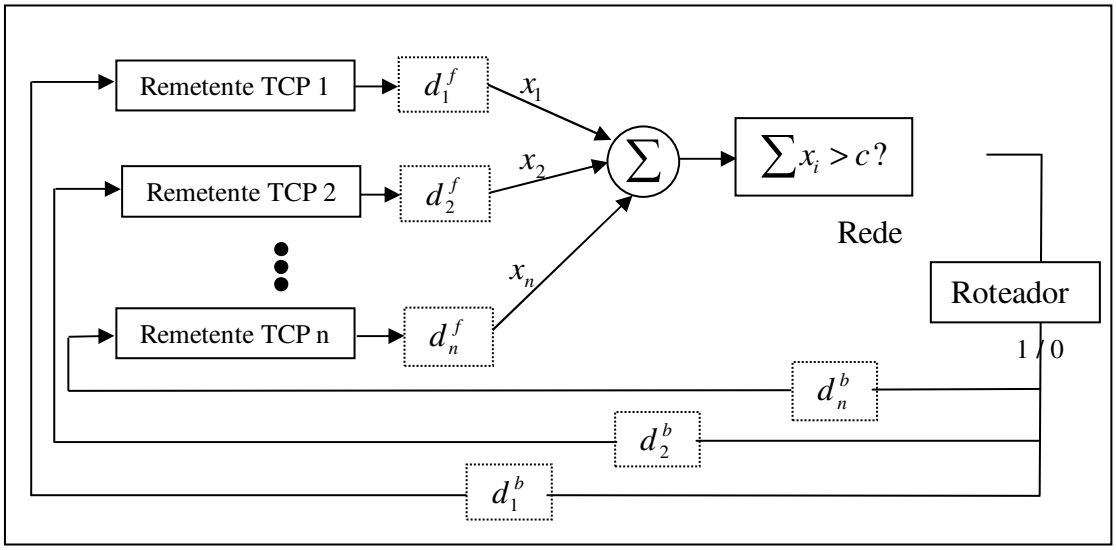


Figura 2.2: Modelo de controle assíncrono do sistema de n usuários compartilhando uma rede.

GORINSKY e VIN em (2002) estendem o modelo de Chiu e Jain a um modelo que permite a diferentes usuários ter diferentes tempos totais de percurso (RTT). Mostram que segundo este modelo, *AIMD* não converge aos estados equitativos, que é sensível a condições inicias e tem múltiplos atratores. Embora, o caos no comportamento do TCP tinha sido mais previamente observado, as conclusões de Gorinsky e Vin indicaram que a natureza caótica do controle do congestionamento do TCP pode ser atribuída a *AIMD*. Outros trabalhos também verificam a afirmação sobre o comportamento caótico do TCP a diferentes atratores (YANG, 2002).

Levando em conta os atrasos d_i^f e d_i^b, se redefine a carga ou taxa total da rede no tempo t como:

$$X(t) = \sum_{i=1}^{n} x_i(t - d_i^f) \tag{2.5}$$

e o índice de equidade no tempo t é:

$$F(t) = \frac{\left(X(t)\right)^2}{n \sum_{i=1}^{n} \left(x_i(t - d_i^f)\right)^2} \tag{2.6}$$

onde $x_i(t - d_i^f)$ representa a carga do usuário i no tempo $t - d_i^f$.

Similarmente ao modelo original, se assume que a rede envia realimentação uniforme $y(t)$

$$y(t) = \begin{cases} 1 & \text{se } X(t) > c \\ 0 & \text{se } X(t) \le c \end{cases} \tag{2.7}$$

e cada usuário ajusta sua taxa uma vez por RTT por médio de um algoritmo linear binário:

$$x_i(t) = \begin{cases} a_I + b_I x_i(t - d_i^f - d_i^b) & \text{se } y(t - d_i^b) = 0 \\ a_D + b_D x_i(t - d_i^f - d_i^b) & \text{se } y(t - d_i^b) = 1 \end{cases} \tag{2.8}$$

Reescrevendo a equação (9) de acordo com a notação utilizada neste trabalho, se tem:

$$\begin{aligned} x_i(t) = {} & (a_I + b_I x_i(t - d_i^f - d_i^b)) \ (-h\,\mathrm{sgn}(x_1(t - d_i^f) + x_2(t - d_i^f) - c))_{d_i^b} \\ & + (a_D + b_D x_i(t - d_i^f - d_i^b)) \ (uh\,\mathrm{sgn}(x_1(t - d_i^f) + x_2(t - d_i^f) - c))_{d_i^b} \end{aligned} \tag{2.9}$$

Onde $(\bullet)_{d_i^b}$ indica que é utilizado o valor obtido d_i^b anterior. Chamaremos **A_AIMD_G** à equação (2.9), que é uma versão assíncrona do algoritmo de Chiu e Jain.

2.1.3 Modelo de Equilíbrios Virtuais

Alguns estudos sobre do AIMD falam de melhoras segundo a escolha dos parâmetros do mesmo. Chiu e Jain afirmavam que para obter uma convergência mais rápida ao estado equitativo, os valores devem ser $a_I > 0$, $b_I = 1$, $a_D = 0$, e $0 \le b_D < 1$. GORINSKY e VIN (2002) estabelecem que, para obter uma convergência mais rápida ao estado equitativo, se deve escolher $b_I \ge 1$. YANG et al. (2000) falam que o valor $b_I > 1$ pode ter maior velocidade de convergência em forma global. Nestes trabalhos as conclusões foram feitas baseadas em estudo de casos particulares, e a escolha de parâmetros nem sempre verificam tais afirmações.

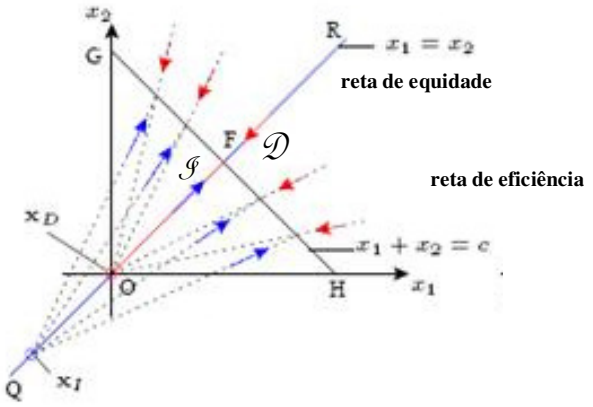
As simulações do algoritmo AIMD demonstram que o algoritmo é sensível aos parâmetros de incremento e decremento. Uma boa escolha dos mesmos assegura uma convergência à ótima equidade. Parâmetros adequados devem eliminar o conflito existente entre os objetivos de grau (ou tempo) de resposta e suavidade; posto que se os parâmetros são selecionados para melhorar o tempo de resposta, se degrada a suavidade e vice versa. BHAYA e KASZKUREWICZ (2007) apresentam uma escolha de parâmetros que leva a ter melhores tempos de resposta, suavidade e índice de equidade baseada em equilíbrios virtuais como será mostrado mais adiante.

Considerando dois usuários compartilhando um único canal, a equação (2.2) do AIMD tem os equilíbrios $\mathbf{x}_I$ e $\mathbf{x}_D$ que se encontram na reta de equidade ($x_1 = x_2$).

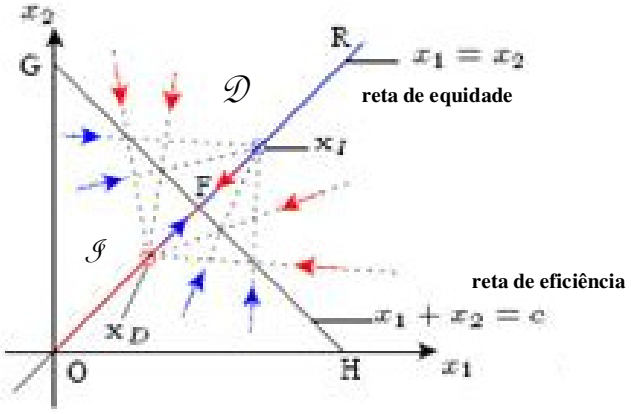
$$\mathbf{x}_I = \left[a_I/(1-b_I),\ a_I/(1-b_I)\right], \qquad \text{para } x_1 + x_2 - c < 0 \qquad (2.10)$$

$$\mathbf{x}_D = \left[a_D/(1-b_D),\ a_D/(1-b_D)\right], \qquad \text{para } x_1 + x_2 - c > 0 \qquad (2.11)$$

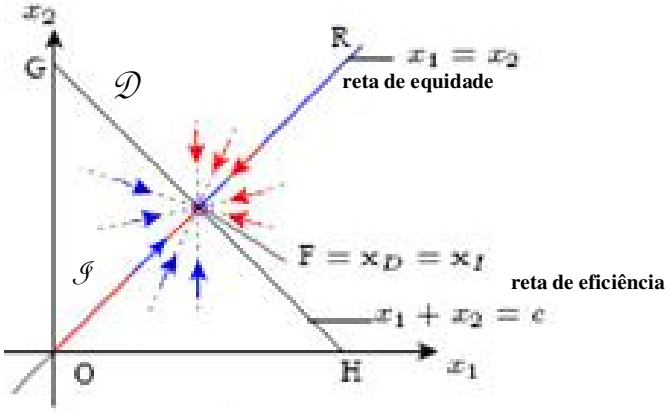
O conceito de equilíbrio virtual aparece formalmente em (BHAYA e KASZKUREWICZ, 2006). O ponto de equilíbrio $\mathbf{x}_I$ para a dinâmica de incremento ($x_i(t) = a_I + b_I x_i(t-1)$) é chamado virtual se estiver localizado na região $\mathcal{D}$ (ver Figura 2.2). Similarmente, o ponto de equilíbrio $\mathbf{x}_D$ para a dinâmica de decremento ($x_i(t) = a_D + b_D x_i(t-1)$) é chamado virtual se estiver localizado na região $\mathcal{I}$. Pelo contrario, o equilíbrio ordinário, para as respectivas dinâmicas ($\mathbf{x}_I \in \mathcal{I}$, e $\mathbf{x}_D \in \mathcal{D}$), são considerados como equilíbrios reais.



A: $\mathbf{x}_I$ é instável, $\mathbf{x}_D$ é assintoticamente estável e virtual (AIMD original de Chiu e Jain)



B: $\mathbf{x}_I$ e $\mathbf{x}_D$ são assintoticamente estáveis e virtuais.



C: $\mathbf{x}_I$ e $\mathbf{x}_D$ são assintoticamente estáveis e reais.

Figura 2.3: Possíveis posições dos equilíbrios $\mathbf{x}_I$ e $\mathbf{x}_D$ e os campos de vetores correspondentes.

A Figura 2.3 apresenta as opções das possíveis localizações dos equilíbrios $\mathbf{x}_I$ e $\mathbf{x}_D$. Os segmentos vermelhos OF representam as possíveis posições do $\mathbf{x}_I$ e os

segmentos azuis QO e FR representam as possíveis posições de $\mathbf{x}_D$. Logo, qualquer escolha de parâmetros a_D e b_D, que faça com que um equilíbrio $\mathbf{x}_D$ fique no segmento OF, será adequado, desde que o equilíbrio seja virtual. Similarmente, qualquer escolha de a_I e b_I, que faça que $\mathbf{x}_I$ fique no segmento FR, é possível, para que ele seja virtual e assintoticamente estável.

Note se que se uma mudança na posição da linha GH, que representa a reta de eficiência, mantendo a virtualidade dos equilíbrios da uma medida de robustez (traduzida numa baixa sensibilidade) a este tipo de perturbação. Isto é a linha GH (c) pode ser colocada dentro do espaço invariante definido pelas líneas paralelas a GH que passem pelos pontos $\mathbf{x}_I$ e $\mathbf{x}_D$ sem que os mesmos deixem de ser virtuais (BHAYA e KASZKUREWICZ, 2007).

Deste modo, segundo (BHAYA e KASZKUREWICZ, 2007), as condições para que a_I e b_I assegurem um equilíbrio virtual assintoticamente estável, para o caso de dois usuários compartilhando um canal de capacidade cap são:

$$a_I > 0, \text{ e } 0 \le b_I < 1, \qquad \text{de forma que } a_I \ge \frac{c}{2}(1-b_I) \tag{2.12}$$

onde a segunda inequação assegura a estabilidade assintótica do equilíbrio. Paralelamente, para garantir que o equilíbrio $\mathbf{x}_D$ seja virtual, ele deve ficar no segmento OF da Figura 4.1 o que leva à seguinte desigualdade:

$$a_D \le \frac{c}{2}(1-b_D), \text{ e } 0 \le b_D < 1 \tag{2.13}$$

Note que, no caso limite em que ambos os equilíbrios virtuais se tornem equilíbrios positivos reais assintoticamente estáveis (ambos coincidem com o ponto de equidade ótimo) $\mathbf{x}_I = \mathbf{x}_D = (c/2, c/2)$ corresponde a escolhas que satisfazem as seguintes desigualdades:

$$\frac{a_D}{(1-b_D)} = \frac{c}{2},\ 0 \le b_D < 1;\ \frac{a_I}{(1-b_I)} = \frac{c}{2},\ 0 \le b_I < 1 \tag{2.14}$$

Os parâmetros b_I e b_D devem ser pequenos, escolhidos apropriadamente, e logo os parâmetros a_I e a_D, segundo (2.12) e (2.13), de tal modo que os equilíbrios sejam virtuais e não muito distante, porque isto assegura uma boa suavidade (BHAYA e KASZKUREWICZ, 2007).

O modelo AIMD que utiliza as equações (2.12) e (2.13), para escolher os parâmetros de incremento e decremento do algoritmo AIMD, será referenciado ao longo deste trabalho como Equilíbrios Virtuais ou VE (pelo acrônimo em inglês de *Virtual Equilibria*).

As equações (2.12) e (2.13) asseguram uma boa convergência somente se os usuários têm conhecimento do valor da capacidade do canal compartilhado e do número de usuários. Entretanto, na Internet os usuários não têm este conhecimento. No capítulo 3 serão apresentados dois métodos para obter esta informação de forma a poder utilizar os Equilíbrios Virtuais para um algoritmo de controle de congestionamento da Internet.

A proposta de BHAYA e KASZKUREWICZ (2007) pode ser e estendida facilmente ao ajuste ponderado de equidade e controles AIMD heterogêneos proposto em (GREVOS e CROWCROFT, 2004) com as vantagens demonstradas para casos não ponderados; como pode verse nos experimentos com não homogeneidade e com pesos diferentes na alocação a cada usuário de (BHAYA e KASZKUREWICZ, 2007).

2.2 Modelos inspirados em otimização

Alguns autores (FAN et.al., 2005, GOLESTANI et.al., 1998, KELLY, 2003, KELLY et.al, 1998, LOW e LAPSLEY, 1999) consideram ao problema de controle de congestionamento como um problema de que os usuários desejam maximizar uma utilidade obtida quando uma taxa de dados é alocada a eles.

A análise e concepção de protocolos de rede em larga escala se tornaram possíveis com o quadro de otimização e o modelo de dualidade (JAMALI et.al., 2007). KELLY et. al. (1998) formularam o problema de alocação de largura de banda como uma maximização da utilidade sobre taxas das fontes com limitações de capacidade. Um algoritmo distribuído também é fornecido por KELLY et. al. (1998) para resolver globalmente a função de penalização do problema de otimização. Este algoritmo é chamado o algoritmo *primal* onde as fontes adaptam suas taxas dinamicamente, e os preços dos canais de comunicação são calculados por uma função estática das taxas de chegada.

KELLY et.al. (1998), KELLY (2003) propõe um algoritmo de controle de congestionamento, que divide o problema em um subproblema de usuário e um subproblema da rede. O modelo de rede de KELLY (1998, 2003) é a interconexão de fontes de informação e de ligações de comunicação através das matrizes do roteamento como mostrado na Figura 2.4.

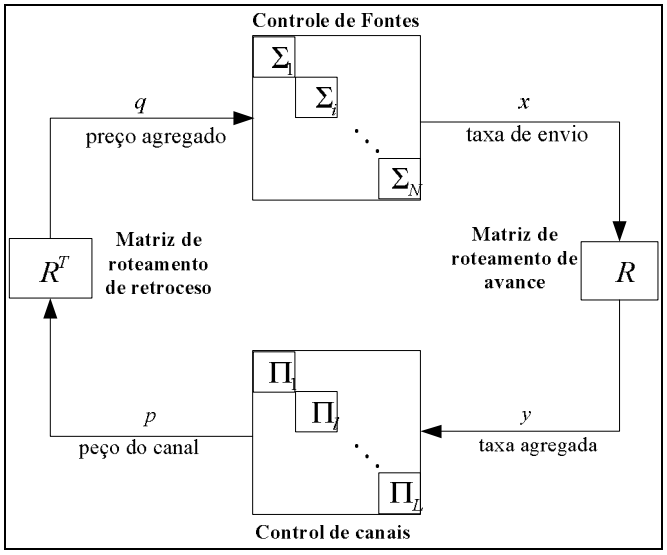


Figura 2.4: Modelo de rede de Kelly.

Os pacotes de cada usuário ou fonte são enviados através de canais com uma taxa de valor agregado do canal

$$y = R_f x \tag{2.15}$$

onde R_f é a matriz de roteamento de avance. Cada canal j tem uma capacidade fixa c_j, e baseado em seu congestionamento e tamanho da fila, um preço do canal p_j é computado:

$$p_j = h_j(y_j), \qquad j = 1, \cdots, L \tag{2.16}$$

A informação de preço do canal é logo enviado de novo a cada fonte com o valor do preço agregado da fonte,

$$q = R_b p \tag{2.17}$$

onde $R_b = R_f^T$, dado que os canais somente realimentam informação do preço aos usuários que os utilizam.

KELLY et.al. (1998), KELLY (2003) formulam o modelo de controle de fluxo como uma combinação de uma otimização estática e um problema de estabilização dinâmica. O problema de otimização estática calcula o equilíbrio desejado maximizando a soma das funções de utilidade das fontes $U_i(x_i)$, cumprindo com as restrições de capacidade dos canais:

$$\max_{x\geq 0} \sum_{i=1}^{N} U_i(x_i) \text{ sujeito a } \underbrace{Rx}_{y} \leq c \tag{2.18}$$

O problema dinâmico é para desenhar a lei de atualização da taxa da fonte baseado no preço agregado e a lei de atualização do preço do canal baseado na taxa agregada, para garantir a estabilidade do equilíbrio. Para este problema, Kelly introduziu dois algoritmos dinâmicos: o algoritmo *Primal* e o algoritmo *Dual*.

O algoritmo *Primal* consiste em uma lei de atualização de fontes de primeira ordem, e uma função de penalidade para o canal para garantir que a taxa agregada permaneça abaixo de sua capacidade:

$$\dot{x}_i = \kappa_i(U_i^{'}(x_i) - q_i), \qquad p_j = h_j(y_j) \tag{2.19}$$

As funções de penalidade $h_l(y_l)$ são designadas para forçar que as restrições da capacidade dos canais $y_l \leq c_l$, $l = 1,\cdots,L$, isto é, manterem o valor agregado y_l abaixo de sua capacidade c_l.

O algoritmo *Dual* consiste numa atualização estática e uma atualização de preço dinâmica de primeira ordem:

$$x_i = U_i^{'-1}(q_i), \qquad p_j = \gamma_j(y_j - c_j)^{+}_{p_j} \tag{2.20}$$

onde a projeção positiva $(\bullet)^+$ para uma função geral $f(\bullet)$ está definida como

$$(f(x))^{+}_{x} := \begin{cases} f(x) & \text{se } x > 0, \text{ ou } x = 0 \text{ e } f(x) \geq 0 \\ 0 & \text{se } x = 0 \text{ e } f(x) < 0 \end{cases}$$

De (2.20), o único equilíbrio para a lei de controle *Dual* é obtida das equações

$$q_i^* = U_i^{'}(x_i^*), i,\cdots,N \tag{2.21}$$

$$p_l^* \begin{cases} = 0 & \text{se } y_l^* \leq c \\ \geq 0 & \text{se } y_l^* = c_l \end{cases} \qquad l = 1,\cdots,L \tag{2.22}$$

As quais, como é mostrado em (KELLY et.al., 1998), corresponde a solução do problema de otimização (2.18), onde $p_l's$ atuam como os multiplicadores de Lagrange para as restrições de capacidade. Para a lei de controle *Primal* (2.19) o equilíbrio obtido de

$$q_i^* = U_i^{'}(x_i^*), \qquad i,\cdots,N \tag{2.23}$$

$$p_l^* = h_l(y_l^*), \qquad i,\cdots,L \tag{2.24}$$

aproxima a condição de otimização (2.21)-(2.22), com ajuda das funções de penalidade $h_l(y_l)$. A estabilidade destes algoritmos e suas extensões foram estabelecidas em (DEB e SRIKANT, 2002, PAGANINI, 2002).

LOW et. al. (2002) mostraram que, dado um modelo de controle de fonte (TCP), se pode derivar dele a função de utilidade associada com o protocolo. Assim, por exemplo, a função de utilidade de TCP Reno é $U_i(x_i) = -\frac{a^2}{RTT_i^2 x_i}$. O valor da constante a, próximo a 1, há sido encontrado empiricamente dependendo dos detalhes de implementação, segundo as variações de TCP. Do mesmo modo, a função de utilidade de TCP Vegas é $U_i(x_i) = \alpha_i d_i \log x_i$, onde α_i é um parâmetro do protocolo e d_i é o retardo de propagação do *RTT*.

FAN et.al. (2005) estendem o modelo de Kelly para usuários não cooperativos e desenvolvem um algoritmo baseado em nós de fronteira de ajuste de preços usando ferramentas da teoria de controle, que é aplicável aos tipos diversos de redes.

2.3 Modelos inspirados em sistemas biológicos

Os modernos sistemas de informação de rede, como a Internet, atingiram tal nível de complexidade que tornaram-se difícil de implementar, gerenciar e manter funcionando corretamente por meio de técnicas tradicionais (JAMALI et.al., 2007). Parte deste problema é deve-se ao tamanho ilimitado que tais sistemas podem alcançar, com milhões de usuários e dispositivos interconectados. Outro aspecto deste problema corresponde ao comportamento global extremamente complexo entre componentes, mesmo quando seus números são modestos. Enquanto estes problemas são bastante recentes para os modernos sistemas de informação, eles são antigos do ponto de vista da biologia.

Muitos sistemas biológicos são adaptáveis, tem memória e podem lidar com situações imprevistas, variações no ambiente ou presença de agentes anormais. Sistemas biológicos são capazes de lidar com muitos dos desafios de construir sistemas de

informação robusta, para implantação em ambientes de redes altamente dinâmicos, com eficiência e elegância.

Baseadas nesta observação, abordagens bio-inspiradas foram propostas nos últimos anos como estratégia para lidar com a complexidade de tais sistemas. Vários exemplos (JAMALI et.al., 2007) estão disponíveis na área de informática, na qual os conceitos biológicos são considerados modelos para imitar. Cada um desses exemplos concentra-se em um aspecto biológico diferente e é aplicado para resolver ou para otimizar um problema tecnológico específico.

No campo dos algoritmos de controle de congestionamento bio-inspirados, Murata et.al. (IGUCHI et al., 2005) propõem um método no qual o tamanho da janela de uma conexão TCP é alterado com base nas larguras de banda disponíveis e reais do caminho fim a fim da rede. A informação da largura de banda é obtida por uma técnica eles denominada técnica medição *inline* da rede. Os autores utilizam o modelo de concorrência Lotka–Volterra da biofísica para atualizar o tamanho da janela de congestionamento.

JAMALI et.al. (2007) propõem que a janela de congestionamento ($cwnd = W$) de toda fonte possa ser considerada como o tamanho da população de uma espécie. Assim, a fim de controlar o congestionamento na comunicação de redes, o tamanho da população W_i deve ser controlado. Isto significa que o problema de controle da população na natureza pode ser mapeado ao problema de controle de congestionamento na comunicação de redes.

O trabalho de JAMALI et.al. (2007) propõe que a Internet seja imaginada como um ecossistema e o controle de congestionamento como o controle de população. Assim, todas as estratégias de controle de população da natureza podem ser aplicadas à questão de controle de congestionamento da Internet. Neste contexto, dois modelos de controle de população são propostos (JAMALI et.al, 2007); o modelo predador-presa e o modelo de concorrência. Estes modelos também podem ser combinados para a obtenção de um modelo híbrido.

Considere uma rede com um conjunto de k nós de origem e um conjunto de k nós de destino. Denote-se $S = \{S_1, S_2, \cdots, S_k\}$ como o conjunto de nós de origem e $D = \{D_1, D_2, \cdots, D_k\}$ como o conjunto de nós de destino. Os com tempos totais de percurso (RTT) são idênticos. Assim, o modelo de rede proposto por JAMALI et.al., (2007) consiste em um canal ou enlace de gargalo de uma LAN para uma WAN e usa

um algoritmo baseado em janela para o controle de congestionamento. O canal de gargalo tem uma capacidade de B pacotes por RTT. A janela de congestionamento (W) é um limite do lado do remetente da quantidade de dados que o remetente pode transmitir na rede antes de receber uma confirmação (ACK).

Esta rede pode ser vista como um ecossistema de habitantes conectados em grande variedade, tais como roteadores, hosts, links, sistemas operacionais, etc. Deste modo, assume-se a existência de espécies, tais como: o tamanho da janela de congestionamento (W), o descarte de pacotes (P), o tamanho das filas dos roteadores (q) e a utilização do canal (u). O tamanho destes elementos de rede refere-se ao seu tamanho de população em um ecossistema de Internet. A Figura 2.5 mostra a tipologia de um ecossistema Internet numa perspectiva de controle de congestionamento. Neste ecossistema, as espécies estão interagindo e o tamanho da população de cada espécie é afetado por essas interações.

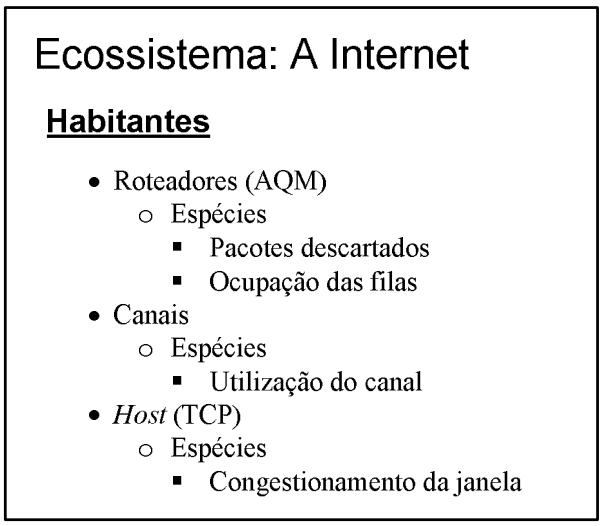


Figura 2.5: Tipología do ecosistema Internet.

Seja W_i (tamanho da janela de congestionamento da conexão i) a população de W na fonte S. Percebe-se que, se o tamanho da população desta espécie aumenta, em seguida, o número de pacotes enviados é inflacionado. Assim, para evitar o congestionamento, o tamanho da população de W (todos os W_i) deve ser controlado. Isso significa que o problema de controle de população na natureza pode ser aplicado ao problema de controle de congestionamento em uma rede de comunicação. A natureza utiliza táticas como predadores, parasitas, concorrência, etc., para controlar o tamanho

da população da espécie. Assim, uma metodologia para usar dinâmica de predação e concorrência para controlar o tamanho da população da espécie W.

A fim de esclarecer a similaridade entre mecanismo de controle de congestionamento TCP/AQM e a interação predador-presa, o seguinte analise da rede TCP/AQM deve ser feito:

(1) Na ausência de descarte de pacotes a janela de congestionamento (W) irá crescer.

(2) Na ocorrência de descarte de pacote o tamanho de janela de congestionamento diminuirá.

(3) A contribuição da taxa de entrada de pacote para ao crescimento do descarte de pacotes é proporcional à intensidade de tráfego disponível, bem como ao próprio descarte de pacotes.

(4) Na ausência de fluxo de pacote, a taxa de descarte de pacotes diminuirá.

(5) A largura de banda do enlace impõe um limite ao tamanho máximo da janela.

O comportamento anterior está próximo a interação predador-presa o que motiva ao uso da abordagem predador-presa para controlar a população W.

Seja P outra espécie que vive no roteador e pode caçar indivíduos da espécie W. Desde que W contenha k espécies, supõe-se que P também tem k espécies, $P_1, P_2, \cdots, P_k$ no roteador congestionado. P_i pode caçar a todos os indivíduos W, mas pode haver uma preferência de ponderação: P_i talvez tenha maior ter preferência de caça sobre W_i que sobre todos os outros indivíduos W_j ($j \neq i$). Da mesma forma, o mesmo relacionamento também pode ser imaginado entre o tamanho da fila no roteador congestionado (q) e o tamanho de janela de congestionamento (W) das fontes que compartilham essa fila. Assim, a interação de (q,W) também é considerado uma interação predador–presa.

JÁMALI et.al. (2007) propõem, então, o seguinte modelo predador-presa para controle de congestionamento bio-inspirado:

$$\frac{dW_i}{dt} = W_i\left(h_i - \sum_{j=1}^{k} b_{ij}P_j - r_j q\right) \qquad \text{onde } i = 1,\cdots,k \tag{2.25}$$

$$\frac{dP_i}{dt} = P_i\left(\sum_{j=1}^{k} c_{ij}W_j - d_i\right) \qquad \text{onde } i = 1,\cdots,k \tag{2.26}$$

$$\frac{dq}{dt} = q\left(\sum_{j=1}^{k} e_{ij} W_j - m\right) \qquad \text{onde } m = Min\left(B, q + \sum W_i\right) \tag{2.27}$$

Neste conjunto de equações diferenciais, existem k espécies presa e k espécies predadoras. Os parâmetros do conjunto de equações são definidos como se segue: h_i é a taxa de crescimento do W_i na ausência de P e q ; b_{ij} é taxa de decremento por encontro com W_i devido a P_j ; r_i é o taxa de decremento por encontro de W_i devido a q ; d_i é a taxa de decremento de P_i na ausência de W ; c_{ij} é a eficiência de W_j ser predado em P_j ; e_j é a eficiência de W_j ser predado em q ; e m é definido como $Min\left(B, q + \sum W_i\right)$.

O segundo modelo proposto por JÁMALI et.al. (2007) se inspira em fluxos da rede que são essencialmente competitivos no sentido que desejam dominar os recursos da rede para maximizar sua qualidade de serviço. Por analogia, todas as espécies W_i que compartilham um canal de gargalo incorporam competência inter-espécies e intra-espécies para maximizar a sua própria parte de largura de banda do canal. Então, o modelo de competitividade pode ser descrito pela equação (2.28).

$$\frac{dW_i}{dt} = W_i\left(h_i - \sum_{j=1}^{k} f_{ij} \frac{W_j}{l_i}\right) \qquad \text{onde } i = 1, \cdots, k \tag{2.28}$$

Quando h_i s são as taxas de natalidade lineares, l_i s são as capacidades ecológicas sustentadas e f_{ij} mede o efeito concorrencial W_j de sobre W_i. Um modelo matemático híbrido que inclui a predação de W por P e q assim como a concorrência entre W_i s pode ser desenvolvido pela combinação de equações (2.25) até (2.28). Esta integração pode ser descrita pelas equações (2.29) até (2.31), nas quais foi considerada a concorrência intra-espécies de P_i s.

$$\frac{dW_i}{dt} = W_i\left(h_i - \sum_{j=1}^{k} b_{ij} P_j - r_j q - \sum_{j=1}^{k} f_{ij} \frac{W_j}{l_i}\right) \text{ onde } i = 1, \cdots, k \tag{2.29}$$

$$\frac{dP_i}{dt} = P_i\left(\sum_{j=1}^{k} c_{ij}W_j - d_i - \delta\frac{P_i}{R}\right) \tag{2.30}$$

$$\frac{dq}{dt} = q\left(\sum_{j=1}^{k} e_j W_j - m\right)\varepsilon \tag{2.31}$$

Onde , $i = 1,\cdots,k$, h_i s, l_i s, f_i s, b_i s, c_{ij} s, d_i s e m são definidos como as equações (2.25) até (2.27), e (2.28). δ mede o efeito da concorrência intra-espécies das espécies P_i e ε é usado para suavizar as alterações de q. Estes modelos conduzem a um modelo de controle de congestionamento que HBICC (*Hybrid BIO-Inspired Congestion Control*). HBICC diz que a população W_i aumentará exponencialmente, mas que esse crescimento vai ser inibido por dois fatores: predação por P e q e a concorrência com W.

HORIE et.al. (1998), encontraram uma relação entre otimização e modelos do tipo predador-presa. O trabalho apresentado por HORIE et.al. (1998) demonstra que um problema de otimização pode ser solucionado por um sistema gradiente que se aproxima a um problema do tipo predador-presa.

HORIE et.al., (1998) estudaram a busca de pontos de equilíbrio de Nash (NE) em jogos de P jogadores não cooperativos e propõem um modelo dinâmico de procura chamado *Parallel Steepest Descent Method with Braking operators* (PSDMB).

Assim, para onde P jogadores não cooperativos com um vetor decisão individual n^p -dimensional $\vec{x}^p \in \Re^{n^p}$ restringido sobre um conjunto $X^p \subset \Re^{n^p}$ e uma função objetivo $E_p(\vec{x}^1,\cdots,\vec{x}^P)$ com funções quadráticas de objetivo:

$$E_p(\vec{\bar{x}}^1,\cdots,\vec{\bar{x}}^p) = \frac{1}{2}\vec{x}^{pT}W^p\vec{x}^p + \vec{\theta}^T\vec{x}^p + \sum_{1=1,q\neq p}^{1}\vec{x}^{pT}V^{pq}\vec{x}^q \tag{2.32}$$

o proposto PSDMB resulta em:

$$\frac{dx_i^p(t)}{dt} = -g(x_i^p)\left\{\sum_{j=1}^{n_p} w_{ij}^p x_j^p(t) + \theta_i^p + \sum_{q=1,q\neq p}^{p}\sum_{j=1}^{n_p} v_{ij}^{pq}x_j^q(t)\right\}$$

$$\text{com } i = 1,\cdots,n_p \ , \ p = 1,\cdots,P \tag{2.33}$$

onde w_{ij}^p , v_{ij}^{pq} , θ_i^p são coeficientes e a função $g(x_i^p(t))$ é o operador de freio que reduz o gradiente ao zero quando a variável $\vec{x}^p(t)$ se aproxima da fronteira da restrição X^p.

Se $n_p = 1, p = 1, \cdots, P$, pode-se omitir todos os sub-índices i, j nas variáveis $\{x_i\}$ e nos coeficientes $\{w_{ij}^p\}$, $\{v_{ij}^{pq}\}$, $\{\theta_i^p\}$ no PSDMB para funções quadráticas. Logo, os coeficientes podem ser substituídos por $w^p = -a_{pp}$, $\theta^p = -a_{p0}$, e $v^{pq} = -a_{pq}$.

A forma da proposta PSDMB para funções quadráticas (equação 2.33) com operador de freio $g(x) = x$ torna-se:

$$\frac{dx_i^p(t)}{dt} = x^p(t)\left\{a_{p0} + \sum_{q=1}^{P} a_{pq} x^q(t)\right\} \qquad \text{com } p = 1, \cdots, P \tag{2.34}$$

Esta equação é a conhecida como equação generalizada de Lotka-Volterra. Esta relação sugere que o problema de controle de congestionamento seja formulado como um sistema de equações diferenciais do tipo Lotka-Volterra.

Síntese do capitulo

Este capítulo apresentou três aspectos da modelagem do problema do controle de congestionamento. Primeiramente, foi apresentado o modelo tradicional AIMD que é utilizado na camada de transporte para o controle de congestionamento. Apresentou-se a formulação utilizada por CHIU e JAIN (1989) para atrasos homogêneos e a modelagem para atrasos heterogêneos estudada por GORINSKY e VIN (2002) e GORINSKY (2004). Neste contexto foi introduzido o modelo Equilíbrios Virtuais (BHAYA e KASZKUREWICZ, 2006) que defende que uma escolha adequada de parâmetros das dinâmicas de incremento e decremento do algoritmo AIMD baseado em equilíbrios virtuais.

Em seguida, apresentou-se brevemente a modelagem do problema de congestionamento a partir da perspectiva de otimização (KELLY et.al. 1998 e KELLY 2003). Finalmente, foram apresentados os modelos inspirados em sistemas biológicos (JAMALI et.al. 2007).

Da mesma forma, o capítulo mostra que o problema de controle de congestionamento pode ser modelado por um sistema de otimização com restrições (KELLY et.al., 1998, KELLY, 2003), utilizando o método de penalidades, que pode ser reformulado como um problema de otimização sem restrições e resolvido mediante um sistema de gradiente que corresponde a um problema do tipo predador-presa (HORIE et.al., 1998) amplamente estudado na literatura.

No capítulo seguinte serão apresentadas novas propostas de algoritmos inspirados no AIMD e baseados em otimização.

3. Algoritmos propostos

Os parâmetros atuais utilizados nas implementações clássicas do AIMD nos protocolos de controle de congestionamento como TCP são fixos e estáticos, o que pode significar um controle eficiente numa configuração de rede, mais não precisamente eficiente em outra. Uma opção de melhora poderia observar-se se ajustando dinamicamente os parâmetros segundo configuração da rede e/ou informação da mesma se obteria melhores resultados que usando parâmetros estáticos.

Visto por outra perspectiva, o controle de congestionamento pode ser considerado como um problema de otimização, como fora apresentado por KELLY et.al. (2003). Assim, um novo algoritmo de controle, para calcular o equilíbrio desejado maximizando uma função objetivo segundo restrições de limitação da capacidade da rede, pode ser proposto.

Neste capitulo, serão apresentados: dois novos algoritmos, considerando o ajuste dinâmico de parâmetros do algoritmo AIMD, e duas propostas, que implicam otimização inspirados no modelo de Kelly.

3.1 Algoritmos inspirados em AIMD

A idéia das novas propostas baseadas no modelo AIMD é ajustar dinamicamente os parâmetros do modelo AIMD seguindo a proposta do modelo de Equilíbrios Virtuais da seção 2.1.3. Como já fora mencionado, para poder aplicar o método, precisamos dos dados da capacidade da rede e do número de usuários. Estes dados podem ser obtidos dinamicamente e passados aos usuários para calcular o tamanho da janela de congestionamento. Se propõem assim dois novos algoritmos: **Alg_WVE (Westwood Equilíbrios Virtuais)** e **Alg_VE (Equilíbrios Virtuais)** descritos a seguir:

3.1.1 Algoritmo Westwood Equilíbrios Virtuais

As equações (2.12) e (2.13) do modelo de Equilíbrios Virtuais asseguram uma boa convergência somente se os usuários têm conhecimento do valor da capacidade do canal compartilhado e do número de usuários. Para poder aplicar o novo método que envolve os equilíbrios virtuais, devemos contar com uma forma de estimativa da capacidade da rede ou largura de banda disponível para a transmissão de dados.

Existem várias formas de estimar a largura de banda (WANG, 2002, XU, 2008). Sem embargo, este trabalho propõe utilizar a estimativa da capacidade da rede em bits (largura de banda) realizada pelo algoritmo do protocolo TCP Westwood (MASCOLO et.al., 2000, UCLA COMPUTER SCIENCE DEPARTMENT, 2008). Em seguida, usar esta estimativa para conseguir uma estimativa da capacidade da rede em pacotes (cap). Assim uma proposta de protocolo utilizaria esta estimativa, dividi-la pelo numero de usuários e assim obter cap/n das equações (2.12) e (2.13) para recalcular os parâmetros a_I e a_D com b_I e b_D fixos.

Como será visto em detalhe na seção 4.2, MASCOLO et.al. (2000) propõem o seguinte filtro de tempo discreto, o qual é obtido discretizando um filtro de passo-baixo continuo, usando a aproximação de Tustin. Obtém se assim:

$$\hat{b}(k) = \frac{\frac{2\tau}{t(k)-t(k-1)} - 1}{\frac{2\tau}{t(k)-t(k-1)} + 1}\hat{b}(k-1) + \frac{b(k)+b(k-1)}{\frac{2\tau}{t(k)-t(k-1)} + 1}$$

onde $\hat{b}(k)$ é a medida filtrada da largura de banda disponível no tempo $t = t(k)$ e $1/\tau$ é a frequência limite do filtro. Nota-se que, $t(k)-t(k-1)$ é considerado o intervalo do sinal de retroalimentação ou ACK. O $\hat{b}(k)$ dá a medida em bits da capacidade da rede.

O segundo dado necessário para aplicar a nova proposta é o número de usuários que se encontram compartilhando a rede de comunicação. O valor do número de usuários se pode obter dos roteadores da rede. O roteador pode manter uma contagem do numero de usuários, contando o identificador do fluxo do cabeçalho IP do pacote em transito (versão Ipv.6), e enviar essa informação ao usuário destino, que reenviaria para o usuário remetente. Assim, com a duas informações da capacidade da rede e número de usuários podemos aplicar as equações (2.12) e (2.13) para ajustar os parâmetros do algoritmo AIMD.

Chamaremos a este algoritmo **Alg_WVE (Westwood Equilíbrios Virtuais)**. As Figuras 3.1 – 3.3 apresentam diagramas do algoritmo em dois níveis de detalhe. O primeiro nível mostra o ciclo geral de toma de decisões para a transmissão de dados. O nível inferior (ou segundo nível) apresenta o detalhe do cálculo da capacidade de transmissão da rede e o ajuste de parâmetros do AIMD e o correspondente ajuste da janela de congestionamento. Estes diagramas servirão de base para a implementação do correspondente protocolo (Capítulo 4).

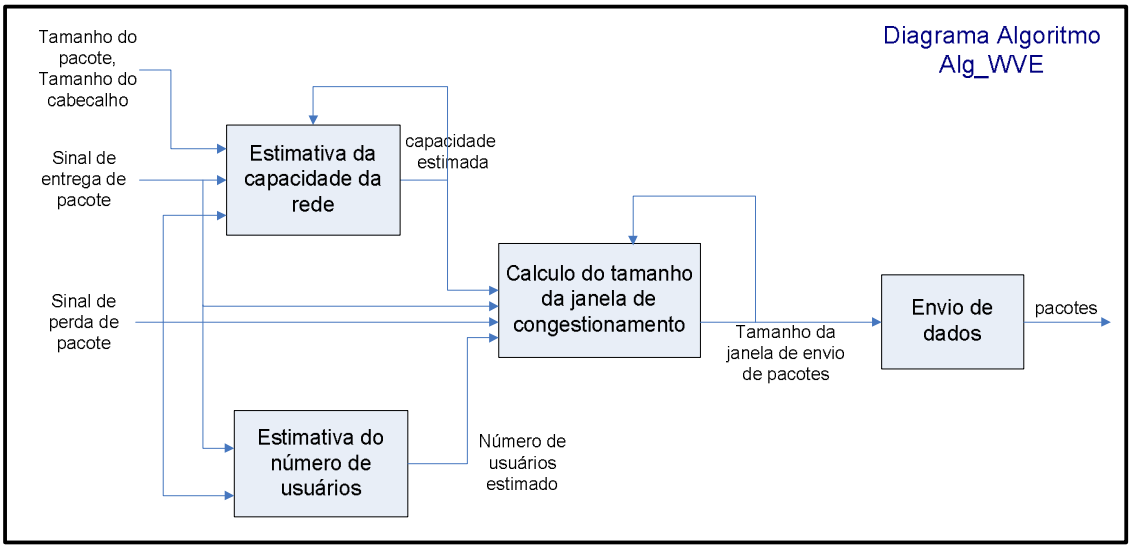


Figura 3.1: Diagrama do algoritmo Alg_WVE - Nível 1.

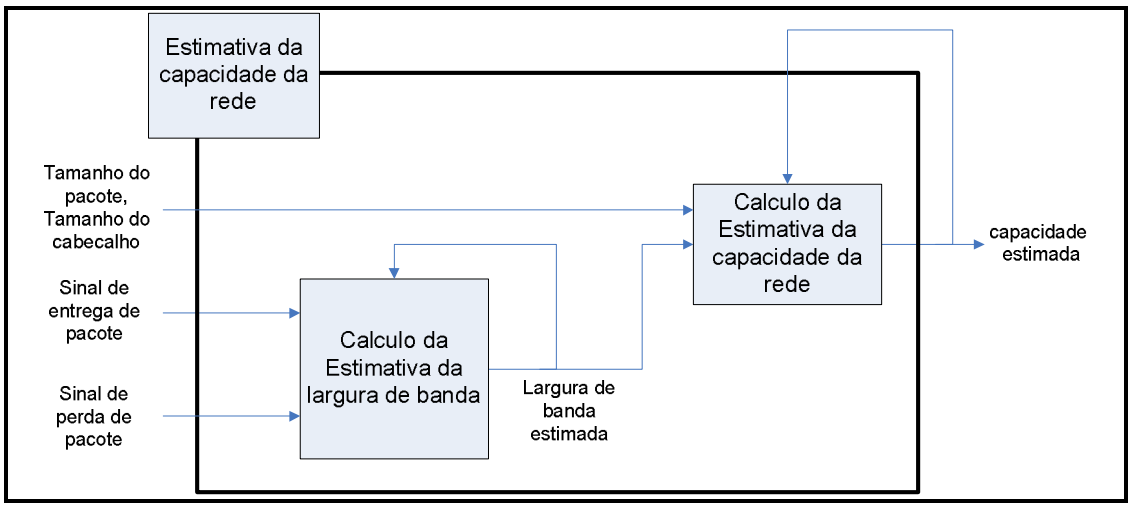


Figura 3.2: Alg_WVE – Nível 2: Estimativa da capacidade da rede.

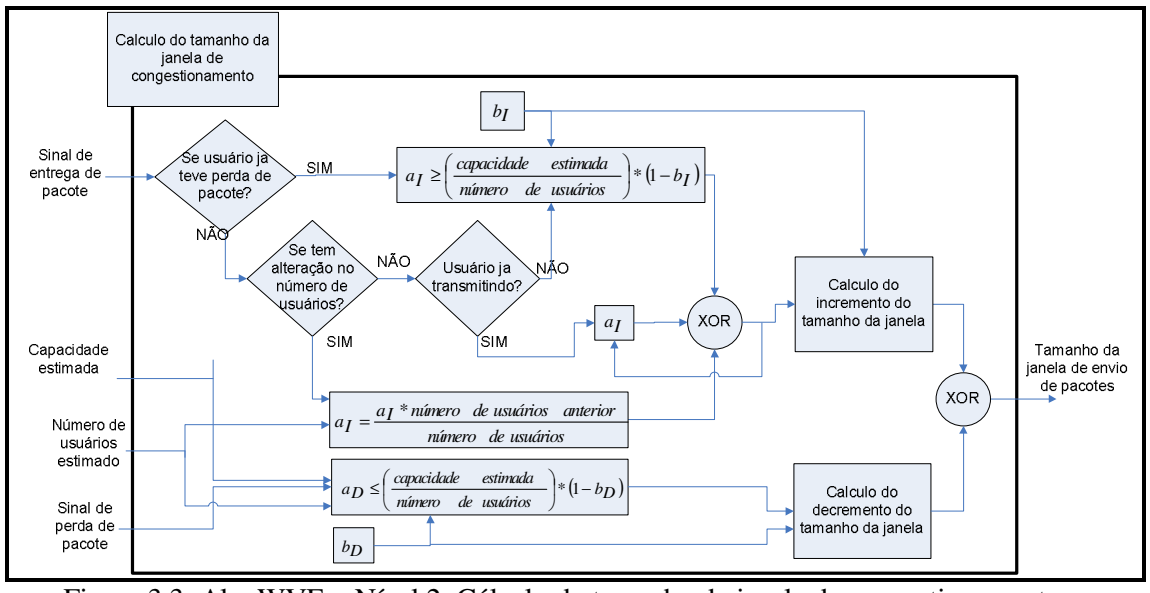


Figura 3.3: Alg_WVE – Nível 2: Cálculo do tamanho da janela de congestionamento.

3.1.2 Algoritmo Equilíbrios Virtuais

Como falado anteriormente, na Internet os usuários não têm o conhecimento do valor da capacidade do canal compartilhado e do número de usuários. Para resolver este problema BHAYA e KASZKUREWICZ (2006) propõem um algoritmo adaptativo chamado VE adaptativo com o seguinte pseudocódigo.

PSEUDOCODIGO DO ALGORITMO VE-adaptativo

//inicialização dos parâmetros

$a_I = 0,\ b_I = 0.9,\ b_D = 0.9,\ \mu_{a_I} = 0.001,\ \mathbf{x}_0 \approx 0$

//adaptação de parâmetros

while não congestionamento

$a_I \leftarrow a_I + \mu_{a_I}$;

$x_i \leftarrow a_I + b_I x_i,\ \forall i$

end while

//cálculo do equilíbrio virtual da dinâmica de incremento (*increase dynamics*).

$(\mathbf{x}_I)_i = a_I / (1 - b_I),\ \forall i$

//cálculo do equilíbrio virtual da dinâmica de decremento (*decrease dynamics*) = 60% do equilíbrio virtual da dinâmica de incremento.

$(\mathbf{x}_D)_i = 0.6\ \mathbf{x}_I,\ \forall i$

//cálculo do parâmetro de decremento do equilíbrio virtual.

$a_D = (1 - b_D)\mathbf{x}_D$

//fim da fase de adaptação

//fase regular do algoritmo AIMD

If não congestionamento

$x_i \leftarrow a_I + b_I x_i,\ \forall i$

else

$x_i \leftarrow a_D + b_D x_i,\ \forall i$

end if

O algoritmo VE adaptativo é muito simples e implementável na camada de transporte, devido a que, a única variação aplicável ao algoritmo do protocolo TCP tradicional seria uma fase de ajuste de parâmetros até o primeiro sinal de congestionamento.

Seguindo esta idéia, este trabalho propõe uma segunda aproximação baseada no conceito de que, quando uma perda de pacote acontece, a rede atinge a capacidade máxima permitida. Assim, se incrementa o parâmetro a_I até atingir uma perda de pacote e se utiliza esta informação na equação (2.12), para deduzir o valor c/n fazendo:

$$\frac{c}{n} = \frac{a_I}{(1-b_I)} \tag{3.2}$$

Com este valor de c/n se pode calcular o parâmetro a_D da dinâmica de decremento segundo (2.13), fazendo:

$$a_D \leq \frac{c}{n}(1-b_D) = \frac{a_I}{(1-b_I)}(1-b_D) \tag{3.3}$$

Depois de uma perda, o valor de a_I pode ficar constante ou fixado a uma percentagem um pouco menor. Logo, cada vez que se tem uma perda de pacotes a_D pode ser recalculado utilizando (3.3).

Nota-se que, quando se tem uma variação do número de usuários se deve recalcular o parâmetro a_I . Chamaremos a este algoritmo **Alg_VE (Equilíbrios Virtuais)**.

O valor do número de usuários pode ser obtido dos roteadores da rede, do mesmo modo que é realizado no algoritmo Alg_WVE.

Os diagramas do algoritmo em dois níveis de detalhe são detalhados a seguir. Estes diagramas servirão de base para a implementação do correspondente protocolo (capítulo 4).

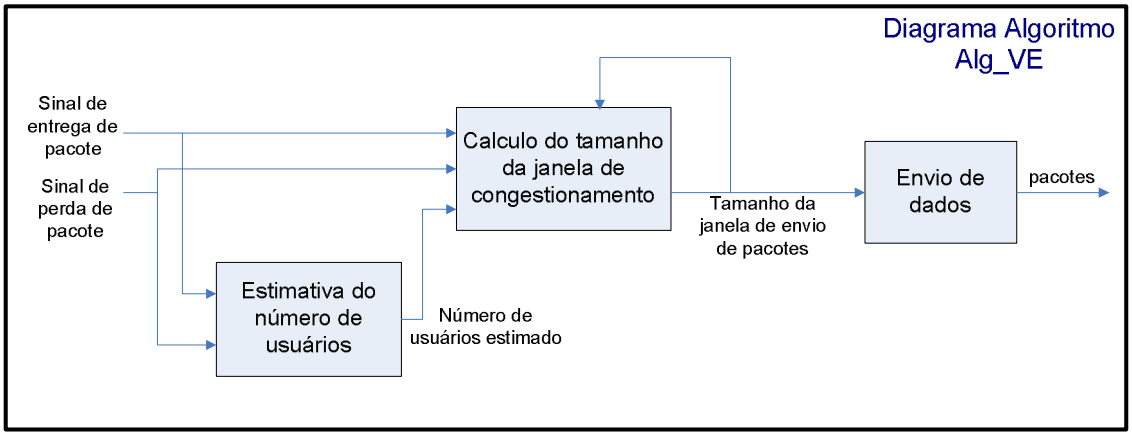


Figura 3.4: Diagrama do algoritmo Alg_VE - Nível 1.

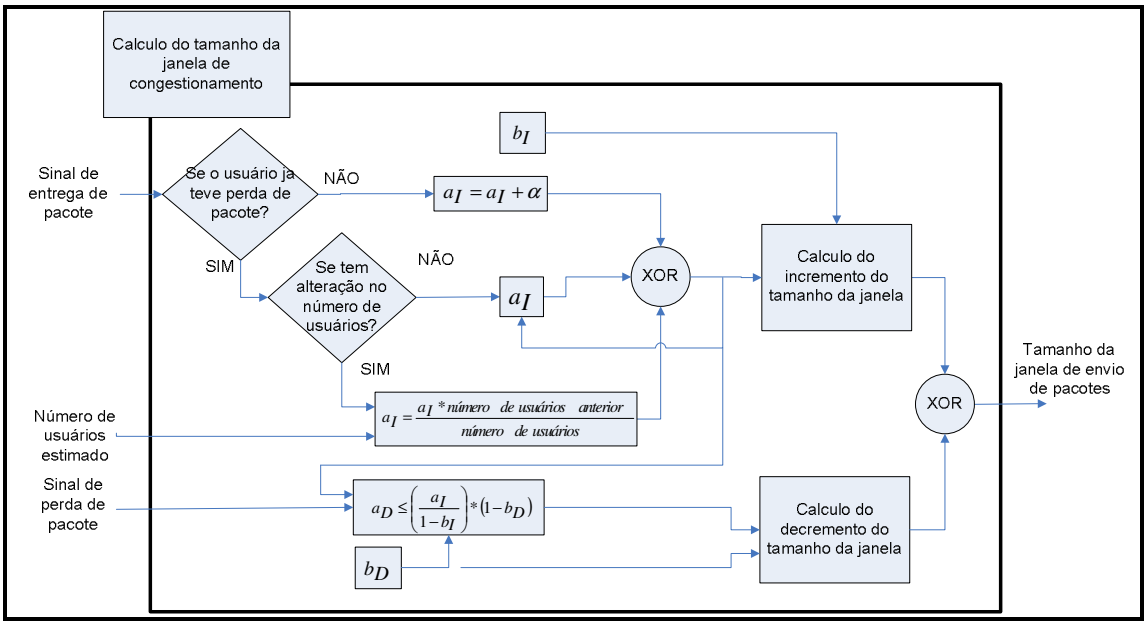


Figura 3.5: Algoritmo Alg_VE – Nível 2: Cálculo do tamanho da janela de congestionamento.

3.2 Algoritmos baseados em otimização

O modelo *AIMD* é um modelo simples que visa alocar recursos de uma forma equitativa, entre usuários que compartilham um único canal, o que, pode ser visto como um paradigma. Outro paradigma, não tratado neste trabalho, é o do KELLY et.al. (1998). Esses autores formulam o problema de alocação de recursos na forma de um problema de otimização de utilidades (SRIKANT, 2003).

O AIMD é um mecanismo de controle de congestionamento que como os esquemas de controle que não usam uma discrepância da equidade ou medida do erro da equidade não podem assegurar a convergência a um estado justo do equilíbrio da rede (KASZKUREWICZ e BHAYA, 2006).

Esta seção propõe uma formulação alternativa do problema de alocação equitativa, que toma como referência o modelo proposto por KELLY et.al. (1998, 2003, FAN, et.al., 1998, 2006), que se baseia em conceitos de otimização, porém utilizando o conceito simples de equidade discutido acima, ao invés de utilizar o conceito de equidade max-min de Kelly e outros, para permitir comparações com o modelo AIMD.

Veremos em seguida que o problema de controle de congestionamento pode ser formulado como um problema de otimização com restrições. Utilizando o método de penalidades, que pode ser reformulado como um problema de otimização sem restrições e resolvido mediante um sistema de gradiente, método que apresenta convergência global (GLAZOS et.al., 1998) sob condições bastante fracas. Assim, baseado numa

proposta de controle, o problema de controle de congestionamento pode ser formulado e resuelto.

A eficiência do método vai depender da escolha da função objetivo adequada. Esta seção apresenta duas propostas que, independentemente das condições iniciais, com uma boa escolha dos parâmetros de penalidade proporcionam estabilidade (convergência) global. A primeira proposta apresenta como função objetivo o quadrado das diferencias entre as taxas de fluxo e a segunda proposta refere-se à minimização do valor absoluto da diferença entre as taxas de fluxo entre usuários.

Ao considerar a diferença entre as taxa de fluxo, a dinâmica considera a equidade entre os usuários. Isto apresenta uma vantagem significativa na convergência à equidade eficiente, mas se tem que pagar o preço de transmitir as taxas de fluxo entre os usuários, fato que pode adicionar tráfego excessivo numa rede grande como a Internet. Considerando este contexto, um novo protocolo que utilize algoritmos baseados em otimização podem ser implementados em nós de fronteira de forma tal que, sirva para um conjunto de nós que compartilham um canal dado de aceso à Internet.

3.2.1 Algoritmo de Otimização Quadrática

Em termos matemáticos, a equidade na alocação de recursos pode ser descrita como:

$$x_i = x_j , \tag{3.4}$$

Enquanto que, a utilização completa da capacidade do gargalo do canal (ou capacidade da rede) pode ser expressa como:

$$\sum_{i=1}^{n} x_i = c , \tag{3.5}$$

Para simplificar, se considera um sistema com duas fontes (usuários) que compartilham um único canal, que pode administrar c pacotes por segundo. A taxa, à qual os usuários injetam pacotes x_i, não deve sobrepassar a capacidade c do canal e deve existir uma justa distribuição das mesas de tal forma que os dois usuários possam ter a mesma capacidade de envio de seus dados.

Para obter equidade na transmissão de fluxos, entre dois usuários, seria necessário que a diferença entre ambas as taxas seja a mínima possível. Então, uma proposta é minimizar o quadrado da diferença entre as taxas de fluxo de ambos os usuários.

$$\min f(x) = \frac{1}{2}(x_1 - x_2)^2 = \frac{1}{2}x^T Q x, \qquad \text{onde } Q = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{3.6}$$

Obviamente o mínimo será no caso em que a diferença entre x_1 e x_2 seja igual ao zero, isto é a restrição de equidade, seja satisfeita. Todavia, deve se atender à segunda restrição, capacidade total do canal ($x_1 + x_2 - c = 0$). Deste modo, (3.6) se reescreve como o seguinte problema de otimização com restrições:

$$\begin{aligned} &\min f(x) = \frac{1}{2}(x_1 - x_2)^2 \\ &\text{sujeito a:} \quad x_1 + x_2 - c = 0 \\ &\qquad\qquad x_i \geq 0 \end{aligned} \tag{3.7}$$

Graficamente o problema (3.7) pode ser visto na Figura 3.6.

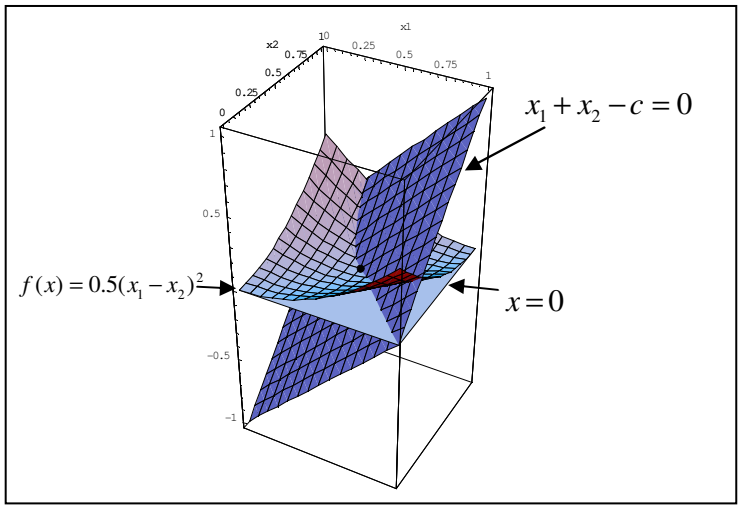


Figura 3.6: $f(x) = 0.5(x_1 - x_2)^2$ com restrições.

Este problema de minimização é resolvido usando o enfoque de função de penalidade combinado com a descida do gradiente. Por conseguinte, uma função de penalidade a qual será referida como uma função de energia computacional é definida como segue:

$$E(x, \alpha_0, \alpha_1, \alpha_3) = \alpha_0 \frac{1}{2}x^T Q x + \alpha_1 \left\| A_1 x - b_1 \right\|_1 - \alpha_3 \sum_{i=1}^{m_3=2} \min\{0, x_i\} \tag{3.8}$$

onde $Q = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$, $A_1 x - b_1 = 0$, $A_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}$, e $b_1 = c$.

Assim, um sistema dinâmico de gradiente que minimiza a função de energia computacional $E(\cdot)$ pode ser escrito como:

$$
\begin{aligned}
\dot{x} &= -\nabla E(x, \alpha_1, \alpha_2, \alpha_3) \\
&= -[\alpha_0 \, Q \, x + \alpha_1 \, A_1^T \, \mathrm{sgn}(A_1 x - c) - \alpha_3 \, (-h \, \mathrm{sgn}(x))] \\
&= -\alpha_0 \begin{bmatrix} x_1 - x_2 \\ -x_1 + x_2 \end{bmatrix} - \alpha_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \mathrm{sgn}(x_1 + x_2 - c) - \alpha_3 h \, \mathrm{sgn}(x)
\end{aligned}
\tag{3.9}
$$

sendo $x = [x_1, x_2]^T$, $A_1 = [1, 1]$.

Ou

$$
\begin{aligned}
\dot{x}_1 &= \alpha_0(-x_1 + x_2) - \alpha_1 \, \mathrm{sgn}(x_1 + x_2 - c) - \alpha_3 \, h \mathrm{sgn}(x_1) \\
\dot{x}_2 &= \alpha_0(x_1 - x_2) - \alpha_1 \, \mathrm{sgn}(x_1 + x_2 - c) - \alpha_3 \, h \mathrm{sgn}(x_2)
\end{aligned}
\tag{3.10}
$$

A equação (3.9) representa a dinâmica do primeiro modelo proposto que de agora em diante será chamado de Otimização Quadrática (**OptQua**) e cuja estrutura de controle corresponde ao diagrama da Figura 3.7.

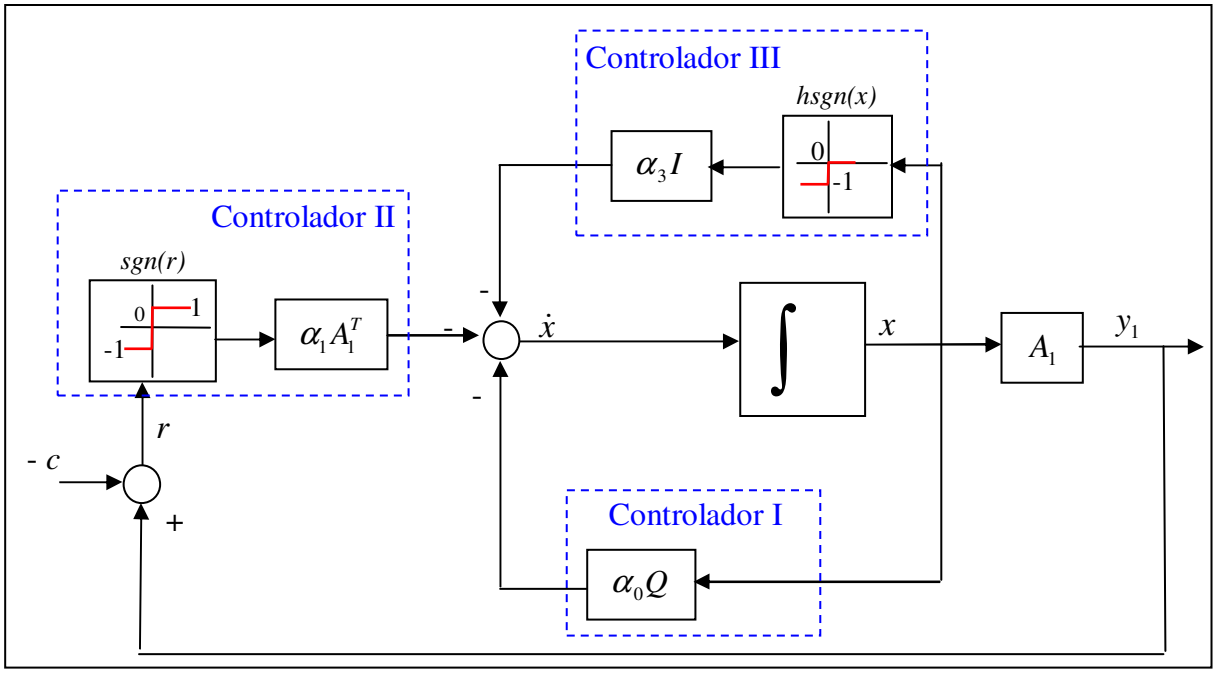


Figura 3.7: Estrutura do sistema de controle da proposta Otimização Quadrática.

A dinâmica da equação (3.9) mostra que além da informação de realimentação binária, cada usuário deve ter a informação das taxas de envio dos vizinhos. Uma situação real, na qual esta informação estaria disponível é um cluster de usuários que utiliza um servidor para conectar-se a Internet através de um único canal de largura de banda c. O problema de assegurar o acesso equitativo dos usuários ou nós do cluster ao canal (também denominado canal gargalo) é exatamente o problema que se está modelando.

A Figura 3.8 mostra o campo de vetores do algoritmo proposto e a Figura 3.9 mostra as trajetórias que convergem ao ponto de equidade e eficiência ótima, isto é, a convergência a um estado justo do equilíbrio.

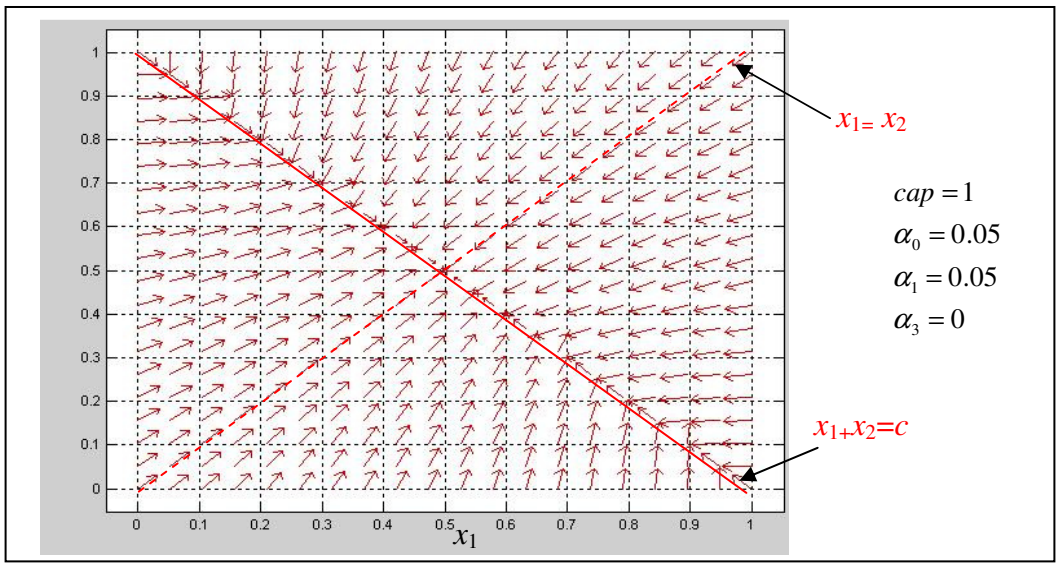


Figura 3.8: Otimização Quadrática - Campo de vetores

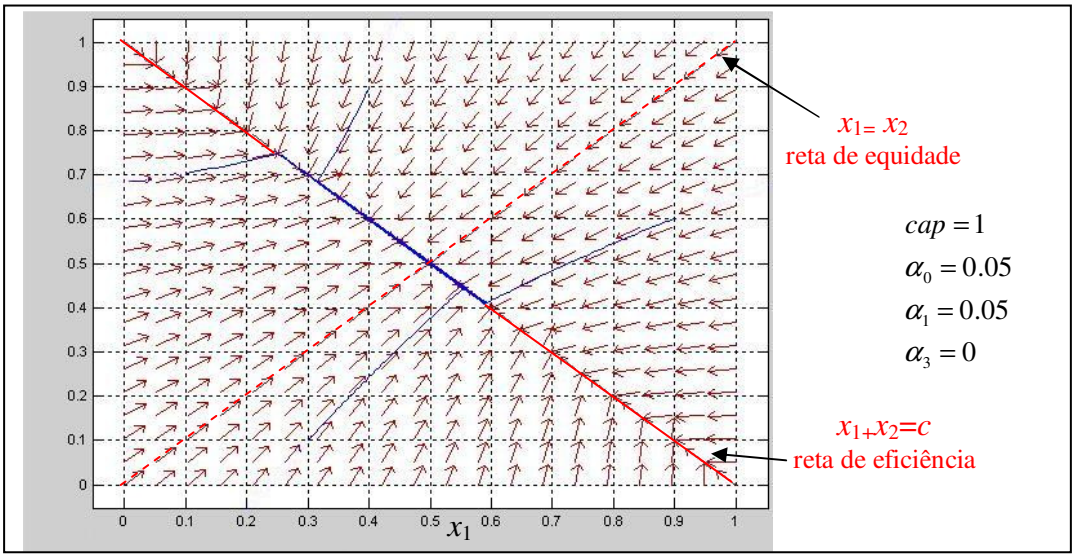


Figura 3.9: Plano de fase do sistema dinâmico proposto mostrando convergência de trajetórias ao ponto de equidade e eficiência ótima.

Assim, OptQua em sua versão discreta com atrasos homogêneos (síncrono) estaria representado pela equação:

$$\begin{aligned} x_1(k) &= x_1(k-1) + \delta\alpha_0(-x_1(k-1) + x_2(k-1)) \\ &\quad - \delta\alpha_1 \operatorname{sgn}(x_1(k-1) + x_2(k-1) - c) - \delta\alpha_3 \, h \operatorname{sgn}(x_1(k-1)) \\ x_2(k) &= x_2(k-1) + \delta\alpha_0(x_1(k-1) - x_2(k-1)) \\ &\quad - \delta\alpha_1 \operatorname{sgn}(x_1(k-1) + x_2(k-1) - c) - \delta\alpha_3 \, h \operatorname{sgn}(x_2(k-1)) \end{aligned} \tag{3.11}$$

Na equação (3.11) o segundo termo avalia que a equidade seja satisfeita, considerando a diferença das taxas de fluxo entre os dois usuários. O terceiro termo mostra que o usuário incrementa (ou decrementa respectivamente) sua taxa quando a capacidade do canal não é excedida (ou é excedida respectivamente).

A versão da equação (3.11) para atrasos heterogêneos (assíncrono) pode ser escrita como:

$$\begin{aligned} x_1(k) &= x_1(k-d_1^f-d_1^b)+\delta\alpha_0(-x_1(k-d_1^f-d_1^b)+x_2(k-d_2^f-d_2^b)) \\ &\quad -\delta\alpha_1\left[\operatorname{sgn}(x_1(k-d_1^f)+x_2(k-d_2^f)-c)\right]_{d_1^b}-\delta\alpha_3\, h\operatorname{sgn}(x_1(k-d_1^f-d_1^b)) \\ x_2(k) &= x_2(k-d_2^f-d_2^b)+\delta\alpha_0(x_1(k-d_1^f-d_1^b)-x_2(k-d_2^f-d_2^b)) \\ &\quad -\delta\alpha_1\left[\operatorname{sgn}(x_1(k-d_1^f)+x_2(k-d_2^f)-c)\right]_{d_2^b}-\delta\alpha_3\, h\operatorname{sgn}(x_2(k-d_2^f-d_2^b)) \end{aligned} \tag{3.12}$$

Formulação para *n* usuários e um canal gargalo.

Seja $x=[x_1,...,x_n]^T$ o vetor das taxas dos usuários, $A_1=[1,1,...,1]$; e Q a matriz dada por:

$$Q=\begin{bmatrix} n-1 & -1 & \cdots & -1 \\ -1 & n-1 & \cdots & -1 \\ -1 & \vdots & \ddots & -1 \\ -1 & 1 & \cdots & n-1 \end{bmatrix}$$

Podemos formular o problema de otimização para *n* usuários compartilhando um mesmo canal gargalo de banda (capacidade) *cap* da seguinte maneira.

$$\min f(x)=\frac{1}{2}x^TQx \text{ tal que } \left\|A_1x-c\right\|_1=0, x_i\geq 0, \forall i \tag{3.13}$$

Introduzindo a função de energia computacional

$$E(x,\alpha_0,\alpha_1,\alpha_3)=\alpha_0\frac{1}{2}x^TQx+\alpha_1\left\|A_1x-b_1\right\|_1-\alpha_3\sum_{i=1}^{m_3=2}\min\{0,x_i\} \tag{3.14}$$

tem-se o sistema gradiente, cujas trajetórias minimizam a função $E(\bullet)$.

$$\dot{x}=-\nabla E(x,\alpha_1,\alpha_2,\alpha_3) \;=-[\alpha_0\, Q\, x+\alpha_1\, A_1^T\operatorname{sgn}(A_1x-c)-\alpha_3\,(-h\operatorname{sgn}(x))] \tag{3.15}$$

Assim, a dinâmica para o usuário *i* será:

$$\dot{x}_i=-\alpha_0((n-1)x_i-\sum_{i\neq j}^{n}x_j)-\alpha_1\operatorname{sgn}(\sum_{i=1}^{n}x_i-c)-\alpha_3\, h\operatorname{sgn}(x_i) \tag{3.16}$$

3.2.2 Algoritmo de Otimização Absoluta

Outra proposta simples, exemplificada para dois usuários, refere-se à minimização do valor absoluto da diferença das taxas de fluxo de entre dois usuários, de tal forma que, os fluxos evoluam a um estado estável, no qual o canal seja utilizado equitativamente, ou seja, minimize a função

$$f(x) = |x_1 - x_2| \tag{3.17}$$

Obviamente, o mínimo é atingido quando a diferença entre x_1 e x_2 seja igual a zero. Considerando que, se deve satisfazer a restrição da capacidade total do canal ($x_1 + x_2 - c = 0$), pode-se escrever o seguinte problema de otimização com restrições:

$$\begin{aligned} &\min |x_1 - x_2| \\ &\text{sujeito a:} \quad x_1 + x_2 - c = 0 \\ &\qquad\qquad x_i > 0 \end{aligned} \tag{3.18}$$

Graficamente o problema pode ser representado pela Figura 3.10.

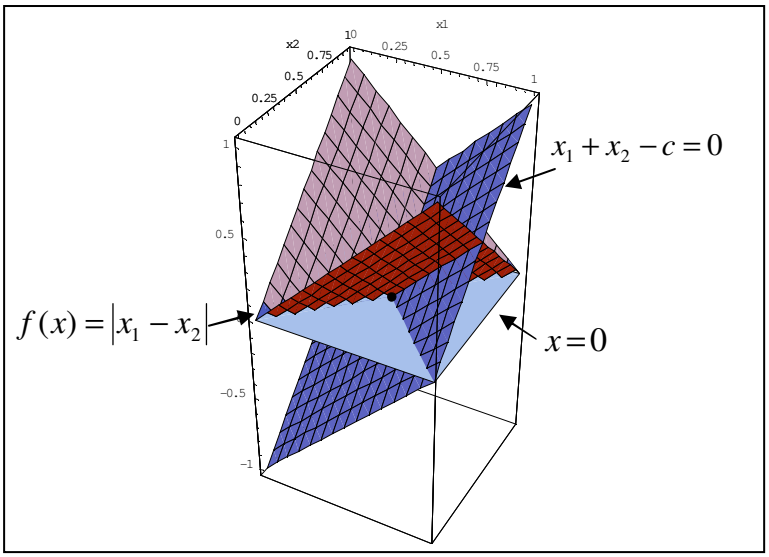


Figura 3.10: $f(x) = |x_1 - x_2|$ com restrições

Do mesmo modo que, a proposta anterior, este problema de minimização pode ser resolvido usando o enfoque de função de penalidade, combinado com um método gradiente simples. A função penalizada é logo interpretada como uma função de energia computacional, e definida como se segue:

$$E(x, \alpha_0, \alpha_1, \alpha_3) = \alpha_0 |x_1 - x_2| + \alpha_1 \|A_1 x - b_1\|_1 - \alpha_3 \sum_{i=1}^{m_3=2} \min\{0, x_i\} \tag{3.19}$$

sendo, $A_1 x - b_1 = 0$, $A_1 = [1 \quad 1]$, $b_1 = c$ e $\alpha_i > 0, \forall i$

O sistema dinâmico de gradiente descendente, que minimiza a função de energia computacional $E(\cdot)$ pode ser escrito como segue (KASZKUREWICZ e BHAYA, 2006).

$$\begin{aligned}\dot{x} &= -[\alpha_0 A_0^T \operatorname{sgn}(A_0 x - b_0) + \alpha_1 A_1^T \operatorname{sgn}(A_1 x - c) - \alpha_3 (-h \operatorname{sgn}(x))] \\ &= -\alpha_0 \begin{bmatrix} 1 \\ -1 \end{bmatrix} \operatorname{sgn}(x_1 - x_2) - \alpha_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \operatorname{sgn}(x_1 + x_2 - c) - \alpha_3 h \operatorname{sgn}(x)\end{aligned} \tag{3.20}$$

sendo $x = [x_1, x_2]^T$, $A_0 = [1, -1]$.

Ou

$$\begin{aligned}\dot{x}_1 &= -\alpha_0 \operatorname{sgn}(x_1 - x_2) - \alpha_1 \operatorname{sgn}(x_1 + x_2 - cap) - \alpha_3 h \operatorname{sgn}(x_1) \\ \dot{x}_2 &= \alpha_0 \operatorname{sgn}(x_1 - x_2) - \alpha_1 \operatorname{sgn}(x_1 + x_2 - cap) - \alpha_3 h \operatorname{sgn}(x_2)\end{aligned} \tag{3.21}$$

A equação (3.20) representa a dinâmica do modelo proposto, que doravante será chamado Otimização Absoluta (**OptAbs**) e cuja estrutura de controle corresponde ao diagrama da Figura 3.11.

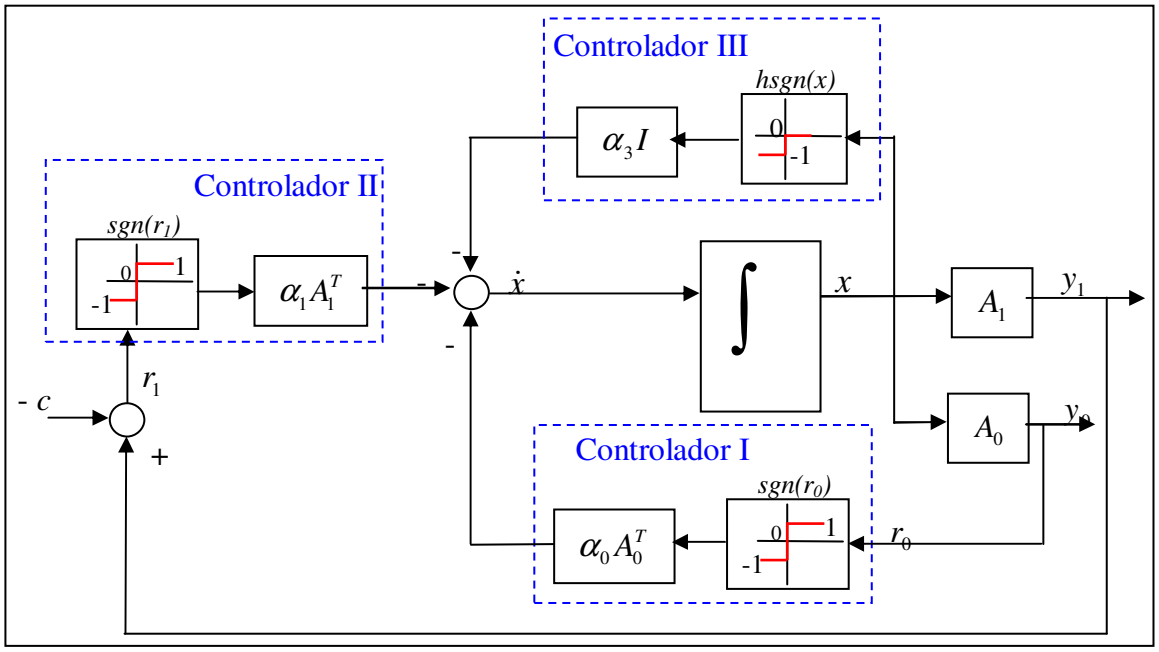


Figura 3.11: Estrutura do sistema de controle da proposta Otimização Absoluta.

Cada usuário deve dispor da informação das taxas de envio dos vizinhos, além da informação de realimentação binária. É a mesma aplicação prática que a proposta anterior. A informação seria disponível através do cluster ligado ao servidor, para conectar se a Internet com largura de banda c. A equação (3.20) administra assim o acesso equitativo dos nós do cluster ao canal gargalo.

Fazendo um diagrama de vetores do algoritmo proposto, para o caso de dois usuários (Figura 3.12), pode se verificar que eles convergem ao ponto de equidade ótimo, isto é, a convergência a um estado justo do equilíbrio. A Figura 3.13, que mostra as trajetórias de diversas condições iniciais, converge realmente à equidade e eficiência ótima.

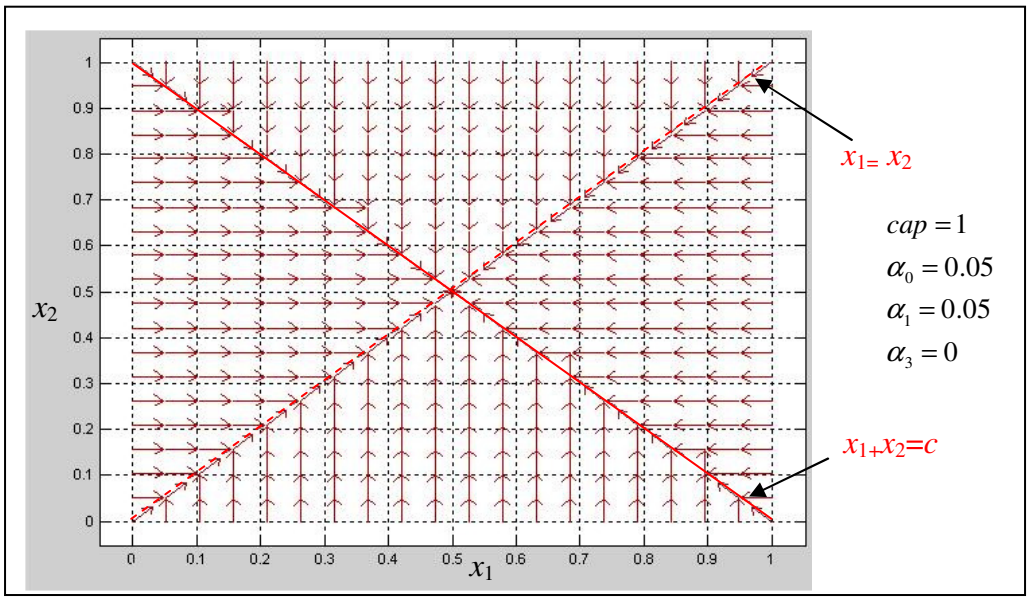


Figura 3.12: Otimização Absoluta - Campo de vetores.

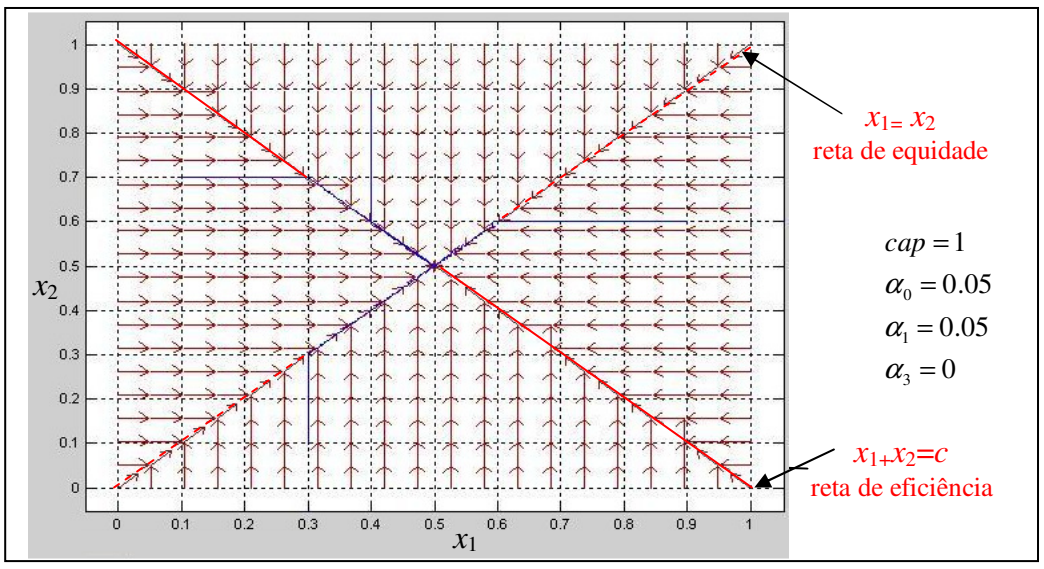


Figura 3.13: Plano de fase do sistema dinâmico proposto mostrando convergência de trajetórias ao ponto de equidade e eficiência ótima.

A proposta OptAbs discreta, com atrasos homogêneos e síncronos, pode ser representado pelas equações.

$$
\begin{aligned}
x_1(k) &= x_1(k-1) - \delta\,\alpha_0\,\mathrm{sgn}(x_1(k-1) - x_2(k-1)) \\
&\quad - \delta\,\alpha_1\,\mathrm{sgn}(x_1(k-1) + x_2(k-1) - c) - \delta\,\alpha_3\,h\,\mathrm{sgn}(x_1(k-1))) \\
x_2(k) &= x_2(k-1) + \delta\,\alpha_0\,\mathrm{sgn}(x_1(k-1) - x_2(k-1)) \\
&\quad - \delta\,\alpha_1\,\mathrm{sgn}(x_1(k-1) + x_2(k-1) - c) - \delta\,\alpha_3\,h\,\mathrm{sgn}(x_2(k-1)))
\end{aligned}
\tag{3.22}
$$

Note que o segundo termo da equação (3.22) avalia a diferença das taxas entre dois usuários, isto é; verifica se a equidade é satisfeita. O terceiro termo mostra que o usuário incrementa (ou decrementa) sua taxa quando a capacidade do canal não é excedida (excedida respectivamente).

A equação (3.22) pode ser reescrita em termos de atrasos heterogêneos (assíncrono).

$$
\begin{aligned}
x_1(k) &= x_1(k - d_1^f - d_1^b) - \delta\alpha_0\,\mathrm{sgn}(x_1(k - d_1^f - d_1^b) - x_2(k - d_2^f - d_2^b)) \\
&\quad - \delta\alpha_1\left[\mathrm{sgn}(x_1(k - d_1^f) + x_2(k - d_2^f) - c)\right]_{d_1^b} \\
&\quad - \delta\alpha_3\,h\,\mathrm{sgn}(x_1(k - d_1^f - d_1^b)) \\
x_2(k) &= x_2(k - d_2^f - d_2^b) + \delta\alpha_0\,\mathrm{sgn}(x_1(k - d_1^f - d_1^b) - x_2(k - d_2^f - d_2^b)) \\
&\quad - \delta\alpha_1\left[\mathrm{sgn}(x_1(k - d_1^f) + x_2(k - d_2^f) - c)\right]_{d_2^b} \\
&\quad - \delta\alpha_3\,h\,\mathrm{sgn}(x_2(k - d_2^f - d_2^b))
\end{aligned}
\tag{3.23}
$$

Formulação para n usuários de um canal gargalo.

Seja $x = [x_1,...,x_n]^T$ o vetor das taxas dos usuários, $A_1 = [1,1,...,1]$; e Q a matriz dada por

$$
Q = \begin{bmatrix}
1 & -1 & 0 & 0 & \cdots & 0 & 0 \\
1 & 0 & -1 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
1 & 0 & 0 & 0 & \cdots & 0 & -1 \\
0 & 1 & -1 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 1 & -1
\end{bmatrix}
\quad \text{de dimensão } \binom{n}{2} \text{x } n
\tag{3.24}
$$

Dessa forma, podemos formular o problema de otimização para *n* usuários compartilhando um mesmo canal gargalo de banda (capacidade) *cap* da seguinte maneira.

$$
\min\left\|Qx\right\|_1 \text{ tal que } \left\|A_1 x - c\right\|_1 = 0, x_i \geq 0, \forall i
\tag{3.25}
$$

Introduzindo a função de energia computacional

$$E(x,\alpha_0,\alpha_1,\alpha_3) = \alpha_0 \left\|Q\,x\right\|_1 + \alpha_1 \left\|A_1 x - b_1\right\|_1 - \alpha_3 \sum_{i=1}^{m_3=2} \min\{0, x_i\} \tag{3.26}$$

tem-se o sistema gradiente, cujas trajetórias minimizam a função $E(\bullet)$.

$$\begin{aligned} \dot{x} &= -\nabla E(x,\alpha_1,\alpha_2,\alpha_{31}) \\ &= -\left[\alpha_0\, \nabla\left\|\mathrm{Qx}\right\|_1 + \alpha_1 \mathrm{A}_1^{\mathrm{T}}\, \mathrm{sgn}(A_1 x - c) - \alpha_3(-h\,\mathrm{sgn}(x))\right] \end{aligned} \tag{3.27}$$

O primeiro termo a direita pode ainda ser simplificado como se segue:

$$\alpha_0\, \nabla\left\|\mathrm{Qx}\right\|_1 = \alpha_0 \sum_{j\neq i, j<i}^{n} \mathrm{sgn}(x_j - x_i) - \alpha_0 \sum_{k\neq i, k<i}^{n} \mathrm{sgn}(x_i - x_k) \tag{3.28}$$

Ambos os algoritmos propostos, baseados em otimização, apresentados neste capítulo, podem ser implementados em protocolos de controle de congestionamento e ser utilizados em cluster de usuários, que utilizam um servidor para ter acesso a Internet. Assim, se teria um protocolo de controle de congestionamento em dois níveis, um protocolo de controle de congestionamento baseado em otimização com OptQua ou OptAbs e um segundo nível com um protocolo tradicional de controle de congestionamento tipo TCP New Reno, para trafego entre servidor e Internet.

Síntese do capitulo

Este capítulo apresentou quatro novas propostas de algoritmos para controle de congestionamento. Duas propostas inspiradas no tradicional AIMD, o Algoritmo Westwood Equilíbrios Virtuais (Alg_WVE) e o Algoritmo Equilíbrios Virtuais (ALG_VE), ajustam dinamicamente os parâmetros das dinâmicas de incremento e decremento segundo configuração da rede e/ou informação da mesma se obteria melhores resultados que usando parâmetros estáticos.

Outras duas propostas, o Algoritmo Otimização Quadrática (Alg_OptQua) e o Algoritmo Otimização Absoluta (Alg_Abs) consideram ao problema de controle de congestionamento como um problema de otimização maximizando uma função objetivo segundo restrições de limitação da capacidade da rede. Estas duas propostas demonstram que baseados num sistema de controle, é possível modelar e resolver o problema de controle de congestionamento (como poderá verse experimentalmente no capitulo 5.

No próximo capítulo, a implementação dos algoritmos Alg_WVE e Alg_VE em protocolos será mostrada.

4. Protocolos de controle de congestionamento

A Internet é utilizada por um grande número de usuários espalhados pelo mundo concorrendo por recursos da rede, com uma demanda de comunicação altamente variante. Quando a demanda dos usuários excede a capacidade disponível da rede se tem o congestionamento e os usuários finais devem reagir ao mesmo. Assim, protocolos de controle de congestionamento devem ajustar a taxas de transmissão dos usuários para evitar o colapso de congestionamento e manter uma alta utilização da rede (FLOYD e FALL, 1999).

Este capítulo propõe novos protocolos de controle de congestionamento que utilizam uma filosofia semelhante ao TCP tradicional, mas com uma adaptação dinâmica de parâmetros, para reagir à demanda de comunicação da Internet de forma eficaz.

Uma diferença fundamental entre os protocolos tipo TCPs (Tahoe, Reno, New Reno, Westwood) é as novas propostas, TCP_Westwood Equilíbrios Virtuais e TCP_Equilíbrios Virtuais, é que estes últimos utilizam uma função de incremento e uma função de decremento dinamicamente adaptativa. Outra diferença fundamental entre é que os protocolos propostos neste trabalho não apresentam as duas fases tradicionais de Partida Lenta e Prevenção do Congestionamento; pelo contrario, utilizam uma dinâmica de incremento e uma dinâmica de decremento dependendo se existe ou não perda de pacotes.

4.1 Protocolos TCPs

O principal objetivo da camada de transporte é oferecer um serviço confiável, eficiente e econômico aos usuários da camada de aplicação (TANENBAUM, 1996). Para alcançar este objetivo, a camada de transporte faz uso do serviço fornecido pela camada de rede. A Internet tem dois principais protocolos na camada de transporte, um protocolo orientado a conexão e um sem conexão. O protocolo orientado a conexão é o TCP ou Protocolo de Controle de Transmissão e o protocolo sem conexão é o UDP ou Protocolo de Dados de Usuário (acrônimo em inglês de *User Data Protocol*).

TCP foi formalmente definido na RFC 793 (POSTEL, 1981) e foi projetado especificamente para oferecer um fluxo de bytes fim-a-fim confiável em uma inter-rede não confiável. Uma inter-rede é diferente de uma única rede, porque suas muitas partes

podem ter topologias, largura de banda, atrasos, tamanhos de pacote e outros parâmetros completamente diferentes. O TCP foi projetado para se adaptar dinamicamente às propriedades da inter-rede e para ser robusto diante dos muitos tipos de falhas que podem ocorrer.

O TCP recebe um fluxo de bytes da camada de aplicação, os divide em pequenos segmentos e envia cada segmento ao TCP receptor (chamado comumente de TCP *sink*). O remetente atribui um número de sequência e comprimento (isto é o número dos bytes) a cada segmento. Para cada segmento recebido corretamente, o receptor emite um reconhecimento ou ACK (acrônimo do inglês *acknowledment*), que inclui o número de sequência do seguinte segmento esperado da sequência. Os bytes reconhecidos são cumulativos (isto é, reconhece todos os bytes recebidos em ordem). Com cada TCP do reconhecimento o receptor anuncia um *buffer*/janela e o remetente assegura-se para não ter emitir (ou para ter na rede) mais bytes do que esta janela permite. Este processo é chamado, geralmente, o controle de fluxo (BERTSEKAS e GALLAGER, 1992).

O mecanismo do controle de fluxo impede que o remetente transmita mais bytes do que o receptor pode consumir ou armazenar no *buffer*. Consequentemente, o controle de fluxo é um mecanismo que impede a perda do pacote de um receptor lento. Quando o TCP foi desdobrado extensamente (porque as maiorias das aplicações o usaram), encontrou-se que a perda do pacote, também, pode acontecer por gargalo da rede (isto é roteadores) devido ao congestionamento. Era necessário adicionar ao TCP, além do controle de fluxo, mecanismos do controle de congestionamento.

O primeiro TCP com mecanismo do controle de congestionamento foi TCP Tahoe (LAHAMAS e TSAUSSIRIS, 2002) e foi seguido, então, por outros TCPs (Reno, Vegas, NewReno, Sack) que são modificações ou melhorias de TCP Tahoe que são as tradicionais protocolos estudados.

Existem, também, outras variações do protocolo TCP como o TCP SACK (MATHIS et.al., 1996), o HSTCP (FLOYD, 2008), o STCP (KELLY, T., 2003), o FAST TCP (WEI et.al., 2006), o TCP Jersey (KU et.al., 2004), o TCP BaLDE (LE, 2006), entre outros; mas no contexto deste trabalho o TCP New Reno (ou TCP_NR) será utilizado como protocolo, com o qual comparar as novas propostas, já que ele é o utilizado atualmente.

4.1.1 TCP Tahoe

TCP Tahoe estende a versão antiga de TCP (JACOBSON 1981 e POSTEL, 1981) com quatro novos mecanismos: Partida Lenta, ACK *clocking*, ajuste dinâmico de janela, e Retransmissão Rápida (JACOBSON, 1988). Além disso, melhora o mecanismo de *Timeout* de TCP para beneficiar o processo de evitação do congestionamento (LAHAMAS, 2003).

A quantidade de dados que o remetente pode ter em transito (isto é não reconhecidos) é controlado por um novo parâmetro chamado janela de congestionamento ou *cwnd* . O remetente não pode enviar na rede mais bytes que o mínimo da janela de congestionamento e a janela de anúncio. A janela do congestionamento é o parâmetro que avalia a justo compartilhamento ou a capacidade da rede, e a janela de anúncio é o parâmetro que determina a janela do receptor. O compartilhamento justo da rede e o *buffer* do receptor são parâmetros dinâmicos e eles podem mudar com o tempo. O parâmetro da janela de anúncio é atualizado quando o remetente recebe um ACK.

Existe um mecanismo que atualiza o parâmetro da janela de congestionamento. Este parâmetro é chamado em TCP Tahoe ajuste dinâmico da janela (LAHAMAS, 2003, PAGANINI, 2002) e é similar ao mecanismo AIMD. O mecanismo prova continuamente a rede para uma largura de banda disponível e reduz a janela de congestionamento, quando a largura de banda não esta disponível. Para cada transmissão de janela bem sucedida, TCP assume que a largura de banda esta disponível e incrementa a janela de congestionamento, aproximadamente, em um segmento (por RTT). Quanto um pacote é perdido, deduzido devido ao congestionamento, o receptor responde a cada subsequente chegada do pacote com o envio de um ACK (ou reconhecimento) ao seguinte segmento em sequência, isto é, o pacote perdido. Isto causará a chegada de ACKs com o mesmo número de sequência ou ACKs duplicados (DACKs). A chegada de DACKs (tipicamente três) é considerada como congestionamento e o TCP envia imediatamente o pacote perdido, isto é, aquele indicado por o DACKs e reduz a sua janela. Esta ação é chamada Retransmissão Rápida.

Ao começar uma nova conexão a fase Partida Lenta é ativada. Neste caso, o TCP não sabe qual é o nível do congestionamento e o *Additive Increase* demorará em estimá-lo. Em vez de examinar a rede com A*dditive Increase* TCP Tahoe tem uma fase, a Partida Lenta, que examina a rede mais rapidamente. A Partida Lenta abre a janela do

congestionamento em forma exponencial. Para cada ACK recebido, aumenta sua janela por um segmento; assim, dobrá-lo a cada RTT.

Quando a congestionamento acontece, um patamar é ajustado à metade da janela do congestionamento, que então, é ajustada a um segmento. O TCP Tahoe tentará a rede com Partida Lenta até que a janela do congestionamento seja igual o parâmetro Limiar da Partida Lenta. Quando a janela alcança este patamar, se entra na fase de Prevenção de Congestionamento onde o TCP Tahoe provará a rede com *Additive Increase* fazendo $cwnd(k+1) = cwnd(k) + \frac{1}{cwnd(k)}$ por cada ACK recebido com êxito e $cwnd(k+1) = 1$ quando acontece uma perda.

ACK *clocking* é um mecanismo que melhora a utilização do canal e prolonga o ciclo do congestionamento (ou o período da perda do pacote). Pretende-se espalhar os segmentos do TCP sobre o canal de modo que se acumulem pacotes na fila dos roteadores somente quando a tubulação do canal é utilizada inteiramente.

O *Timeout* ou intervalo de parada é um mecanismo que retransmite os dados quando a retroalimentação não chega ao receptor. Em redes com fio, o congestionamento severo (por exemplo, muitos fluxos no sistema) é a razão principal para a ausência de ACKs (e perda dos dados). *Timeout* reforça uma estratégia de retransmissão conservadora no TCP. Para cada transmissão bem sucedida da janela, o TCP mede o tempo de percurso e grava uma estimativa dos RTTs (isto é um RTT ponderado de todos os RTTs medidos) da conexão. Cada vez que o RTT estimado é atualizado, o TCP fixa o valor do *Timeout* ao valor duplo do RTT estimado. Cada vez que uma janela nova é transmitida o TCP começa um temporizador, que expire quando a janela inteira é reconhecida ou após período de *Timeout*. Quando o temporizador expira, o valor do *Timeout* é dobrado. Isto significa que, na ausência de ACKs não tomará mais de um RTT real ou retardo de propagação para começar a retransmissão.

Basicamente, todas as versões de TCP diferem na política de redução, depois da retransmissão rápida. TCP Tahoe, conhecido como o mais conservativo dos protocolos, reduz sua janela em um segmento; visto que, as outras versões do TCP o reduzem à metade.

4.1.2 TCP Reno

O TCP Reno é uma versão melhorada do TCP Tahoe que introduze a fase de Recuperação Rápida que faz o TCP recuperar se de forma mais eficiente de uma perda

(PAGANINI, 2002). Assim, o tempo desde a detecção da perda (através dos DACKs) até a recepção do ACK do pacote retransmitido é chamado de fase Retransmissão Rápida/Recuperação Rápida. Em TCP Tahoe, o tamanho da janela é fixo na fase Retransmissão Rápida/Recuperação Rápida. Isto significa que um novo pacote pode ser retransmitido, somente um RTT depois. Ademais, o canal desde a fonte até o destino é limpado quando o pacote retransmitido chega ao receptor, e alguns dos roteadores do caminho ficam sem ser utilizados, o que conduz a uma perda de eficiência.

A Retransmissão Rápida permite ao remetente TCP Reno incrementar temporariamente sua janela em um pacote ao receber cada DACK, em quanto esta na fase Retransmissão Rápida/Recuperação Rápida. O raciocínio é que cada DACK indica que o pacote há deixado a rede. Quando a janela torna-se maior que o número de pacotes saídos, um novo pacote pode ser retransmitido na fase Retransmissão Rápida/Recuperação Rápida, enquanto esta aguardando um ACK (não duplicado) para o pacote retransmitido.

A Recuperação Rápida ajusta a janela ao final da fase Retransmissão Rápida/Recuperação Rápida à metade (LAHAMAS, 2003, e PAGANINI, 2002) do valor ao inicio da fase e passa diretamente à fase de Prevenção de Congestionamento fazendo $cwnd(k+1) = 0.5\ cwnd(k)$. Por isso, a fase de Partida Lenta é executada raramente em TCP Reno, unicamente quando o congestionamento primeiro inicia e quando a perda é detectada por *timeout* em vez de por DACKs. TCP Reno implementa o algoritmo algoritmos AIMD com os parâmetros $a_I = 1$ e $b_D = 0.5$.

4.1.3 TCP Vegas

O TCP Vegas (PAGANINI, 2002) melhora o TCP Reno a través de três técnicas: A primeira é um novo mecanismo de retransmissão, onde o *Timeout* é verificado na recepção do primeiro DACK, em vez de aguardar ao terceiro DACK (como no caso de TCP Reno), e resulta em uma detecção mais oportuna da perda. A segunda técnica é uma forma mais prudente de crescimento do tamanho da janela durante o uso inicial da Partida Lenta, quando a conexão se inicia, e resulta em menos perdas. A terceira técnica é um novo mecanismo de prevenção de congestionamento, que corrige o comportamento oscilatório de TCP Reno. A idéia é que a fonte tenha um número estimativo de seus próprios pacotes guardados no trajeto e manter esse número entre α (tipicamente 1) e β (tipicamente 3) ajustando o tamanho de sua janela. O tamanho da

janela é incrementado linearmente no seguinte RTT caso a estimativa atual seja menor que α ou maior que β. De outra forma, o tamanho da janela é inalterado.

Em resumo, a dinâmica do TCP Vegas (LIAN et.al., 2007) para incremento é:

$$cwnd(k+1) = \begin{cases} cwnd(k)+1 & \text{quando } d(k) < \alpha \\ cwnd(k)-1 & \text{quando } d(k) > \beta \\ cwnd(k) & \text{em caso contrario} \end{cases} \quad \text{onde} \quad d(k) = \left[\frac{cwnd(k)}{RRT_{\min}} - \frac{cwnd(k)}{RTT(k)} \right] RTT_{\min}$$

e a dinâmica de decremento, aplicada quando acontece uma perda, é:

$$cwnd(k+1) = 0.5\, cwnd(k)$$

Sendo $cwnd(k)$ é o tamanho da janela de congestionamento do remetente TCP (correspondente ao fluxo x_i) no tempo k, $d[k]$ é uma medida da diferencia da taxa esperada e da taxa real, $(RTT(k), RTT_{\min})$ são o RTT real e o mínimo, e (α, β) são constantes de nível usadas para especificar a condição desejada de conexão de rede.

4.1.4 TCP New Reno

Uns dos protocolos TCP mais utilizados atualmente é o protocolo TCP New Reno. O TCP New Reno é um algoritmo de congestionamento desenvolvido por FLOYD e HENDERSON (1999), que evita múltiplas reduções da janela de congestionamento, quando vários segmentos de uma janela de dados são perdidos. Assim, o TCP New Reno foi implementado com o objetivo de otimizar o TCP Reno no caso de múltiplas perdas de pacotes, em um única janela de congestionamento. A fase de Retransmissão Rápida é a mesma que o Reno, a diferença é que New Reno admite múltiplas retransmissões.

Quando acontece a perda de um pacote, através do recebimento de reconhecimentos duplicados, um ACK de número diferente só será enviado quando o pacote retransmitido chegar ao destino. Se ocorrer uma única perda, esse reconhecimento confirma a recepção de todos os segmentos transmitidos, até antes da execução da fase de Retransmissão Rápida. Porém, quando existem múltiplas perdas de pacotes, e um reconhecimento chega, este somente confirma alguns deles. Por isso, este reconhecimento é denominado parcial ou ACK parcial (FLOYD e HENDERSON, 1999).

No TCP Reno, na chegada de ACKs parciais o tamanho da janela de transmissão de dados é reduzido ao parâmetro *ssthresh*, a fase de Recuperação Rápida é

interrompida, e se inicia a fase de Prevenção do Congestionamento. Este processo é repetido cada vez que chega um novo grupo de ACKs parciais fazendo com que o TCP reduza a janela de transmissão pela metade, seguidas vezes. Enquanto, ao receber reconhecimentos parciais, o TCP New Reno fica na fase Retransmissão Rápida evitando as múltiplas reduções no valor da janela de congestionamento. Assim, cada reconhecimento parcial é visto como uma indicação de que mais de um pacote foi perdido e deve ser retransmitido. Deste modo, quando vários pacotes são perdidos na mesma janela de dados, o TCP New Reno é capaz de evitar o *timeout*. Para isto, ele retransmite um pacote perdido por RTT até que todos os pacotes perdidos desta janela tenham sido retransmitidos. Depois, para sair da fase Retransmissão Rápida espera pelo recebimento de um ACK que confirme todos os pacotes pendentes no momento de ser iniciada esta fase.

Os protocolos Tahoe, Reno, Vegas e New Reno reagem ao congestionamento reduzindo a janela de congestionamento sem ter em conta nenhuma estimação ou medida da largura de banda. O TCP Westwood, apresentado a continuação, utiliza uma medida da largura de banda para ajustar adequadamente a janela de congestionamento à capacidade de transferência da rede.

4.2 Protocolo TCP Westwood

O TCP Westwood (CASETTI et.al., 2001 e MASCOLO et.al., 2000) consiste em uma versão melhorada do TCP Reno, que modifica a forma de controle da janela de congestionamento (*cwnd*), do lado do remetente, usando uma estimativa da largura de banda. Monitorando a taxa de recepção de ACKs, a fonte TCP Westwood estima constantemente a taxa de pacotes da conexão. Essa estimativa é usada para calcular o tamanho da janela *cwnd* e o limiar *ssthresh*, que podem ser utilizados após a detecção de congestionamento, isto é, após o recebimento de três ACKs duplicados ou após um *timeout*. O raciocínio desta estratégia é simples, em contraste com o TCP Reno, que "cegamente" divide a janela de congestionamento pela metade, depois de três ACKs duplicados, o TCP Westwood tenta selecionar o Limiar de Partida Lenta (*ssthresh*) e uma janela de congestionamento (*cwnd*), a qual seja consistente com a largura de banda efetiva usada no tempo em que o congestionamento é experimentado. Este mecanismo é chamado Recuperação Mais Rápida. A vantagem deste mecanismo é que o emissor TCP se recupera mais rapidamente depois de perdas, especialmente, sobre conexões com

largos tempos totais de percurso ou sobre enlaces sem fio, onde perdas esporádicas, devido aos problemas de canais de radio, são frequentemente, mal interpretados, como sintomas de congestionamento e assim, conduz a uma redução desnecessária de janela.

4.2.1 Medição da largura de banda fim-a-fim do TCP Westwood

Uma filosofia do desenho fundamental do algoritmo de controle de congestionamento de TCP é que o algoritmo de controle de congestionamento deve ser realizado fim-a-fim. A rede é considerada uma "caixa preta". A fonte TCP não pode receber nenhuma realimentação explicita de congestionamento da rede e deve trabalhar com realimentação implícita como *timeout*, ACK duplicados, e medidas de tempo total de percurso. Assim, TCP realiza um controle fim-a-fim.

TCP Westwood (GRIECO, 2002, MASCOLO, 2000) propõe um novo mecanismo que adaptativamente fixa a janela de congestionamento depois de um episodio de congestionamento considerando a largura de banda usada quando o congestionamento acontece. TCP Westwood faz que a fonte realize uma estimativa fim-a-fim da largura de banda disponível ao longo da conexão TCP, medindo a taxa de retorno dos ACKs. Para que esta estimativa seja útil, a fonte deve inferir a quantidade de dados entregues ao receptor no tempo.

Quando um ACK é recepcionado pela fonte, ele comunica a informação de que uma quantidade correspondente a um pacote específico foi entregue ao destino. Se o processo de transmissão não é afetado por perdas, simplesmente calculando uma medida da contagem entregada dos dados sobre o tempo, rende uma avaliação justa da largura de banda usada atualmente pela fonte.

Quando ACKs duplicados (DUPACKs), indicando uma recepção fora de sequência, alcança a fonte, deve contar igualmente para a estimativa de largura de banda, e uma nova estimativa deve ser computada justo depois da sua recepção.

Entretanto, a fonte não poderá dizer certamente qual segmento provocou a transmissão de DUPACKs, e é assim é incapaz de atualizar a contagem dos dados pelo tamanho desse segmento. Uma media do tamanho do segmento enviado até esse momento na conexão em curso deve consequentemente ser usada**.**

Supondo que o tamanho é o mesmo para todos os pacotes, indicaria que o número de sequência está incrementado em um por segmento emitido, embora a execução real

do TCP se mantenha, de preferência, a par do número de bytes: as duas notações são permutáveis se os segmentos têm todos o mesmo tamanho.

É importante observar que, imediatamente depois de um episódio da congestão, seja por um *timeout* ou por *n* ACKs duplicados, o largura de banda está em saturação e a taxa de entrega da conexão é igual à parte da largura de banda do melhor esforço, isto é, a largura de banda de saturação, disponível para essa conexão. No estado estável, baixo as condições próprias estabelecidas na seção 3.3. do (MASCOLO et.al., 2000), esta deveria ser realmente a parte equitativa. Isto é confirmado pelo fato de que os pacotes tenham sido descartados, uma clara indicação de que filas estão em (ou perto de) saturação. Antes de um episodio de congestionamento, a largura de banda utilizada é menor ou igual à largura de banda disponível, porque a fonte TCP ainda está testando a capacidade da rede.

Como resultado, o TCP Westwood ajusta a entrada considerando a largura de banda disponível no tempo do congestionamento, enquanto outros TCP, como o TCP Reno, simplesmente divide a janela de congestionamento pela metade.

Se o ACK é recepcionado pela fonte no tempo $t(k)$, isto indica que $d(k)$ dados foram recepcionados pelo receptor TCP. Consequentemente, se pode medir a seguinte amostra de largura de banda utilizada como:

$$b(k) = \frac{d(k)}{t(k) - t(k-1)}$$

onde $t(k-1)$ é o tempo em que o ACK anterior é recepcionado. Devido a que o congestionamento acontece quando a taxa de trafego de entrada de baixa-frequência excede a capacidade do link (MASCOLO et.al., 2000), o TCP Westwood utiliza um filtro passo-baixo para achar a media das amostras e obter os componentes de baixa frequência da largura de banda disponível.

A seleção do filtro é importante. O TCP Westwood apresenta três tipos de filtros (UCLA COMPUTER SCIENCE DEPARTMENT, 2008), mas CASSETI et.al. (2001) dizem que a média exponencial simples do tipo usado pelo TCP para avaliação do RTT é incapaz de filtrar eficientemente os componentes de alta frequência das medições de largo de faixa. Logo, propõem o seguinte filtro de tempo discreto, o qual é obtido fazendo discreto um filtro de passo-baixo continuo, usando a aproximação de Tustin (CASETTI, 2001). Obtém-se assim:

$$\hat{b}(k) = \frac{\frac{2\tau}{t(k)-t(k-1)} - 1}{\frac{2\tau}{t(k)-t(k-1)} + 1}\hat{b}(k-1) + \frac{b(k)+b(k-1)}{\frac{2\tau}{t(k)-t(k-1)} + 1} \quad (4.1)$$

onde $\hat{b}(k)$ é a medida filtrada da largura de banda disponível no tempo $t = t(k)$, e $1/\tau$ é a frequência de corte do filtro. Note-se que $t(k) - t(k-1)$ é considerado o intervalo do ACK. A implementação do $\hat{b}(k)$ é referenciada como BWE do inglês *bandwith estimation* segundo o pseudocódigo de abaixo. Este código foi obtido da implementação do TCP Westwood para o simulador NS2, que pode baixar-se do *site* oficial do protocolo TCP Westwood da Universidade da Califórnia (UCLA COMPUTER SCIENCE DEPARTMENT, 2006).

A. Pseudocódigo da estimativa da largura de banda da rede[4]

```
int acked_size = size_ * 8 * cumul_ack;
double ack_interv = fr_now - lastackrx_;
double sample_bwe;
```

// Calcula quantos intervalos de comprimento tau_/2
// desde a recepção do último ACK.

```
int idle_intervals = (int)(ack_interv / tau_*2);
ack_interv -= tau_ /2 * idle_intervals;
sample_bwe = acked_size/ack_interv;

if (idle_intervals > 0)
  {
  for (int i=0; i<idle_intervals; i++)
    {
    BWE = BWE * 3 / 5  + last_bwe_sample_/5;
    last_bwe_sample_ = 0;
    }
  }
last_tmp_bwe = BWE;
```

// Calcula a estimativa da capacidade em bits.

```
BWE = BWE*(2*tau_/ack_interv - 1)/2*tau_/ack_interv + 1)
   +(sample_bwe+last_bwe_sample_)/(2*tau_/ack_interv+1);
```
(4.2)

[4] Por razões de espaço é apresentado somente parte deste código.

4.2.2 Algoritmo de controle de congestionamento do TCP Westwood

O algoritmo de controle de congestionamento do TCP Westwood é executado no emissor, para realizar uma recuperação mais rápida, depois do evento de congestionamento. A idéia geral é utilizar a largura de banda estimada BWE para ajustar o tamanho da janela de congestionamento (*cwnd*) e o Limiar de Partida Lenta (*ssthresh*) depois de algum episodio de congestionamento. Lembrando que o rol básico jogado por *cwnd* e *ssthresh* no controle de congestionamento TCP é que *cwnd* é incrementado e decrementado, para seguir o produto largura de banda-atraso que deve ser representado por *ssthresh*.

Se o congestionamento acontece devido a *n* ACK duplicados, se tem o seguinte pseudocódigo:

B. Pseudocódigo na ocorrência de *n* ACKs duplicados

```
if (n DUPACKs são recepcionados)
  ssthresh = (BWE * RTTmin) / seg_size;
  if (cwnd > ssthresh)          // evitar o congestionamento
    cwnd = ssthresh;
  endif
endif
```

Note-se que *seg_size* identifica o tamanho do segmento TCP em bits.

Durante a fase de Prevenção de Congestionamento se testa por largura de banda disponível extra. Consequentemente, quando *n* DUPACKs são recepcionados, significa que se tem atingido a capacidade da rede (ou no caso de conexões sem fio, um ou mais segmentos foram descartados devido à perdas esporádicas). Assim, o Limiar de Partida Lenta é ajustado ao tamanho da conexão disponível quando a fila do gargalo está vazia, que é $BWE * RTT_{\min}$, a janela de congestionamento é ajustada igual ao *ssthresh* e na fase de Prevenção de Congestionamento é se testa o novo largo de faixa disponível. O valor $RTT_{\min}$ é ajustado à menor amostra do RTT observado sobre a duração da conexão. Este ajuste permite à fila ser drenada depois de um episodio de congestionamento. Durante a fase Partida Lenta ainda se esta testando o largo de faixa disponível. Consequentemente, o *BWE* é obtido depois de *n* ACK duplicados e é usado para ajustar o patamar da Partida Lenta. Depois que o *ssthresh* for ajustado, a janela de congestionamento é ajustada ao patamar de Partida Lenta, somente se

cwnd > *ssthresh*. Em outras palavras, durante a fase de Partida Lenta, *cwnd* ainda caracteriza um incremento exponencial como na implementação do TCP Reno.

Se o congestionamento acontece por *timeout* se tem o seguinte pseudocódigo:

C. Pseudocódigo na ocorrência da expiração do *timeout*

```
if (timeout expira)
  ssthresh = (BWE * RTTmin)/seg_size;
  if (ssthresh < 2)
    ssthresh = 2;
  endif;
  cwnd = 1;
endif
```

Depois que um evento timeout acontece, o *cwnd* é ajustado, igual a 1, o comportamento básico do Reno, quando uma recuperação rápida é garantida por o *ssthresh* ajustado à estimativa da largura de banda no tempo da expiração do *timeout*.

A dinâmica do algoritmo pode ser representada pelo diagrama da Figura 4.1.

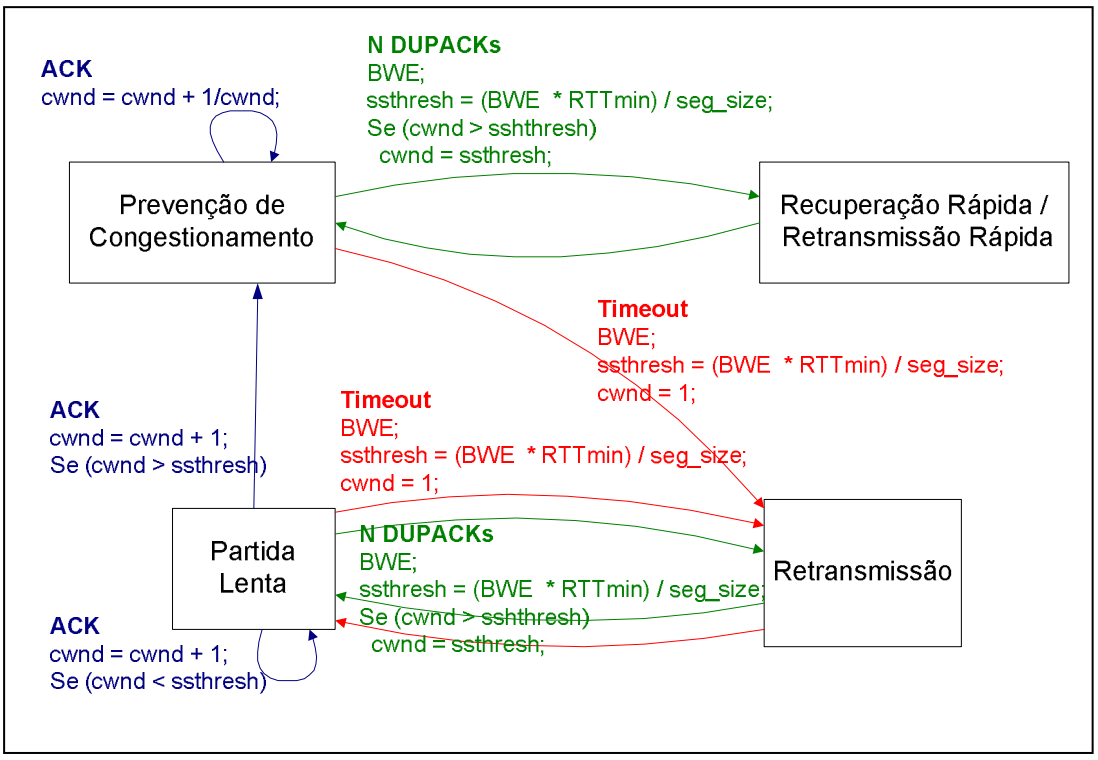


Figura 4.1: Diagrama do algoritmo de controle de congestionamento do TCP Westwood.

4.2.3 TCP Westwood rendimento, equidade e compatibilidade

Os resultados experimentais apresentados em (CASETTI et.al., 2001) demonstram a efetividade do algoritmo na estimativa da largura de banda. A Figura 4.2 apresenta o comportamento do processo de estimativa da largura de banda do algoritmo TCPW de uma única conexão, compartilhando o canal com duas conexões UDP. A configuração da rede tem um canal de gargalo de 5Mb/s com tempo de atraso de propagação de um sentido de 30 ms.

A Figura 4.2 além de demonstrar a exatidão do algoritmo de estimativa da largura de banda, ilustra a capacidade de uma conexão TCP Westwood de utilizar largura de banda restante dos fluxos UDP dinâmicos.

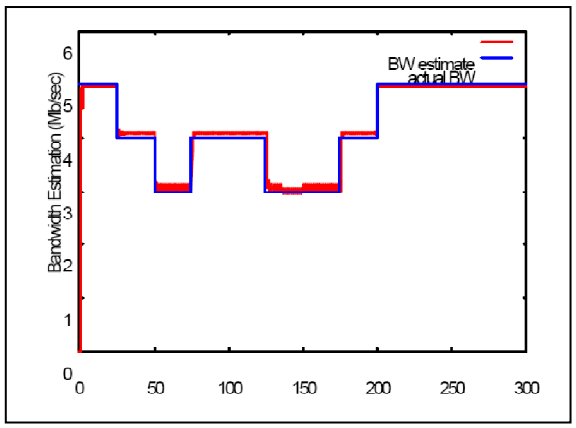


Figura 4.2: TCP_W com trafego UDP concorrente – estimativa da largura de banda.

CASETTI et.al. (2001) também apresenta resultados onde o TCP Westwood apresenta melhor rendimento, comparado com TCP Reno, assim como, a capacidade de coexistir com outras variações de TCP em forma satisfatória.

4.3 Protocolo TCP Westwood Equilíbrios Virtuais

Baseados no algoritmo proposto na seção 3.2 podemos implementar um novo protocolo de camada de transporte, que ajuste os parâmetros de controle da janela de congestionamento, segundo dados da estimativa da capacidade da rede (usando a estimativa do TCP_W) e conhecendo o número de usuários que compartilham a mesma. A Figura 4.3 mostra o esquema da implementação desta nova proposta, chamada TCP Westwood Equilíbrios Virtuais devido a que utiliza a estimativa da capacidade do protocolo Westwood e as equações (2.12) e (2.13) da proposta de Equilíbrios Virtuais.

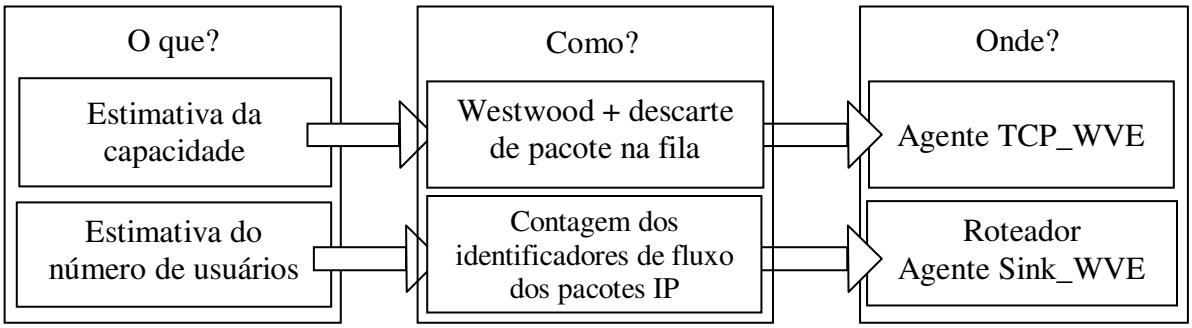


Figura 4.3: Esquema da implementação do protocolo TCP_WVE.

4.3.1 Estimativa da largura de banda fim-a-fim do TCP Westwood Equilíbrios Virtuais

A estimativa da capacidade da rede, segundo Westwood, é realizada pela implementação da equação (4.1), como foi apresentada na seção anterior. No entanto, como este valor realmente é utilizado para ajustar o limiar que separa as fases Partida Lenta e Prevenção de Congestionamento, ele não representa realmente a capacidade máxima da rede em uso. Os resultados experimentais realizados no marco de este trabalho demonstram que se utilizamos a estimativa pura do TCP_W como dado para calcular dinamicamente os parâmetros AIMD, o ajuste do tamanho da janela de congestionamento se estabiliza por abaixo do valor real da capacidade da rede.

Este trabalho propõe uma sobre estimativa suave que faz com que a dinâmica alcance uma capacidade mais próxima à real da rede e oscile entorno a ela. Deste modo, quando acontece uma perda de pacote (indicada por ACK duplicados ou *timeout*) a capacidade real da rede é alcançada e este valor pode ser utilizado para recalcular os parâmetros do algoritmo AIMD, segundo (2.12) e (2.13).

O agente TCP_WVE recepciona um pacote e ao estimar a largura de banda da rede segundo (4.2) adiciona a seguinte linha ao pseudocódigo[5] do TCP_W para aplicar a nova estimativa mencionada. A estimativa da capacidade é implementada no agente emissor TCP_WVE.

```
// Se sobreestima o calculo anterior.
BWE = sobre_estimativa * BWE;                    (4.3)
```

[5] Código completo dos protocolos implementados encontra se na versão digital.

4.3.2 Cálculo do número de usuários do TCP Westwood Equilíbrios Virtuais

Para a estimativa do número de usuários da rede é criado um novo cabeçalho que será adicionado ao pacote (com IPV6). Ele tem dois campos, um campo de *tipo* e um campo para o número de fluxos ou *nflu*. O campo *tipo* indica a direção do pacote; quando *tipo* é igual a 1 o pacote vai da fonte (agente TCP_WVE) para o destino (TCP_WSink); e quando é igual a 2 é o pacote ACK que retorna. O segundo campo, o de número de fluxos, guarda o número de fluxos (ou usuários) que compartilham um canal.

Antes do envio do pacote, o agente TCP_WVE armazena no cabeçalho tipo = 1 e *nflu* o valor 1, porque como padrão será, pelo menos, um usuário que esta transmitindo dados. Quando o pacote chega ao roteador este lê o identificador de fluxo ou *flowid* e verifica que esse fluxo já esta na contagem olhando para uma estrutura (por exemplo, uma tabela) donde armazena o *flowid* de todos os fluxos que compartilham esse canal. Se o novo fluxo não está ainda guardado, ele é adicionado e o contador de fluxos é incrementado em um. Assim, o roteador pode determinar quantos usuários estão transmitindo nesse canal. Se o *flowid* do pacote já foi contado, o contador não é incrementado. Antes que o pacote seja reenviado para o destinatário, o roteador armazena o valor do contador no campo *nflu* do cabeçalho e reenvia o pacote. A estrutura deve ser restaurada cada certo tempo para que os fluxos que deixam de transmitir não sejam tomados em conta na contagem.

Cada roteador tem a sua própria contagem. Quando um pacote chega, o roteador compara o campo *nflu* do cabeçalho do pacote que chega com o contador interno, e coloca no campo *nflu* o maior valor encontrado. Isto é necessário porque se deve escolher o máximo valor contado no caminho do pacote, já que corresponde ao canal gargalo.

Quando o pacote chega ao destino (agente TCP_WSink), o campo *tipo* é trocado para o valor 2, e o cabeçalho é adicionado ao ACK para ser reenviado ao agente emissor. Assim, o emissor tem um valor atualizado do número de usuários cada RTT. Se o número de usuários varia, o parâmetro de incremento aditivo deve ser recalculado segundo

$$a_1 = (a_1 * nflu\_anterior) / nflu \qquad (4.4)$$

4.3.3 Controle de congestionamento do TCP Westwood Equilíbrios Virtuais

O agente emissor TCP_WVE controla a janela de congestionamento incrementando cada vez que recebe o último reconhecimento (ACK) da mencionada janela, e se acontecer uma indicação de perda de pacotes (por *timeout* ou ACKs duplicados) ele retransmite o pacote perdido e decrementa a janela a fim de reduzir o fluxo de dados enviados.

O TCP_WVE propõe que a janela de congestionamento se incremente em um valor calculado segundo a equação (2.1.a) considerando o valor estimado da capacidade da rede e do número de usuários. Assim, a dinâmica de incremento será mais rápida levando a melhores tempos de resposta.

Quando o agente emissor TCP_WVE recebe o último ACK da uma janela de congestionamento, ele pode incrementar o seu tamanho, já que todos os dados enviados nessa janela foram confirmados. Assim, ele deve aplicar a dinâmica de incremento segundo o seguinte pseudocódigo.

A. Pseudocódigo sem ocorrência de perdas (dinâmica de incremento)

```
//cálculo do número de pacotes da rede usando o máximo
//valor encontrado de estimativa de largura de banda em bits[6].
BWEpq = (maximo_BWE * RTT_min)/((pksize_+headerSize_)*8);

dbwepq = (maximo_BWE *
min_rtt_estimate)/((size_+headerSize_)*8);

if(max_bwepq < bwepq)
  max_bwepq = bwepq;

ant_dai = dai;

//se ainda não aconteceu a primeira perda
  if(F_FDROP == 0)
    {
    razon = max_bwepq/nflu_;
    if(max_razon < razon)
      max_razon = razon;
    dai = razon * (1-dbi) + aux1;    //aux1 < 1
    if(max_dai < dai)
      max_dai = dai;
    //aplicar a din. de incremento (2.1.a) com bi fixo
    cwnd = max_dai + dbi * cwnd;
```

[66] O máximo valor encontrado é utilizado porque o método vai detectando o valor lentamente a medida que o tempo transcorre o valor é incrementado. Logo é utilizado o máximo encontrado até esse momento.

```
  }
//depois da primeira perda de pacote
else
  {
  //se não existe mudança no número de usuários
  if(F_NFLU ==0)
    {
    //o usuário estava transmitindo quando teve troca no nflu
    if(F_TRANS == 1)
      {
      dai = dai;
    //aplicar a din. de incremento (2.1.a) com bi fixo
      cwnd = dai + dbi * cwnd;
      razon = max_dbwepq/nflu_;
      }
    else
      {
      razon = max_bwepq/nflu_;
      if(max_razon < razon)
        max_razon = razon;
      //calcular o parâmetro da din. de incremento (2.1.a) com bi fixo
      ai =  max_razon * (1-dbi) + α * aux1; //α < 1
      if(max_ai < ai)
        max_ai = ai;
      //aplicar a din. de incremento (2.1.a) com bi fixo
      cwnd = max_ai + bi * cwnd;
      }
    }
//se teve cambio no nflu, se usa (4.4)
else
    {
    razon = max_bwepq/nflu_;
    max_razon = razon;
    //aplicar a din. de incremento (2.1.a) com bi fixo
    cwnd = dai + dbi * cwnd;
    }
  }
```

O pseudocódigo anterior recalcula o parâmetro de incremento a_I segundo a equação (2.12), sendo o valor de b_I fixo e utiliza este valor enquanto o número de usuários ou fluxos não varia. A variável aux_1 é o incremento aplicado para que se cumpra a relação "≥" da equação (2.12).

O comportamento da dinâmica de decremento no caso de evento de n ACKs duplicados (normalmente n=3) ou *timeout* segue a equação (2.1.b) do algoritmo AIMD de Chiu e Jain, com a diferença que, o valor a_D é recalculado cada vez que acontece

um evento de perda de pacote. O calculo do valor a_D segue a equação (2.13) do modelo AIMD_VE e a variável aux_2 é o decremento aplicado para que se cumpra a relação "≤" da equação.

Assim, para o caso de perdas o protocolo TCP_WVE apresenta o seguinte pseudocódigo

B. Pseudocódigo na ocorrência de perdas (dinâmica de decremento)

```
//cálculo do número de pacotes da rede usando o máximo
//valor encontrado de estimativa de largura de banda em bits.
BWEpq = (maximo_BWE_ * RTTmin)/((pksize_+headerSize_)*8);

//utilizar a máxima estimativa encontrada
if(max_BWEpq < BWEpq)
  max_BWEpq = BWEpq;

razon = max_dBWEpq/nuser_;

//utilizar a máxima razão "cap/n" encontrada
if(max_razon < razon)
  max_razon = razon;

//calcular o parâmetro mult. da dinâmica de decremento
ad = aux2 * max_drazon * (1-bd);     // aux2 > 1

//se é a primeira perda de pacote, guardar o valor
if(F_FDROP == 0)
  {
  max_BWEpq = BWEpq;
  F_FDROP = 1;
  }

//aplicar a dinâmica de decremento (2.1.b) com bd fixo
cwnd = ad + bd * cwnd;

reenviar o pacote perdido();
```

4.4 Protocolo TCP Equilíbrios Virtuais

O Protocolo TCP Equilíbrios Virtuais é a implementação do algoritmo proposto na seção 3.2.2. Nesta proposta, o protocolo incrementa um valor inicial do parâmetro a_I da dinâmica de incremento (equação 2.1.a) até acontecer a primeira perda de pacotes indicação que o protocolo deve aplicar uma dinâmica de decremento (equação 2.1.b).

A Figura 4.5 mostra o esquema da implementação desta nova proposta que chamada TCP Equilíbrios Virtuais.

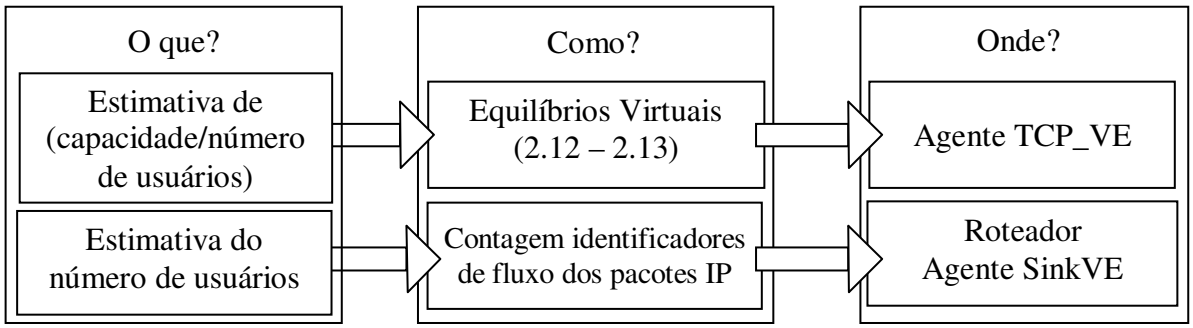


Figura 4.4: Esquema de implementação do protocolo TCP_VE.

4.4.1 Controle de congestionamento do TCP Equilíbrios Virtuais

O controle de congestionamento do TCP_VE aplica a dinâmica de incremento partindo com um valor inicial do a_I e com o valor de b_I fixo. Se não acontece nenhuma perda incrementa o valor de a_I em um valor constante; assim, até acontecer a primeira perda de pacotes (por *timeout* ou 3 ACKs duplicados). Esta primeira perda de pacote é uma indicação que a capacidade máxima foi atingida e que o agente TCP emissor deve aplicar a dinâmica de decremento (equação 2.1.b). O protocolo deve calcular os parâmetros de decremento da equação 2.1.b. Se mantemos fixo o parâmetro multiplicativo da equação, o parâmetro aditivo a_D será calculado segundo a equação (3.3).

Lembre se que depois de uma perda o valor de a_I pode ficar constante e fixado a uma percentagem um pouco menor, e que cada vez que se tem uma perda de pacotes a_D pode ser recalculado. Também, que quando se tem uma variação do número de usuários deve-se recalcular o parâmetro a_I que pode ser feito do mesmo modo que a implementação do protocolo TCP_WVE (equação 4.4).

O valor do número de usuários pode ser obtido se dos roteadores da rede do mesmo modo que no protocolo TCP_WVE. Este valor é necessário, especialmente, para poder reinicializar a variável a_I, devido a que os parâmetros das dinâmicas de incremento e decremento são estipulados dependendo do valor c/n. Se, por exemplo, um dado valor do parâmetro a_I é eficiente para dois usuários transmitindo, esse mesmo

valor de a_1 resultaria muito elevado se um terceiro usuário transmitisse seus dados, o que acarrearia perdas de múltiplos pacotes e, por conseguinte um protocolo ineficiente.

Deste modo o controle de congestionamento do TCP_VE pode ser resumido pelos pseudocódigos:

A. Pseudocódigo sem ocorrência de perdas (dinâmica de incremento)

```
//quando chegar o último ACK da janela de congestionamento
//se ainda não aconteceu a primeira perda
if(F_FDROP == 0)
  {
  ai = ai + α;        //α<1, exemplo 0.01
  if(max_ai < ai)
    max_ai = ai;
  //aplicar a din. de incremento (2.1.a) com bi fixo
    cwnd_ = max_ai + bi * cwnd_;
  }
//perda de pacote
else
  {
  max_ai = ai;
  //aplicar a din. de incremento (2.1.a) com bi fixo
  cwnd_ = max_ai + bi * cwnd_;
  }
```

B. Pseudocódigo na ocorrência de perdas (dinâmica de decremento)

```
// calcular o valor "cap/n"
razon = ai / (1-bi);
ad = β * razon * (1-bd);  // β<1, por exemplo 0.8
//aplicar a dinâmica de decremento (2.1.b) com bd fixo
cwnd = ad + bd * cwnd;
F_FDROP = 1;
reenviar o pacote perdido();
```

Os protocolos TCP_WVE e TCP_VE foram implementados no simulador NS2 e comparados com os protocolos TCP New Reno e TCP Westwood, como se verá no seguinte capítulo[7].

Síntese do capitulo

Este capítulo apresentou primeiramente um resumo dos protocolos de controle de congestionamento tradicionais TCP Tahoe, Reno, Vegas e New Reno. Em TCP Tahoe e

[7] Os códigos fontes dos protocolos encontram-se no CD anexo.

TCP Reno a fase *Additive Increase*, onde a janela de congestionamento é incrementada, é adotada exatamente como no AIMD, quando os protocolos estão em Prevenção do Congestionamento. Em caso de perda de um pacote, TCP Tahoe reinicializa a janela de congestionamento e o protocolo entra de novo na fase Partida Lenta. Em TCP Reno, quando o remitente recebe três DACKs, um *Multiplicative Decrease* é usado para justar a janela de congestionamento (*cwnd*) e o valor do Limiar de Partida Lenta (*ssthresh*). Neste caso, o protocolo permanece na fase Prevenção do Congestionamento. Quando o *Timeout* expira, TCP Reno entra na fase Partida Lenta, igual ao TCP Tahoe. O TCP Vegas faz que o incremento ou decremento do tamanho da janela seja linear de acordo com o valor estimado dos pacotes guardados. O TCp New Reno é o mesmo que Reno porém com mais inteligência durante a recuperação rápida. Utiliza a idéia de ACKs parciais: quando há perda múltiple de pacotes, os ACKs para o pacote retransmitido reconhecerão alguns, mas não todos os segmentos enviados.

Seguidamente, foi introduzido o protocolo TCP Westwood. O TCP Westwood introduz um mecanismo de recuperação rápida para evitar uma sobre-redução da janela de congestionamento depois de três ACK duplicados considerando para isto uma estimativa fin-a-fin da largura de banda disponíveis.

Finalmente, o capítulo mostrou dos novos protocolos de controle de congestionamento, o TCP Westwood Equilíbrios Virtuais e o TCP Equilíbrios Virtuais. As novas propostas, ao contrario dos protocolos tradicionais TCPs, utilizam uma função de incremento e uma função de decremento dinamicamente adaptativa. Os protocolos propostos neste trabalho tampouco apresentam as duas fases tradicionais de Partida Lenta e Prevenção do Congestionamento; pelo contrario, utilizam uma dinâmica de incremento e uma dinâmica de decremento dependendo se acontece ou não perda de pacotes reagindo à demanda de comunicação da Internet de forma eficaz como será mostrado no seguinte capítulo.

5. Resultados de simulações comparativas

Este capítulo apresenta e discute os resultados dos estudos realizados através de simulações lineais utilizando a linguagem C com a ferramenta *Visual Studio.NET* para simulação dos algoritmos propostos e simulações no simulador *Network Simulator NS2* versão 2.32 para implementação e simulação dos protocolos propostos.

Para cada estilo de simulação são feitas considerações generais sobre a ferramenta utilizada, o objetivo dos estudos, os cenários escolhidos e as métricas de desempenho. Além disso, também é apresentado um comparativo das propostas com algoritmos e protocolos tradicionais para variados cenários.

5.1 Simulações de propostas inspiradas em AIMD

5.1.1 Técnica de avaliação e objetivos

Com o objetivo de avaliar o desempenho dos algoritmos propostos baseados no modelo AIMD, foram realizadas primeiramente simulações lineais utilizando a linguagem C com a ferramenta *Visual Studio.NET*[8] para estudo de casos com atrasos homogêneos e heterogêneos comparando o AIMD com o modelo de Equilíbrios Virtuais (VE).

O protocolos propostos TCP_WVE e TCP_VE no capítulo 4, foram implementados utilizando o software de simulação Network Simulator versão 2.33 ou NS-2 (UNIVERSITY OF SOUTHERN CALIFORNIA, 2006)[9]. Amais do agente emissor, foi implementada uma fila tipo DropTail modificada, chamada *DDropTail*, que implementa a contagem do número de usuários necessário para os protocolos propostos (ver detalhe secção 4.3.2). A fila DDropTail foi implementada em forma de uma tabela com uma entrada por cada usuário (ou fluxo) que passa pelo roteador. O roteador lê o *flowid* do cabeçalho criado para isso (ver Seção 4.3.2) e adiciona à tabela se este ainda não está na mesma, incrementando assim o número de usuários. Como deve ser considerado o decremento do número de usuários, neste trabalho a tabela é

[8] Os códigos fontes dos algoritmos encontram-se no CD anexo
[9] Os códigos fontes dos protocolos encontram-se no CD anexo

reinicializada de tempo em tempo, sendo também reiniciada a contagem. Esta simples implementação foi escolhida somente para verificar o funcionamento dos protocolos propostos. Implementações mais eficientes e inteligentes, do registro de fluxos que passam pelo roteador, podem ser feitas; por exemplo, algum tipo de tabela *hash* de um o dois níveis.

Além disso, um agente do tipo TCPSink foi implementado. Este agente, ademais das funcionalidades do TCPSink, recebe o pacote, lê o número de usuários calculado no caminho pelos roteadores e reenvia a informação em um campo especial do ACK.

O simulador *Network Simulator NS2* versão 2.32 conta com módulos dos protocolos tradicionais como o TCP Tahoe, Reno, Vegas e New Reno, dentre outros. Os módulos do TCP Westwood estão disponíveis em (UCLA COMPTER SCIENCE DEPARTMENT) e foram adicionados ao simulador. Para ambos os protocolos, propostos novos módulos foram programados para a versão correspondente do NS2.

O objetivo dos estudos lineais realizados é avaliar o melhor comportamento dos algoritmos propostos em termos de uma melhor utilização da largura de banda, tempo de resposta, convergência, tamanho das oscilações e equidade entre taxas de transmissão dos usuários quando os parâmetros das dinâmicas de incremento e decremento se escolhem de forma adequada seguindo as equações (2.12) e (2.13) do capítulo 2.

O objetivo dos estudos realizados no simulador NS2 é avaliar o comportamento dos protocolos propostos em comparação com os protocolos TCP New Reno (tradicionalmente utilizado) e TCP Westwood, que serviu de base para o protocolo TCP_WVE. Além disso, avaliar também se os mesmos podem conviver juntamente com os protocolos utilizados hoje em dia, como o TCP New Reno.

5.1.2 Simulações em Linguagem C

A escolha adequada dos parâmetros do AIMD proposto por BHAYA e KASZKUREWICZ (2007), chamado VE, propõe o algoritmo tradicional AIMD com uma escolha de parâmetros segundo as equações (2.12) e (2.13), que conduzem a um melhor tempo de resposta, suavidade, e uma melhor convergência à equidade ótima.

Conforme mencionado no Capítulo 2, a escolha do parâmetro tem uma influência direta no comportamento do algoritmo AIMD. Com o objetivo de ilustrar as melhoras

sobre os parâmetros AIMD tradicionais de Chiu e Jain, considere os estudos de casos descritos.

Estudo de caso #1: Considere duas fontes, ou usuários, x_1 e x_2 que compartilham um único canal de capacidade cap pacotes/segundo.

O modelo S_AIMD_CJ [10] proposto por CHIU e JAIN (1989) com atrasos homogêneos (ou síncrono), os parâmetros $cap = 1$, $a_I = 0.01$, $b_I = 1$, $a_D = 0$, $b_D = 0.9$, e as condições iniciais $x_1(0) = 0.3$ e $x_2(0) = 0.1$ apresenta como resultado o comportamento ilustrado na Figura 5.1. Pode-se observar que partindo das condições iniciais, com os parâmetros escolhidos, o sistema atinge o ponto de estabilidade ótimo $x_1^* = x_2^* = 0.5$.

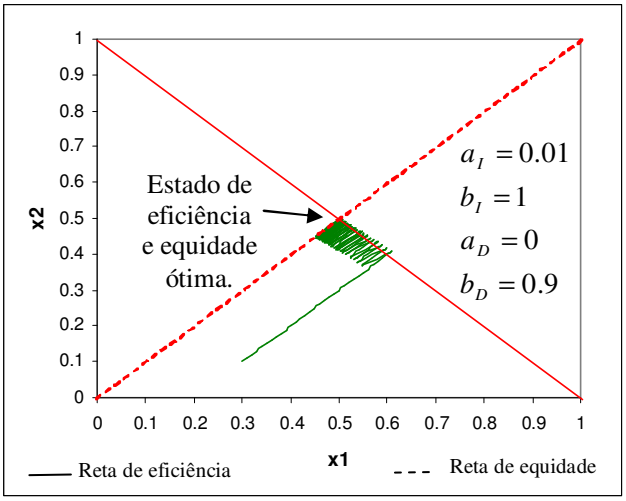


Figura 5.1: Diagrama de estados do S_AIMD_CJ para o estudo de caso #1 – Dados obtidos da simulação sequencial.

Considere o estudo de caso #1 com os parâmetros $a_I = 0.06$, $b_I = 0.9$, $a_D = 0.04$, e $b_D = 0.9$, seguindo (2.12) e (2.13) considerando o valor desconhecido $0.4 < cap / 2 < 0.6$. A Figura 5.2 mostra o resultado da simulação do algoritmo VE para o caso exemplo com as condições iniciais $x_1(0) = 0.3$ e $x_2(0) = 0.1$, enquanto a Figura 5.3 mostra a comparação dos dois modelos.

[10] Lembre se da sessão 2.1 a notação de S_AIMD_CJ para a versão síncrona do AIMD e A_AIMD_G para a versão assíncrona do mesmo.

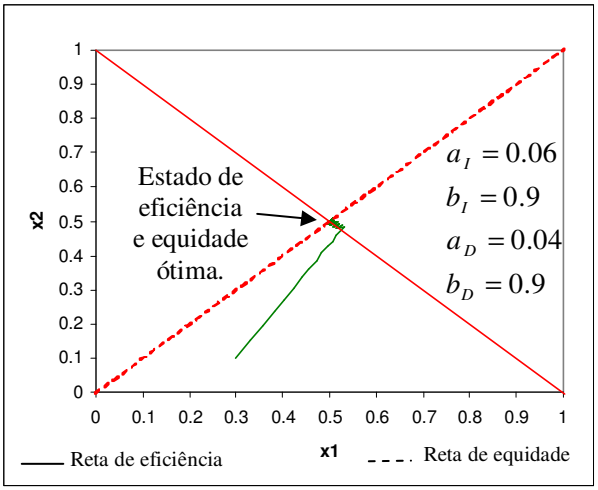


Figura 5.2: Diagrama de estados do modelo VE para o estudo de caso #1 – Dados obtidos da simulação sequencial.

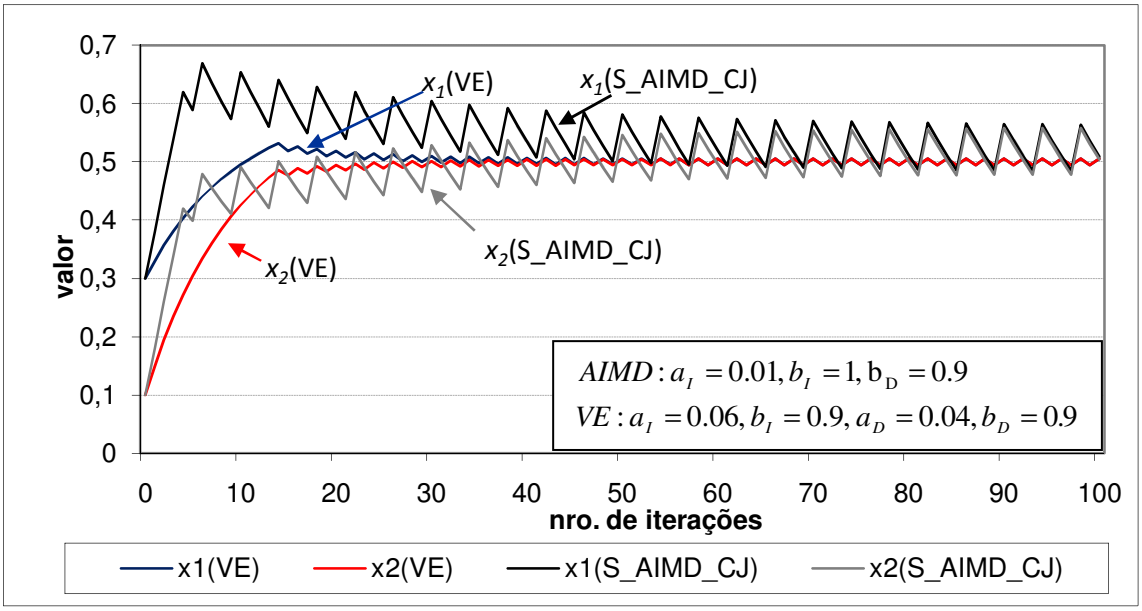


Figura 5.3: Comparação dos modelos S_AIMD_CJ e VE para o estudo do caso #1. - Respostas no tempo – Dados obtidos da simulação sequencial.

As Figuras 5.1, 5.2 e 5.3 mostram que esta escolha de parâmetros apresenta um melhor comportamento do algoritmo com as mesmas condições iniciais. Nota se que a dinâmica de incremento do S_AIMD_CJ faz com que a resultante se mova para a reta de eficiência. Sem embargo, uma melhor escolha de parâmetros (VE) faz com que ela se mova para a reta de eficiência, assim como para a reta de equidade. Similarmente, os parâmetros da dinâmica de decremento do VE fazem que a mesma se mova para a reta de equidade e para a reta da eficiência.

Além disso, o valor da suavidade das oscilações é notavelmente menor com a melhor escolha de parâmetros. Pode-se conferir isso considerando-se o intervalo de suavidade das oscilações definida em (BHAYA e KASZKUREWICZ, 2007) como:

$$dist\_smoothness = \sqrt{2}\left[(a_I - a_D) + (cap/2)(b_I - b_D)\right] \quad (5.1)$$

Esta distancia tem valor 0.084 para o modelo S_AIMD_CJ e um valor de 0.028 para VE.

Consideremos agora o comportamento do modelo VE frente ao AIMD em presença de atrasos heterogêneos

Estudo de caso #2: Considere duas fontes, ou usuários, x_1 e x_2 que compartilham um único canal de capacidade $cap = 100$ pacotes/segundo. Os valores dos atrasos heterogêneos utilizados são $d_1^f = d_1^b = 4$ para o usuário 1 e $d_2^f = d_2^b = 1$ para o usuário 2.

O comportamento dos modelos A_AIMD_G e VE na presencia de atrasos heterogêneos é estudado primeiro, com condições iniciais justas sem embargo não eficientes (FI: $x_1(0) = 0, x_2(0) = 0$) e, em seguida, com condições iniciais não justas, mas eficientes (EU: $x_1(0) = 100, x_2(0) = 0$).

Os parâmetros escolhidos para o caso do algoritmo A_AIMD_G são os parâmetros tradicionais do protocolo TCP: $a_I = 1$, $b_I = 1$, $a_D = 0$, e $b_D = 0.5$; e para VE: $a_I = 6$, $b_I = 0.9$, $a_D = 4$, e $b_D = 0.9$.

As Figuras 5.4 e 5.5 mostram os resultados da simulação para as condições iniciais mencionadas. Observando ambas as figuras, pode-se verificar que, na presencia de atrasos heterogêneos a escolha dos parâmetros do modelo VE segundo (2.12) e (2.13) mostra um melhor tempo de convergência assim como uma suavidade das oscilações muito menor.

Assim, o modelo VE na presencia de atrasos heterogêneos exibe um melhor comportamento, já que a escolha adequada dos parâmetros das dinâmicas de incremento e decremento faz que o mesmo convirja rapidamente, suavemente e eficientemente.

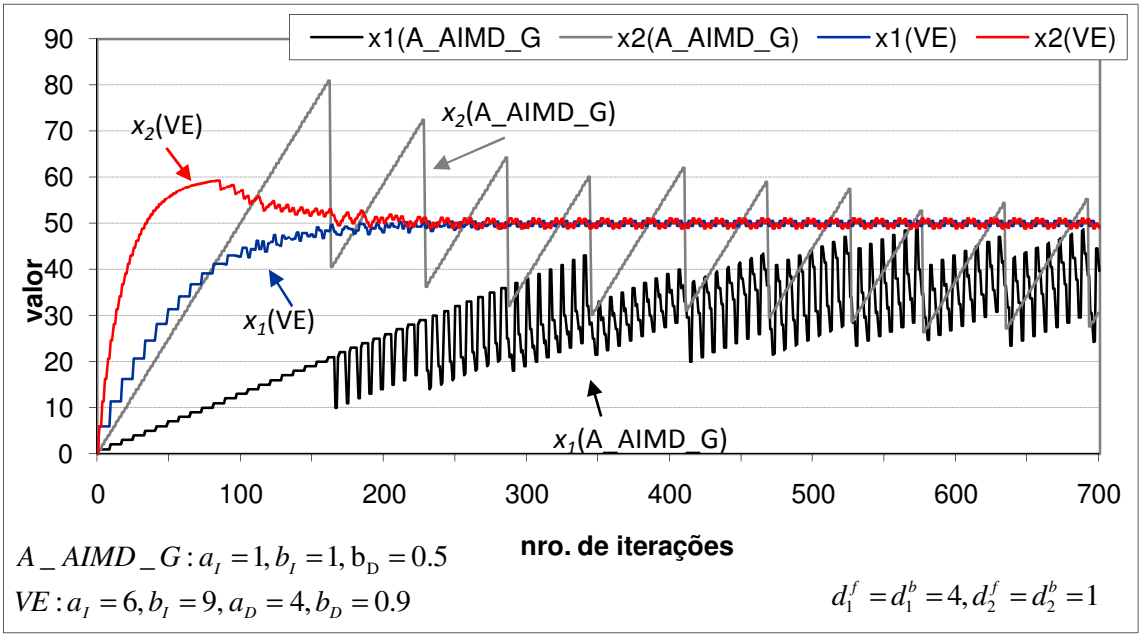


Figura 5.4: Comparação dos modelos A_AIMD_G, VE para o estudo de caso #2 com presença de atrasos heterogêneos e condições iniciais FI $x_1(0) = 0, x_2(0) = 0$ – Dados obtidos da simulação sequencial.

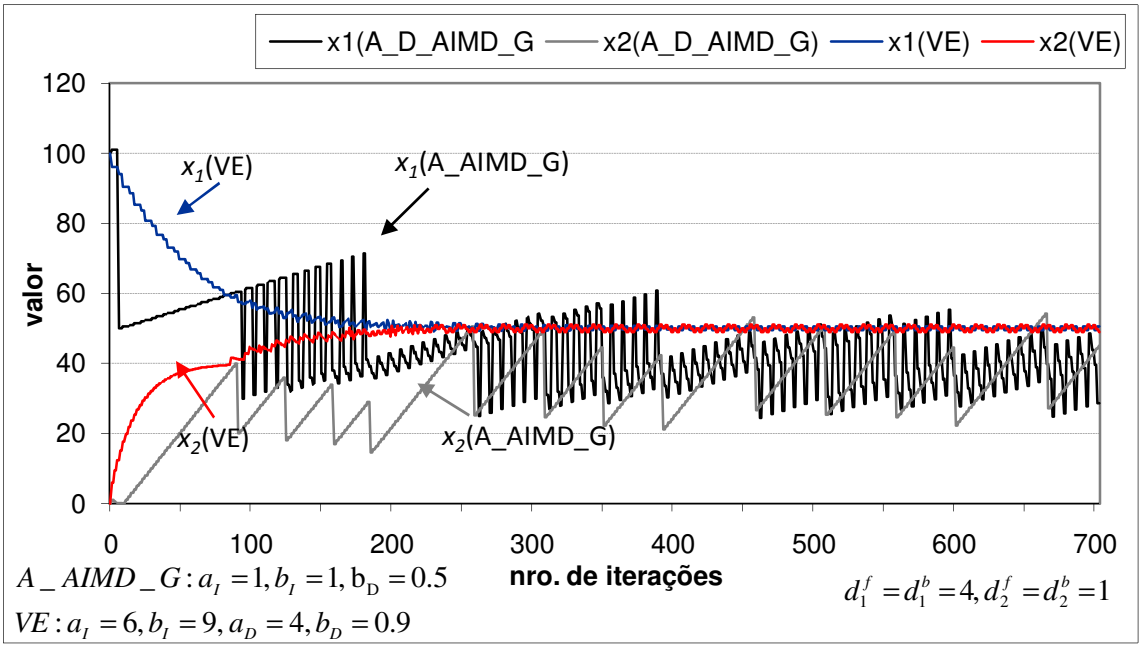


Figura 5.5: Comparação dos modelos A_AIMD_G, VE para o estudo de caso #2 com presença de atrasos heterogêneos e condições iniciais EU $x_1(0) = 100, x_2(0) = 0$ – Dados obtidos da simulação sequencial.

5.1.3 Simulações em NS2

Nesta seção são apresentados resultados experimentais que exemplificam o comportamento dos novos protocolos propostos no Capítulo 4: o TCP Westwood Equilíbrios Virtuais TCP_WVE e o TCP Equilíbrios Virtuais TCP_VE.

Para o estudo das simulações foi utilizado o modelo tradicional conhecido como *dumbbell* que é freqüentemente utilizado no estudo de redes (LEE et.al, 2005). A topologia do *dumbbell* é um conceito muito comum em redes de computadores (OSISCHOOL). A topologia do *dumbbell* mostra um número de nós (geralmente mais de 2) conectados a um único roteador. O roteador é conectado a um outro roteador sobre uma ligação lenta de série. Um grupo de nós é conectado a esse roteador, criando a topologia do *dumbbell*. A topologia do *dumbbell* é usada para indicar encenações da rede comum. Por exemplo, um grupo de computadores está tentando conectar a um grupo de usuários que compartilham toda de uma ligação lenta da série da largura de faixa. Nas simulações deste trabalho utiliza se uma topologia simples do *dumbbell* de 2 remetentes e de 2 receptores como pode verse a continuação.

As topologias simples, como uma topologia do *dumbbell* com um canal congestionado, são suficientes para estudar muitas propriedades do tráfego (FLOYD e KOHLER, 2002). Esta afirmação baseia se no fato que é possível reduzir o problema de modelar uma rede grande rede de computadores em um problema de topologia *dumbbell*. Una grande rede modelada pode ser decompor em sub-redes com um único canal de gargalo em cada sub-rede, cada sub-rede que estará usado uma topologia do *dumbbell* é ao redor do seu canal de gargalo (DAS et.al, 2008). Assim, para as simulações em NS2 se utiliza o:

Estudo de caso #3: Para as simulações foi utilizada uma rede com dois nós que compartilham um único canal através do qual três usuários enviam seus dados (fluxos FTP); como pode ser observado na Figura 5.6.

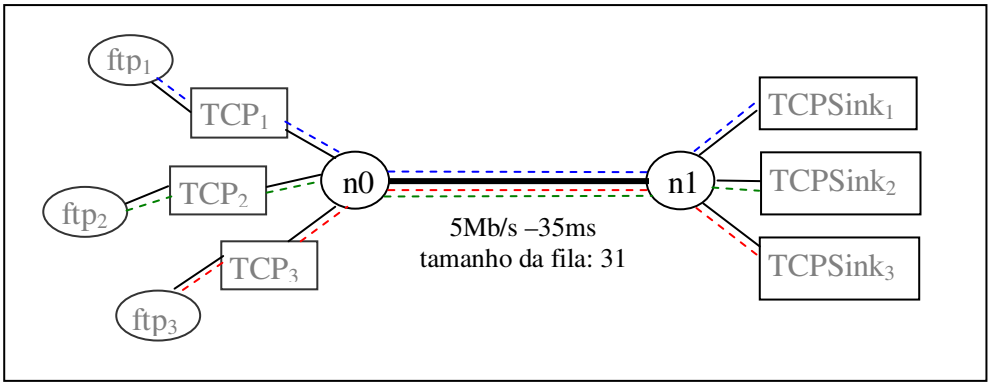


Figura 5.6: Configuração da rede utilizada para simulação no NS2.

O canal entre os nos n_0 e n_1 é *half-duplex* de 5 Mb/s, o tempo de propagação é de 35 ms. É utilizada uma fila do tipo Drop Tail de tamanho 31 para conter um número de pacotes igual ao produto da largura de banda por o atraso. Os pacotes têm tamanhos iguais à 1400 Bytes. A capacidade máxima da rede é a soma da capacidade do canal (pacotes em trânsito) e a capacidade da fila (GORINSKY e VIN, 2002). Assim, tem-se:

$$\begin{aligned}\text{Capacidade total da rede} &= \text{capacidade do enlace} + \text{capacidade da fila} \\ &= 2 * \text{largura de faixa} * \text{atraso} + \text{tamanho da fila.} \\ &= 2 * 5\,\text{Mbs} * 35\,\text{ms} + 31\,\text{pacotes} \\ &= 2 * \frac{5*1000*1000}{8*1400} * \frac{35}{1000} + 31 \\ &= 2 * 15.625 + 31 = 62.25\,\text{pacotes.}\end{aligned}$$

A rede foi simulada com agentes emissores do tipo TCP NewReno, TCP Westwood, TCP Westwood Equilíbrios Virtuais, e TCP Equilíbrios Virtuais variando as condições iniciais e dependendo do estudo a ser realizado.

Nas simulações foram calculados as seguintes grandezas (ou os seguintes valores de referência) para comparar o desempenho dos protocolos estudados:

- Desperdício: Capacidade máxima possível menos somatório do tamanho da janelas de congestionamento ao longo da simulação, medida em pacotes.

$$\sum_{t=1}^{tempo\_simulação} (cap - \sum_{i}^{n} cwnd_i)$$

- Transferência efetiva: somatório dos bytes enviados.
- Retransmissões: somatório dos bytes retransmitidos.
- *Goodput*: (Transferência efetiva – retransmissões) / tempo de simulação.

Cenário #1: Considera-se que o usuário TCP_1 inicia a transmissão em 0 segundo, o TCP_2 aos 15 segundos, o TCP_3 aos 30 segundos e o tempo total de simulação é de 60 segundos. Os três usuários utilizam agentes TCP_WVE e uma fila tipo DDropTail que implementa o contagem do número de usuários necessário para o protocolo correspondente (ver detalhes Seção 4.3.2).

Os valores utilizados para os parâmetros de TCP_WVE são: $b_I = 0.9$, $b_D = 0.9$, $aux1 = 1$, $aux2 = 0.5$. A Figura 5.7 mostra o resultado da simulação utilizando a

estimativa pura da implementação do protocolo TCP_W (equação 4.3) e a Figura 5.8 com uma sobre estimativa (equação 4.4) igual a $\varepsilon = 1.0019$. Comparando ambas as figuras, vemos que aplicando um fator (ε) à estimativa do TCP_W, se tem um melhor comportamento. Quando é utilizada a estimativa do TCP_W os tamanhos das janelas de congestionamento se estabilizam abaixo da capacidade da rede, que acaba sendo desperdiçada. Este comportamento é explicado quando a estimativa utilizada pelo TCP_W é para ajustar o Limiar da Partida Lenta (*ssthresh*) entre as fases de Partida Lenta e Prevenção de Congestionamento cada vez que acontece uma perda. Então, uma sobre estimativa suave pode assim dar uma medida mais adequada para ser utilizada por TCP_WVE.

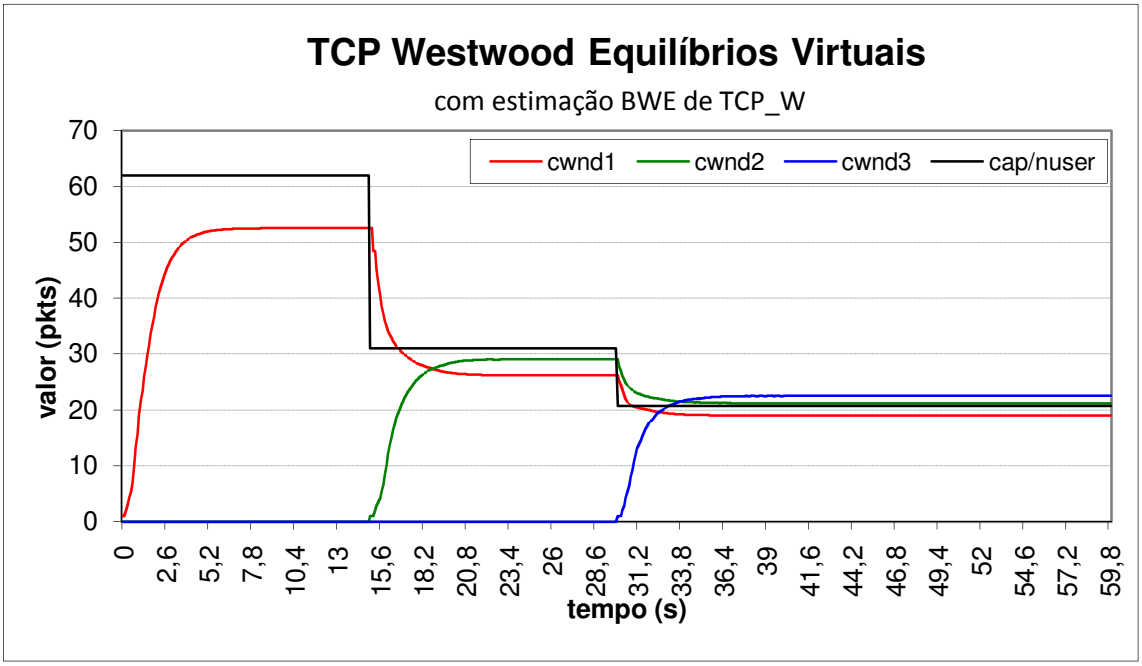


Figura 5.7: Simulação da rede no NS2 com TCP_WVE (sem sobre estimativa).

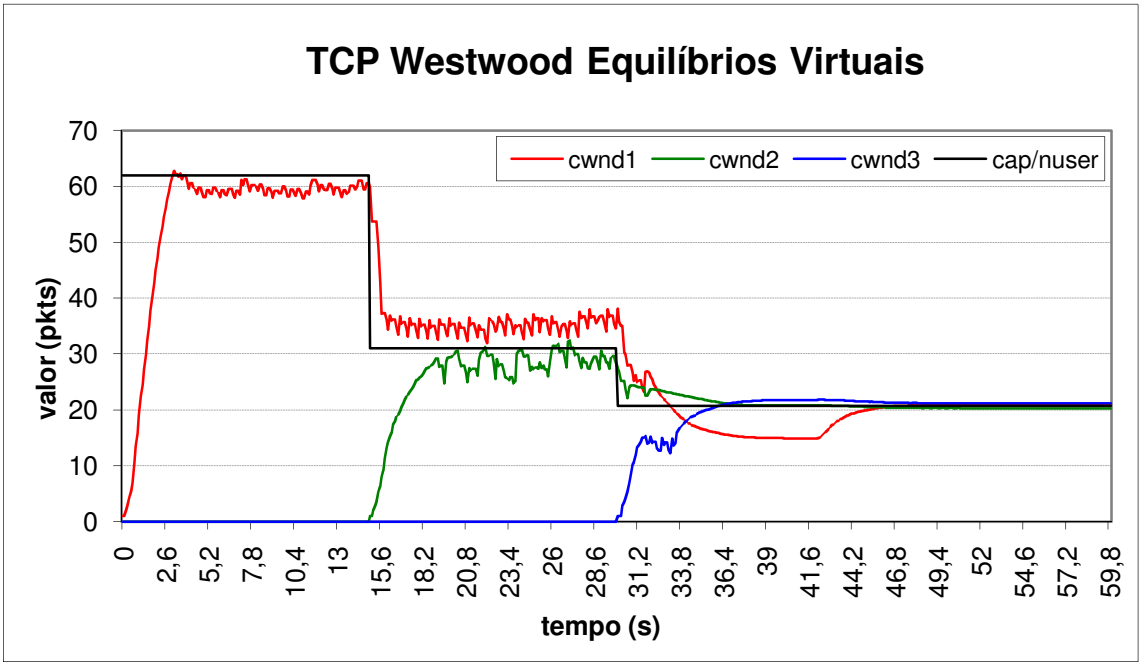


Figura 5.8: Simulação da rede no NS2 com TCP_WVE (com sobre estimativa $\varepsilon = 1.0019$).

No algoritmo Alg_WVE e sua correspondente implementação TCP_WVE, cada usuário utiliza sua própria estimativa de largura de banda para ajustar sua janela de congestionamento, como faz o TCP_W. Duas possíveis modificações desta técnica foram realizadas para verificar se o protocolo apresentava um melhor comportamento.

Na segunda versão do TCP_WVE chamada TCP_WVE_Máximo, cada usuário (emissor) faz o calculo de sua estimativa e envia o valor dentro do cabeçalho do pacote. Os roteadores procuram dentre os pacotes que chegam o máximo valor estimado da capacidade. Logo em seguida, os roteadores enviam esse valor para os receptores, que reenviam aos usuários. Quando um ACK chega ao emissor, este avalia qual é a maior capacidade dentre a recebida e a calculada naquele instante e utiliza este valor para calcular o tamanho da janela.

Na terceira versão, o TCP_WVE_Média, cada usuário (emissor) calcula sua estimativa e envia o valor dentro do cabeçalho do pacote. Os roteadores calculam, dentre os pacotes que chegam, a média entre os valores estimados de capacidade. Em seguida, este valor é enviado aos receptores, que reenviam aos usuários. Quando um ACK chega ao emissor, este utiliza o valor recebido para calcular o tamanho da janela.

Cenário #2: Considere três usuários iniciando a transmissão no instante 0 segundo. Quando comparados os valores de comparação de desempenho encontrados para os três casos de WVE tem se Tabela 5.1.

	Tabela 5.1: Resultado das simulações – Versões TCP_WVE			
	Desperdício (pkts)	**Trans. Efetiva (bytes)**	**Retransmissões (Bytes)**	***Goodput* (Bytes)**
TCP_WVE	1,001.179	9,658,720	2,006,000	384,558.793
TCP_WVE_Máximo	959.020	9,719,920	2,015,520	387,155.780
TCP_WVE_Média	953.076	9,669,600	1,997,840	385,515.570

Para o caso de três usuários que iniciam a transmissão ao mesmo tempo, os valores de referencia são similares. Isto pode ser confirmado observando as Figuras 5.9, 5.10, 5.11 e 5.12. Logo, não se justifica a utilização do máximo valor de largura de banda ou a média da mesma calculada pelos roteadores e utilizada pelos usuários para o seguinte cálculo do tamanho da janela de congestionamento.

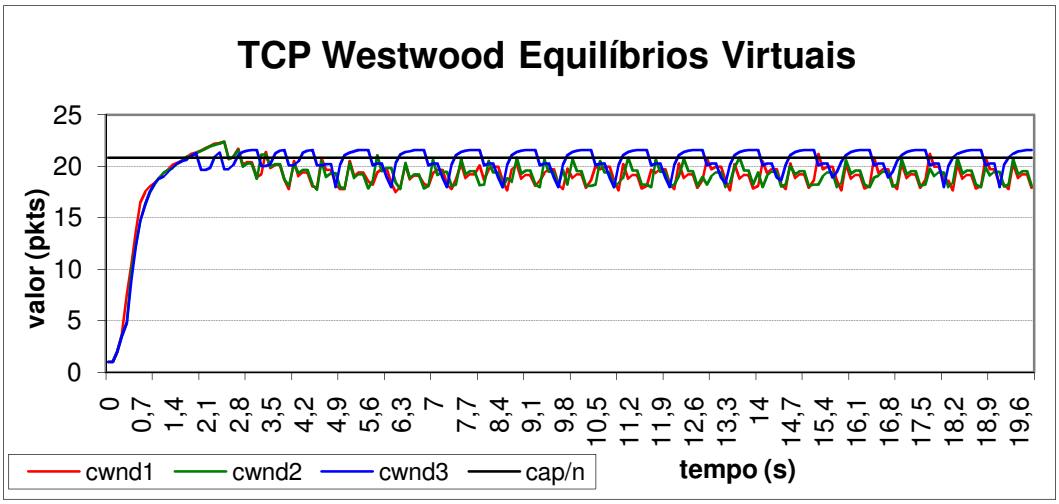


Figura 5.9: Resultados da simulação no NS2 com agentes TCP_WVE.

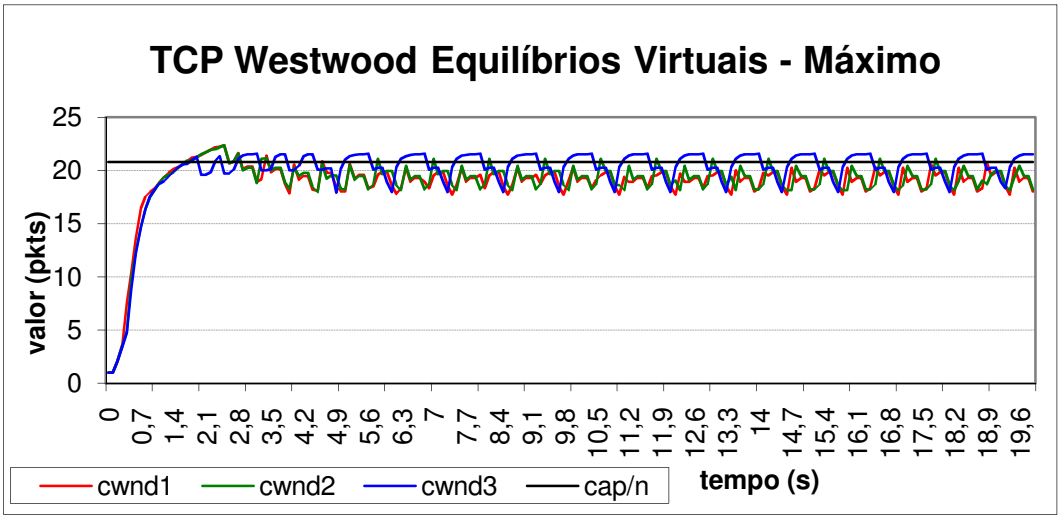


Figura 5.10: Resultados da simulação no NS2 com agentes TCP_WVE_Máximo.

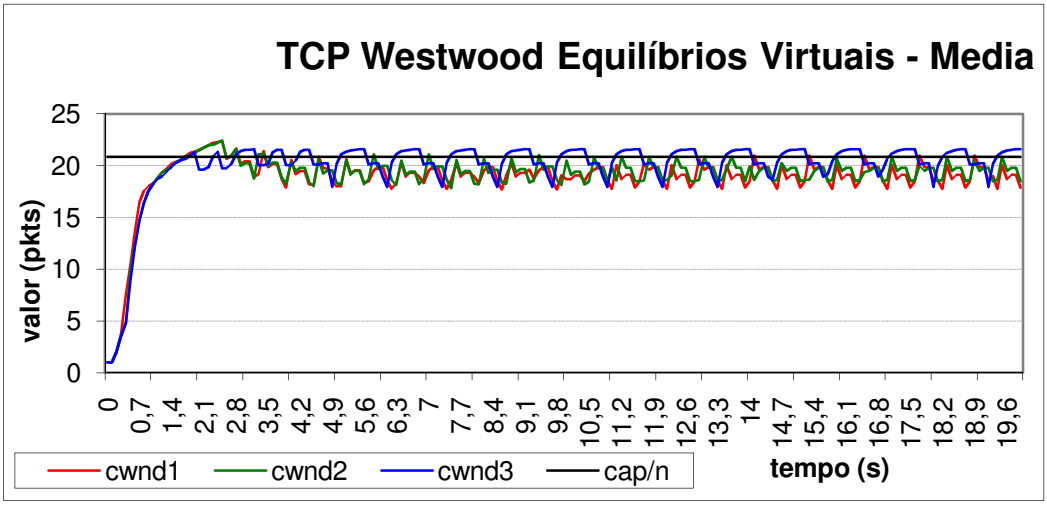


Figura 5.11: Resultados da simulação com agentes TCP_WVE_Média.

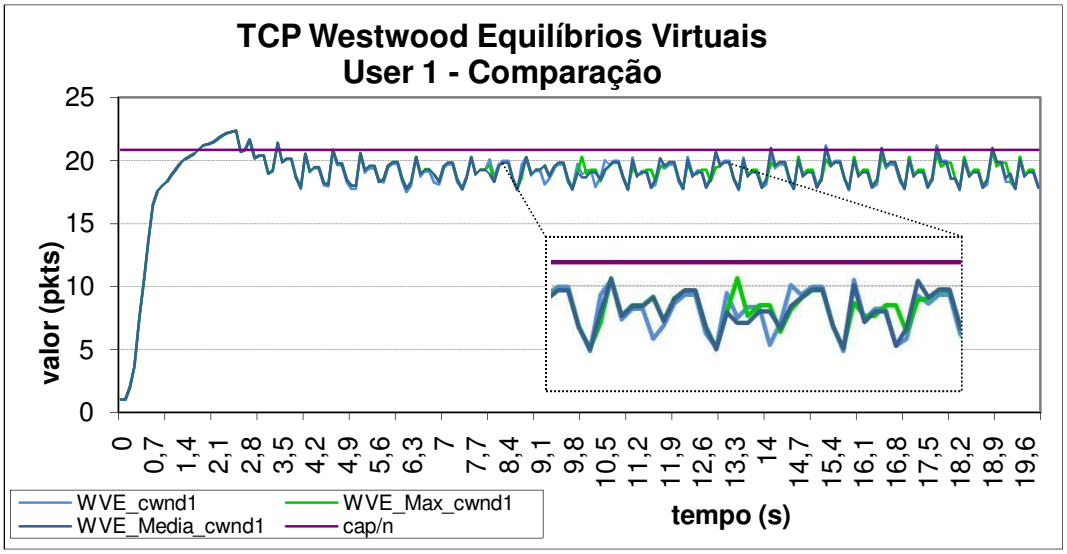


Figura 5.12: Comparação de resultados da simulação no NS2 para o usuário 1 – TCP_WVE vs. TCP_WVE_Máximo vs. TCP_WVE_Média.

Cenário #1: Consideram-se três usuários transmitindo no instante 0 segundo utilizando agentes TCP do tipo TCP New Reno (NR), TCP Westwood (WP), TCP Westwod Equilíbrios Virtuais (WVE) e Equilíbrios Virtuais (VE).

As Figuras 5.13 – 5.16 mostram o resultado da simulação dos protocolos estudados. Observando a evolução do tamanho da janela de congestionamento pode-se constatar que o TCP New Reno tem aquela instabilidade inicial devido à fase de Partida Lenta. Depois, apresenta um bom comportamento desperdiçando a capacidade da rede. Para o cenário ilustrado TCP Westwood não tem um bom comportamento. Os protocolos propostos TCP Westwood Equilíbrios Virtuais e TCP Equilíbrios Virtuais mostram, apesar de tudo, um bom comportamento ajustando a janela de congestionamento ao valor médio disponível para cada usuário.

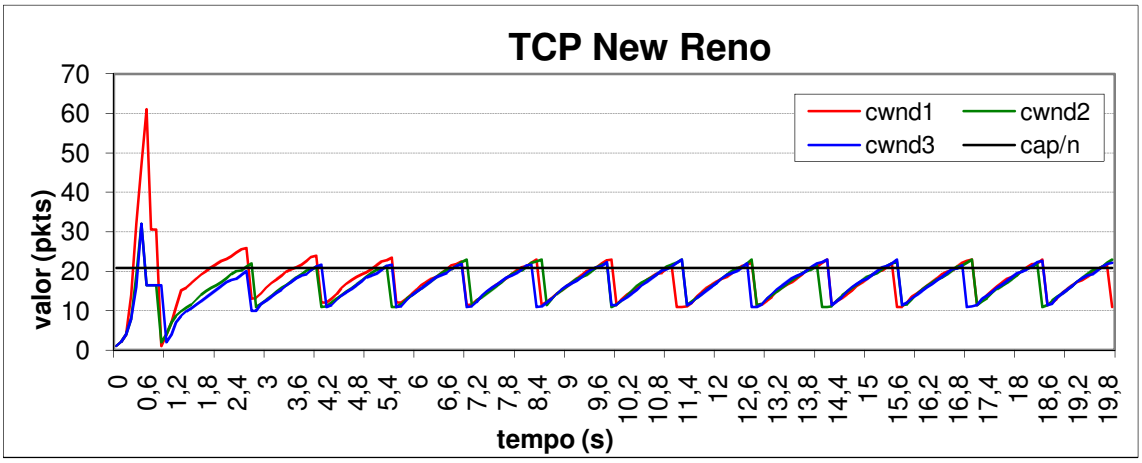


Figura 5.13: Simulação TCP New Reno no NS2 para três usuários transmitindo no inicio da simulação – Evolução da janela de congestionamento no tempo.

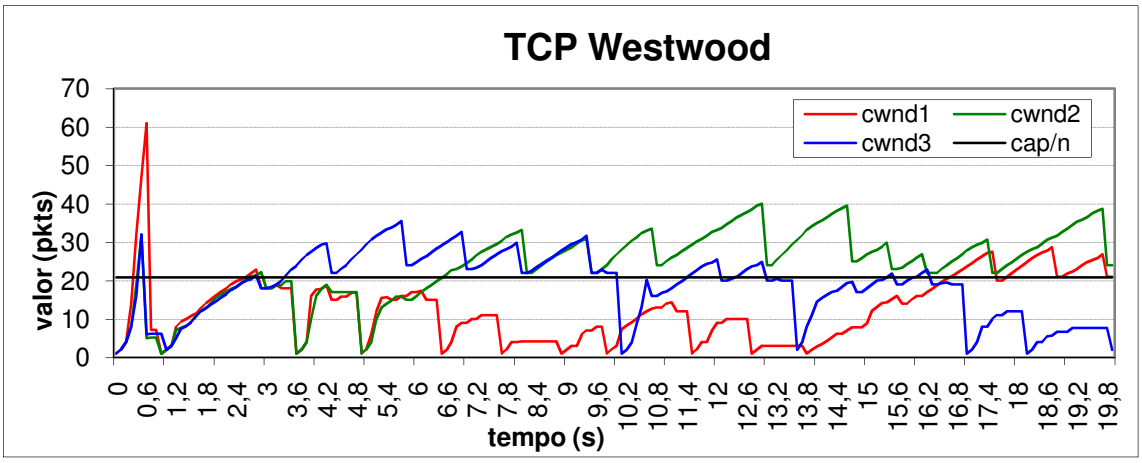


Figura 5.14: Simulação TCP Westwood no NS2 para três usuários transmitindo no inicio da simulação – Evolução da janela de congestionamento no tempo.

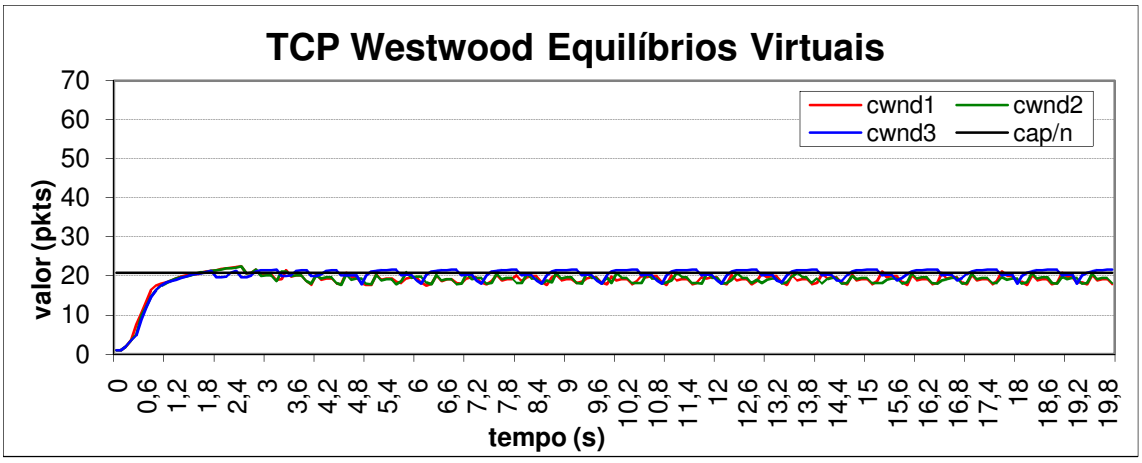


Figura 5.15: Simulação TCP Westwood Equilíbrios Virtuais no NS2 para três usuários transmitindo no inicio da simulação – Evolução da janela de congestionamento no tempo.

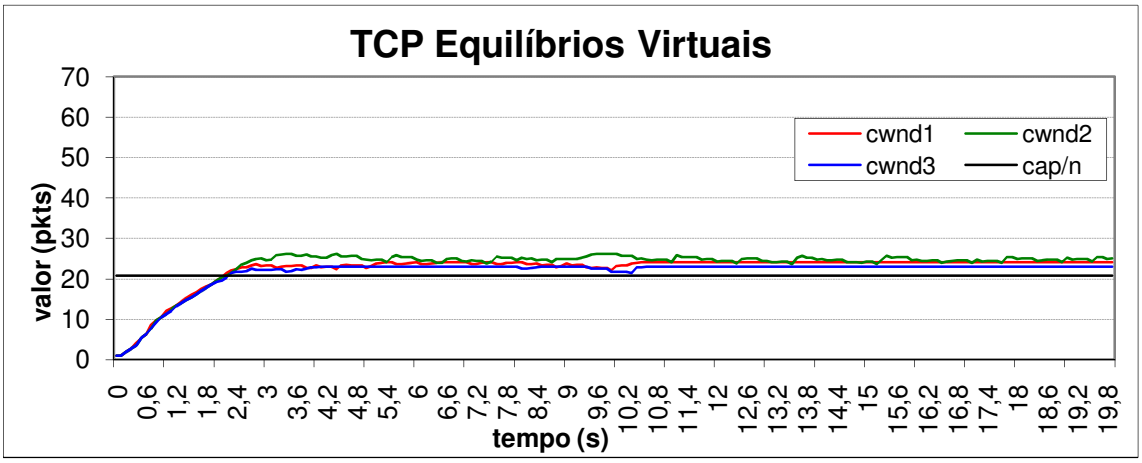


Figura 5.16: Simulação TCP Equilíbrios Virtuais no NS2 para três usuários transmitindo no inicio da simulação – Evolução da janela de congestionamento no tempo.

Os protocolos TCP_WVE e TCP_VE estão mais perto à utilização efetiva da largura de banda disponível (isto pode observar se ao ver que as curvas dos fluxos estão muito próximas ao valor equitativo). As Figuras 5.17 – 5.20 apresentam as diferenças entre a capacidade máxima da rede e a capacidade alcançada pelos protocolos TCP_NR, TCP_W, TCP_WVE e TCP_VE, respectivamente. Os valores obtidos das áreas (desperdício) são: TCP_NR: 2394.26 pkts, TCP_W: 1521.16 pkts, TCP_WVE: 1001.18 pkts, e TCP_VE: -949 pkts[11]. Assim, o TCP_VE tem um menor desperdício, pois a capacidade do canal é utilizada ao máximo todo o tempo. Contrariamente, o TCP_NR tem o maior desperdício dentre os protocolos estudados.

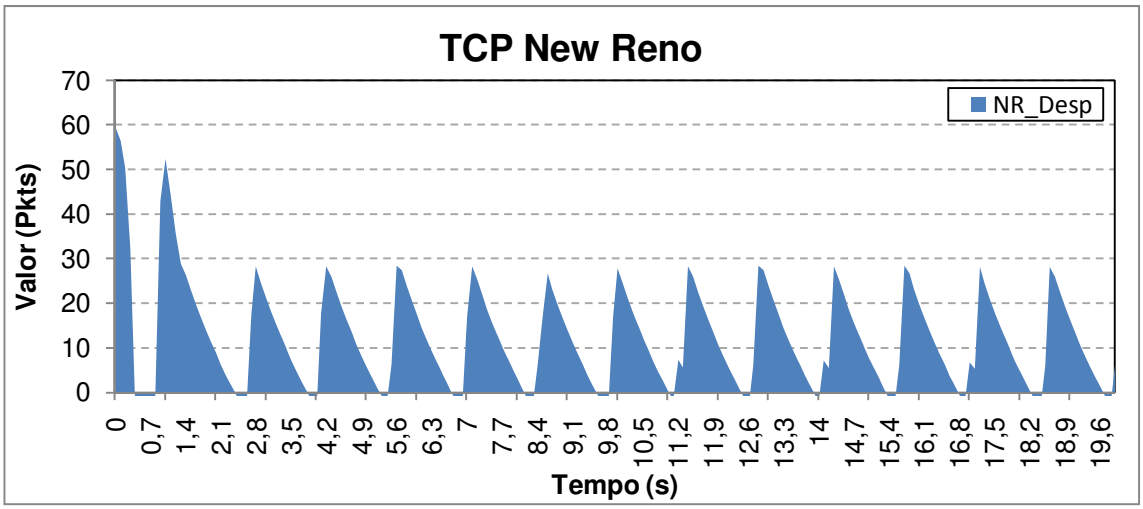


Figura 5.17: Capacidade não utilizada - Simulação no NS2 do TCP New Reno

[11]: O símbolo '- ' indica aos pacotes perdidos porque a transmissão excede a capacidade máxima da rede.

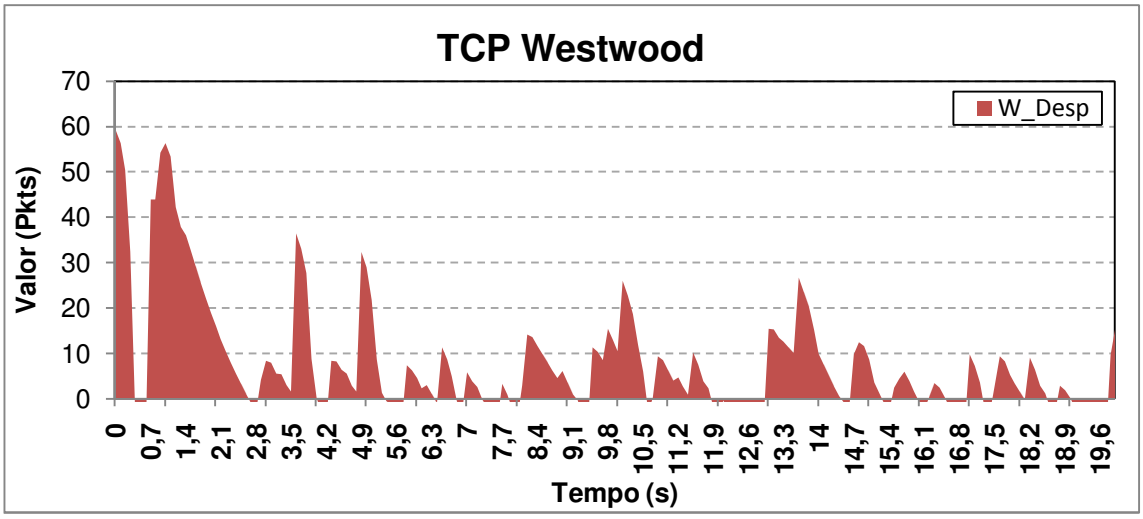


Figura 5.18: Capacidade não utilizada - Simulação no NS2 do TCP Westwood.

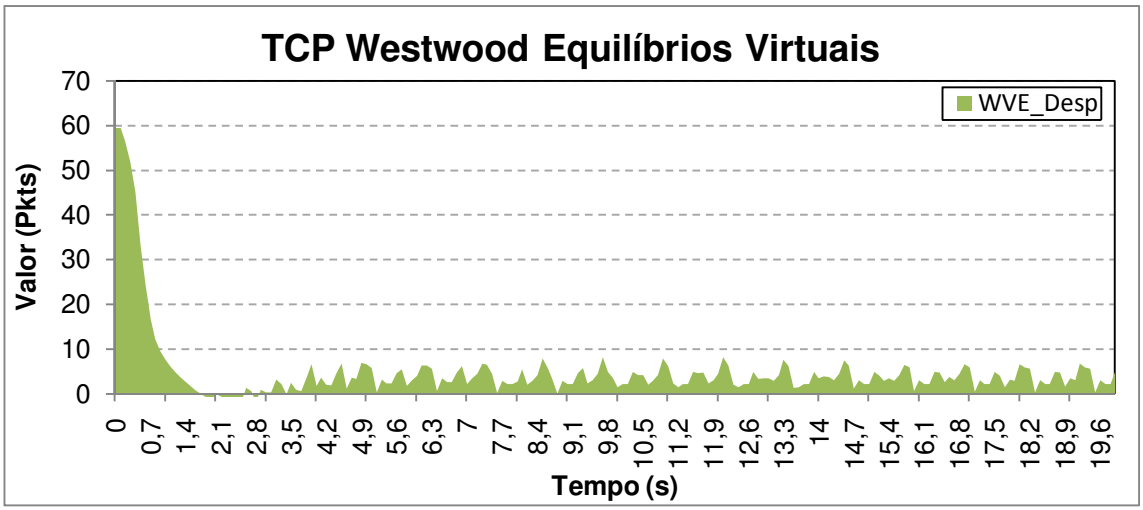


Figura 5.19: Capacidade não utilizada - Simulação no NS2 do TCP Westwood Equilíbrios Virtuais.

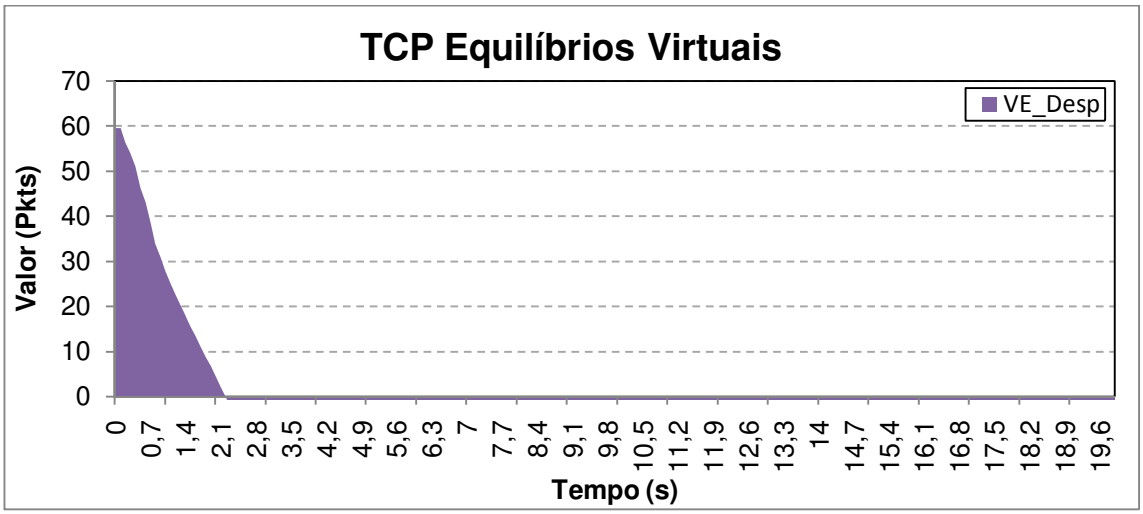


Figura 5.20: Capacidade não utilizada - Simulação no NS2 do TCP Equilíbrios Virtuais.

Se comparados as grandezas de evacuação de desempenho (transmissão efetiva, retransmissões e *goodput*) para os quatro protocolos estudados obtém se a Tabela 5.2 que resume as medições encontradas em cada caso.

	Tabela 5.2: Resultado das simulações de 3 usuários transmitindo ao instante 0s. (20 segundos)		
	Trans. Efetiva (BYTES)[12]	**Retransmissões (BYTES)**	**Goodput**
TCP_NR	9,798,800.00	47,600.00	**605,664.60**
TCP_WP	9,908,960.00	243,440.00	578,773.65
TCP_WVE	8,761,120.00	2,014,160.00	372,760.22
TCP_VE	**11,132,960.00**	1,260,720.00	557,753.67

Considerando a quantidade de dados enviados (transferência efetiva), pode-se perceber que o TCP_VE se comporta melhor, já que envia uma maior quantidade que os outros. Isto pode verificar se na Figura 5.21.

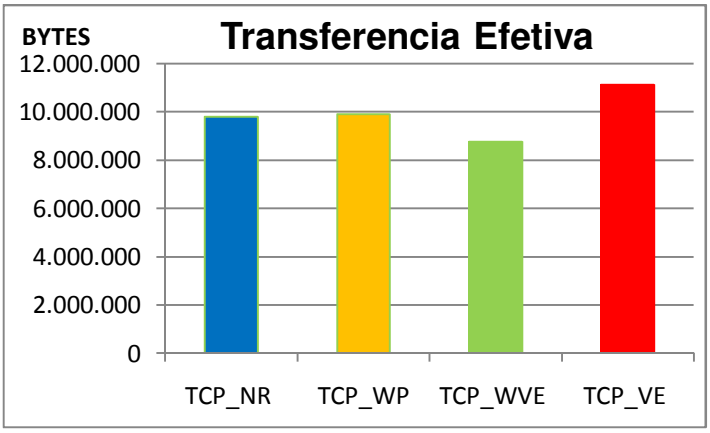


Figura 5.21: Comparacao de transferencia efectiva dos protocolos estudados para para três usuários transmitindo no inicio da simulação.

Entretanto, considerando a métrica do *Goodput*, o TCP_NR se comporta melhor já, que o número de retransmissões é menor que o número nos demais protocolos. O TCP_NR se comporta desta maneira devido à forma em que estabelece o limiar entre Partida Lenta/Prevenção de Congestionamento, o que faz com que as transmissões sejam por menores que a capacidade da rede, penalizando assim a eficiência.

[12] Para o cálculo das grandezas de comparação de desempenho foi truncado o transitório ou *warm-up* da simulação (JAIN, 1991).

Considerando o mesmo cenário a Tabela 5.3 mostra os resultados em media de varias rodadas (obtidos variando a semente de aleatoriedade do simulador NS2) por cada protocolo estudado. De novo aqui em media a transferência efetiva do TCP_VE é maior que os outros protocolos; em media ele envia uma maior quantidade de dados no tempo de simulação sendo o *goodput* no mesmo grado de grandeza que o TCP_NR.

	Tabela 5.3: Media de Simulações de 3 usuários transmitindo ao instante 0s. (20 segundos)		
	Media de Trans. Efetiva (BYTES)	**Media de Retransmissões (BYTES)**	**Media de Goodput**
TCP_NR	11,928,560.00	231,219.80	**587,806.04**
TCP_W	11,410,128.00	431,112.00	551,709.35
TCP_WVE	10,709,560.00	903,720.00	492,755.78
TCP_VE	**12,111,616.00**	663,000.00	575,307.34

OBS: ver detalhe no apêndice

Analisa-se agora qual dos protocolos estudados envia uma quantidade de dados em forma mais rápida. Para este estudo foi estabelecido um valor do tempo para cada 100 pacotes. A Tabela 5.4 apresenta os bytes enviados por cada usuário durante os 20 segundos da simulação.

	Tabela 5.4: Quantidade de dados enviados durante simulação (Bytes)			
	TCP_NR	**TCP_W**	**TCP_WVE**	**TCP_VE**
TCP1	4,080,000	2,312,000	2,584,000	4,624,000
TCP2	3,944,000	5,304,000	2,584,000	2,720,000
TCP3	3,808,000	3,808,000	4,352,000	4,760,000
soma	11,832,000	11,424,000	9,520,000	**12,104,000**
média	3,944,000	3,808,000	3,173,333	**4,034,667**

Como os usuários TCPs não transmitem a mesma quantidade de pacotes (bytes) no tempo total de simulação, para fazer uma comparação mais adequada considerou-se a transmissão até o menor valor alcançado por algum TCPs (ver detalhe dos dados no apêndice). Assim temos as seguintes figuras comparativas para cada usuário.

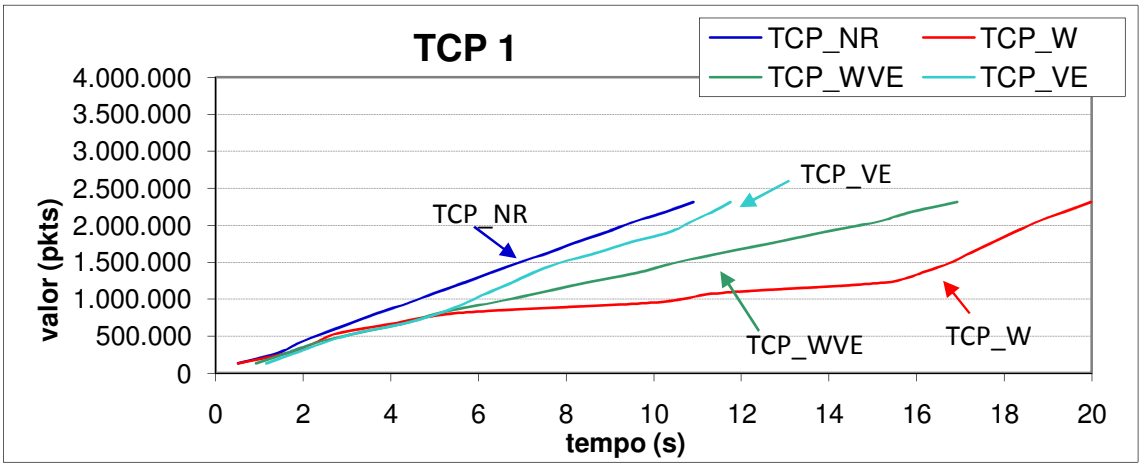


Figura 5.22: Transmissão de bytes do usuário TCP 1 no tempo – Amostra para cada 100 pacotes obtida da simulação no NS2.

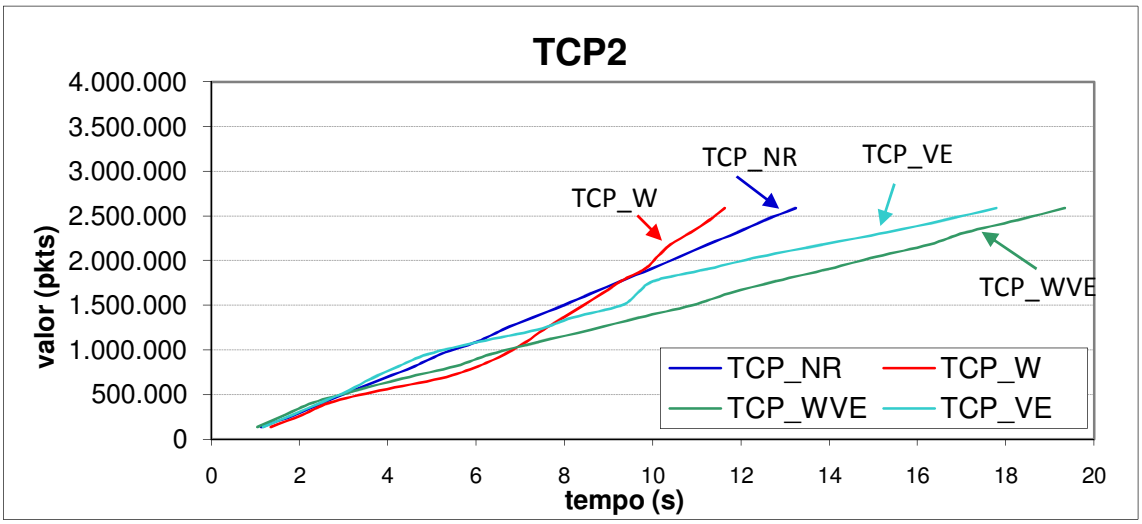


Figura 5.23: Transmissão de bytes do usuário TCP 2 no tempo – Amostra para cada 100 pacotes obtida da simulação no NS2.

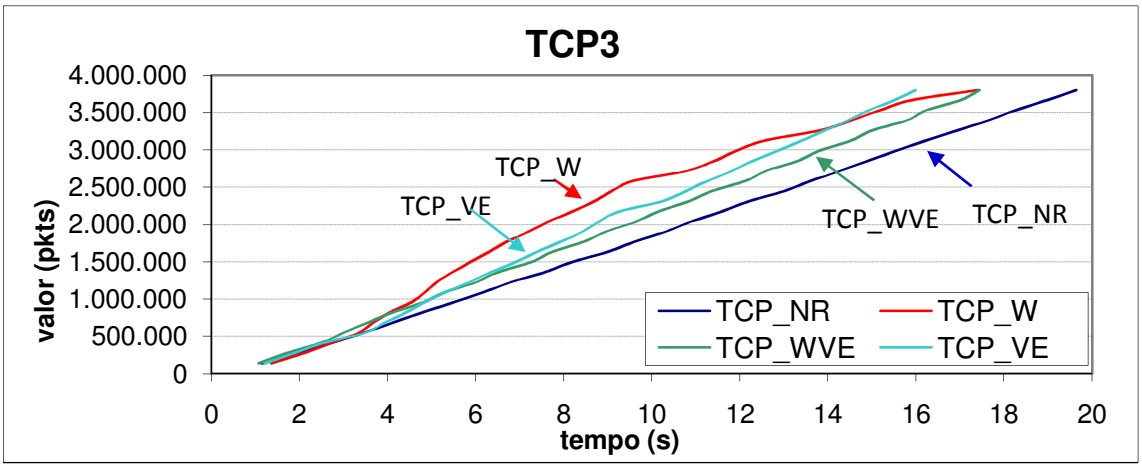


Figura 5.24: Transmissão de bytes do usuário TCP 2 no tempo – Amostra para cada 100 pacotes obtida da simulação no NS2.

Comparando os protocolos das Figuras 5.22, 5.23, e 5.24 pode-se observar que no caso do usuário TCP_1 o protocolo TCP_NR envia os 2,312,000 bytes (1,700 pacotes) em menor tempo que os outros, cerca de 10.91 segundos; para o usuário TCP_2 o protocolo TCP_W envia 2,584,000 (1,900 pacotes) em 11.64 segundos; e o usuário TCP_3 envia 3,808,000 (2,800 pacotes) em 15.99 segundos. Assim dependendo do usuário, alguns protocolos são melhores que outros, não sendo possível concluir qual é o melhor protocolo para os três usuários.

Em média, o protocolo TCP_VE envia mais dados que os demais protocolos, conforme citado anteriormente. Se, por exemplo, a quantidade de dados enviados e o tempo total de simulação forem utilizados, o tempo de espera de cada protocolo para transmitir 10,000,000 bytes seria; $t = 16.90$ segundos para o TCP_NR, $t = 17.50$ segundos para o TCP_W, $t = 21.00$ segundos para o TCP_WVE, e $t = 16.52$ segundos para o TCP_VE. Portanto pode-se concluir que o protocolo TCP_VE envia 10,000,000 bytes mais rapidamente que os outros protocolos.

Cenário #4: Considera-se novamente os três usuários que transmitem em tempos diferentes. O usuário TCP_1 inicia a transmissão no instante 0 segundo, o usuário TCP_2 no instante 15 segundos e o usuário TCP_3 no instante 27 segundos, com um tempo total de simulação de 60 s. Para o TCP New Reno e Westwood é utilizada uma fila do tipo Drop Tail tradicional. No entanto, para o caso de TCP Westwood Equilíbrios Virtuais e TCP Equilíbrios Virtuais a fila é do tipo DDropTail. As Figuras 5.25 – 5.28 mostram a dinâmica do tamanho da janela (*cwnd*) de congestionamento para os três usuários, assim como uma média móvel tomada de cinco amostras consecutivas do tamanho da janela e a taxa de transmissão ideal igual à capacidade da rede dividida pelo número de usuários que se encontram transmitindo nesse momento.

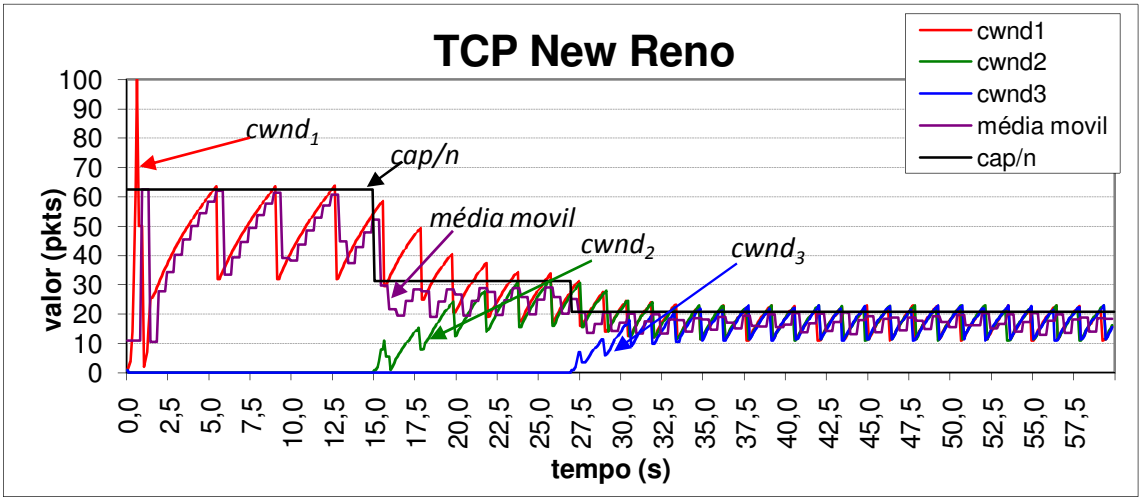


Figura 5.25: Simulação TCP New Reno no NS2 para três usuários transmitindo em tempos deferentes – Evolução da janela de congestionamento em função do tempo.

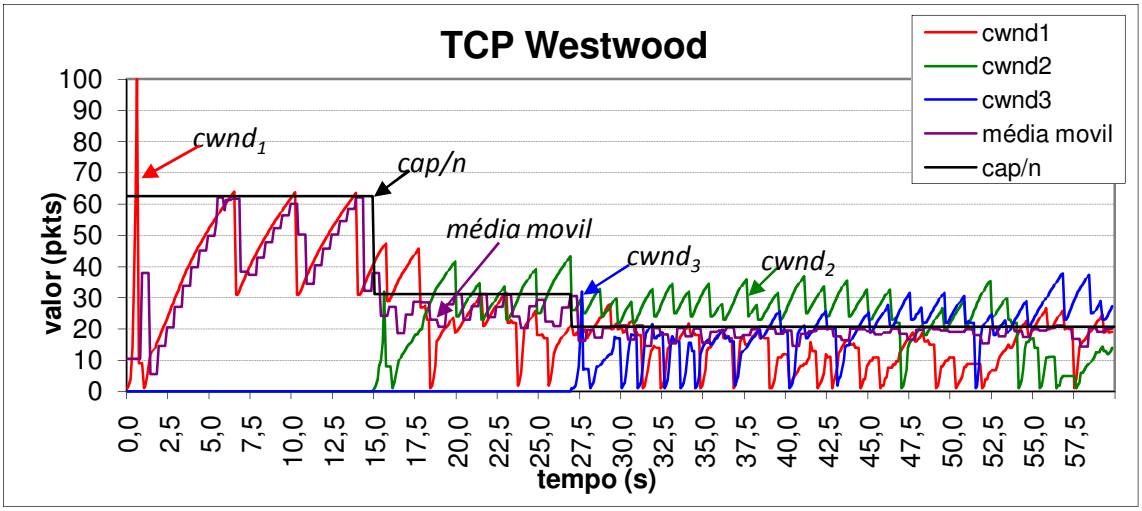


Figura 5.26: Simulação TCP Westwood no NS2 para três usuários transmitindo em tempos deferentes – Evolução da janela de congestionamento em função do tempo.

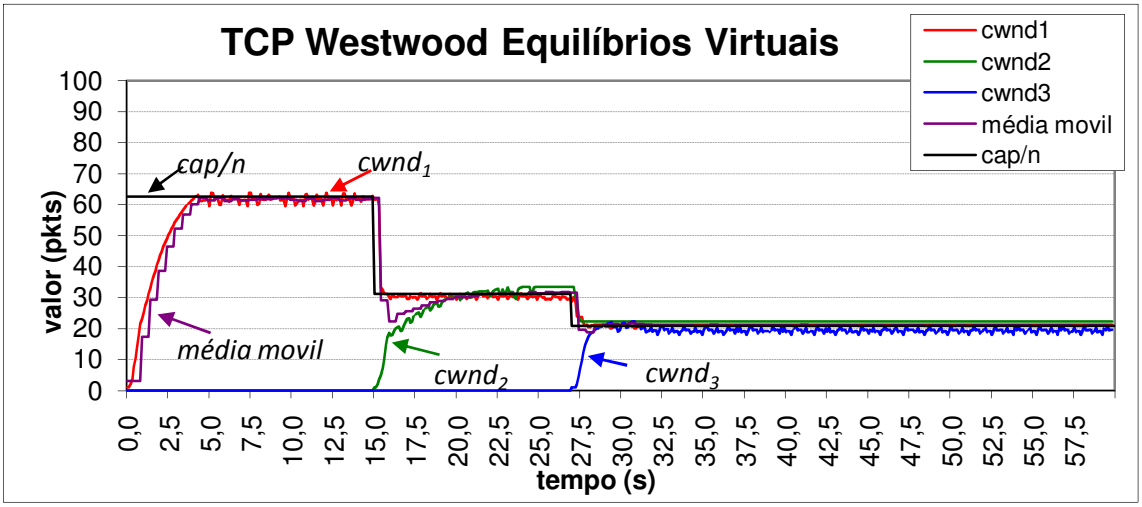


Figura 5.27: Simulação TCP Westwood Equilíbrios Virtuais no NS2 para três usuários transmitindo em tempos deferentes – Evolução da janela de congestionamento em função do tempo.

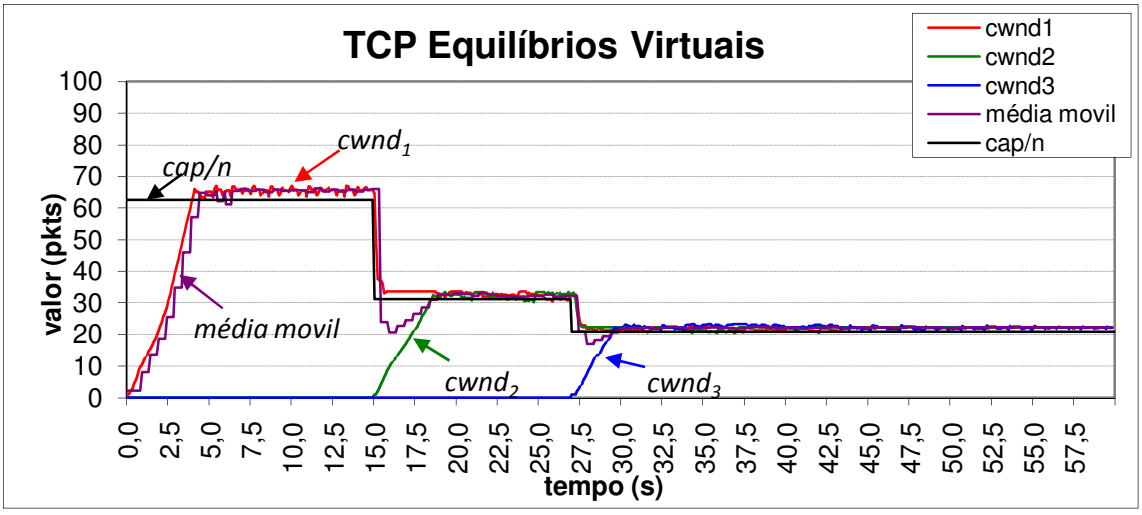


Figura 5.28: Simulação TCP Equilíbrios Virtuais no NS2 para três usuários transmitindo em tempos deferentes – Evolução da janela de congestionamento em função do tempo.

Os resultados experimentais dos protocolos TCP Westwood Equilíbrios Virtuais e TCP Equilíbrios Virtuais para este cenário apresentam um comportamento mais próximo à taxa de transmissão eficiente e equitativa especialmente se a taxa cap/n e a média móvel calculadas forem comparadas (Figuras 5.29 e 5.30).

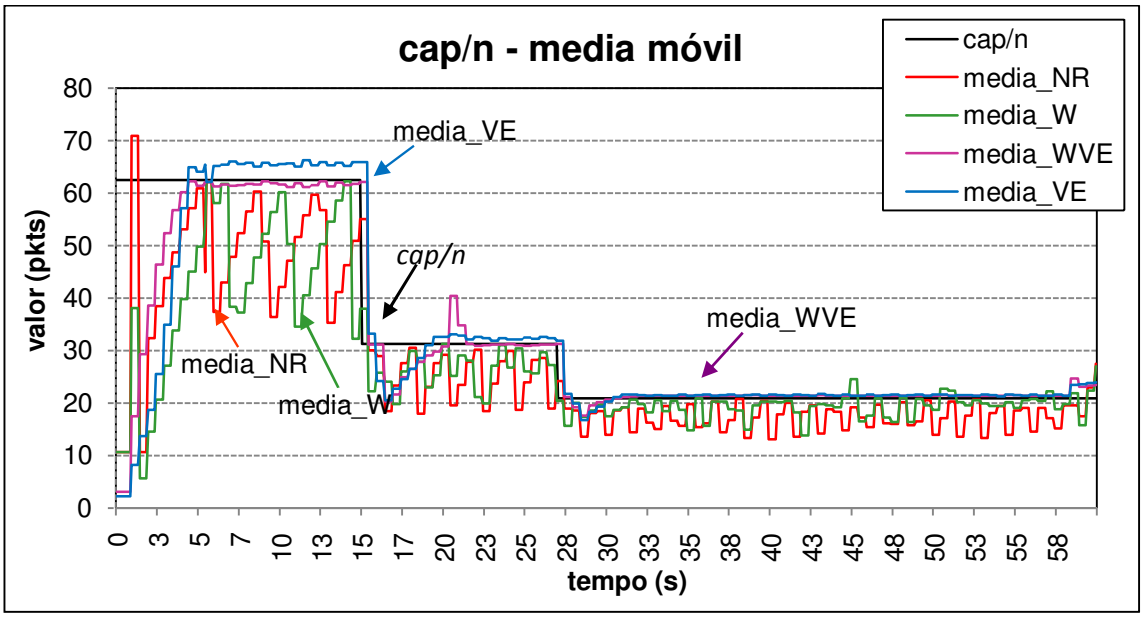


Figura 5.29: Média móvel vs. cap/n - Simulação no NS2 para três usuários transmitindo em tempos deferentes.

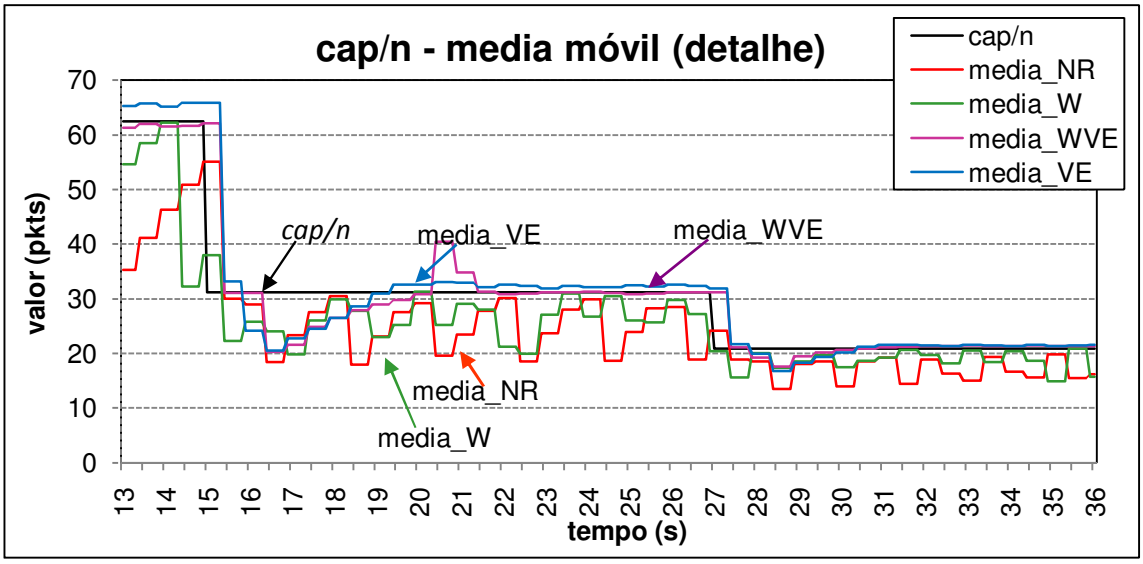


Figura 5.30: Detalhe média movil vs. cap/n – Simulação no NS2 para três usuários transmitindo em tempos diferentes.

Através da Figura 5.30 pode-se identificar o momento que os usuários 2 e 3 começam a transmitir. Os protocolos TCP Westwood Equilíbrios Virtuais e TCP Equilíbrios Virtuais reagem mais rapidamente ao aumento do número de usuários, apresentando um melhor tempo de resposta que os demais protocolos. Também, como já fora mencionando, o TCP_WVE e TCP_VE têm uma menor amplitude de oscilações.

A Figura 5.31 mostra o resultado medido do índice de equidade para os primeiros 5 segundos da simulação. Pode conferir se que os protocolos TCP_WVE e TCP_VE tem um comportamento mais justo para este cenário. Também, no cenário estudado, e possível verificar que a rede converge à justiça mais rapidamente nos dois protocolos propostos que no TCP_NR.

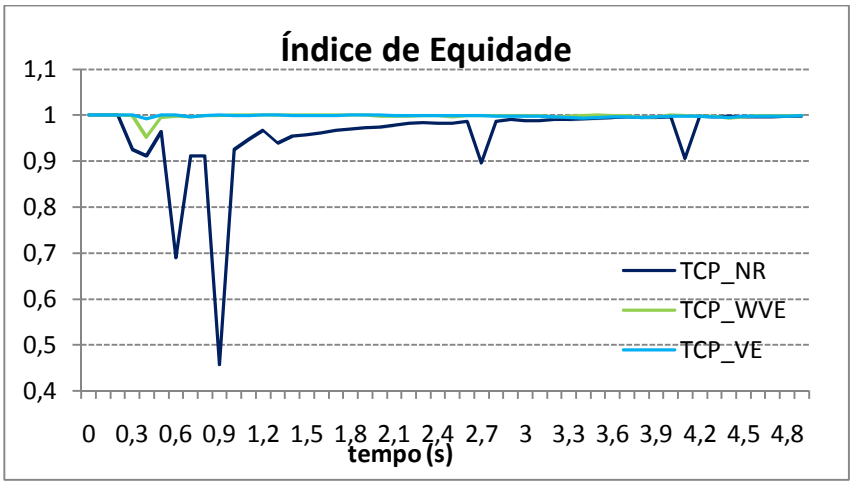


Figura 5.31: Índice de equidade medido na simulação do cenario #4

Para avaliar a transferência efetiva de dados entre os usuários deste cenário, se apresenta a Tabela 5.5 que apresenta a media de resultados obtidos de diversas rodadas variando o inicio da simulação. O TCP_VE, como pouca diferencia do TCP_WVE, envia mais dados em media para o estudo realizado. Neste caso o TCP_WVE apresenta melhor *goodput*.

	Tabela 5.5: Media de Simulações de 3 usuários. Inicio aleatorio (20 segundos)		
	Media de Trans. Efetiva (BYTES)	**Media de Retransmissões (BYTES)**	**Media de Goodput**
TCP_NR	9,076,622.50	239,530.00	444,075.00
TCP_WVE	11,185,640.00	758,880.00	**523,957.79**
TCP_VE	**11,248,820.00**	1,004,485.00	514,790.70

OBS: ver detalhe no apêndice

Cenário #5: Considera-se o cenário #4 com o tempo de simulação é estendido para 70 segundos. Considere que o segundo usuário para de transmitir aos 58 segundos.

Espera-se que o protocolo de controle de congestionamento reaja o mais rapidamente possível à redução do número de usuários incrementando aumentando a taxa de transmissão dos usuários restantes. Comparando as Figuras 5.32 – 5.35 pode observar-se que tanto o protocolo TCP_WVE como o TCP_VE reagem mais rapidamente aos câmbios no número de usuários que o TCP_NR e TCP_W.

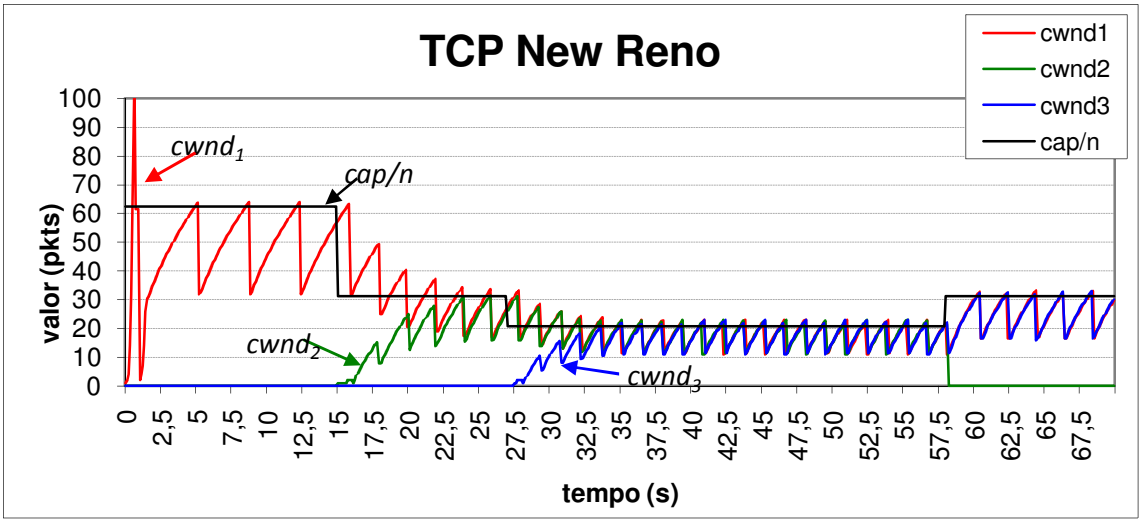


Figura 5.32: Simulação TCP New Reno no NS2 para três usuários transmitindo em tempos diferentes – Usuário 2 para de transmitir os 58s. - Evolução da janela de congestionamento em função do tempo.

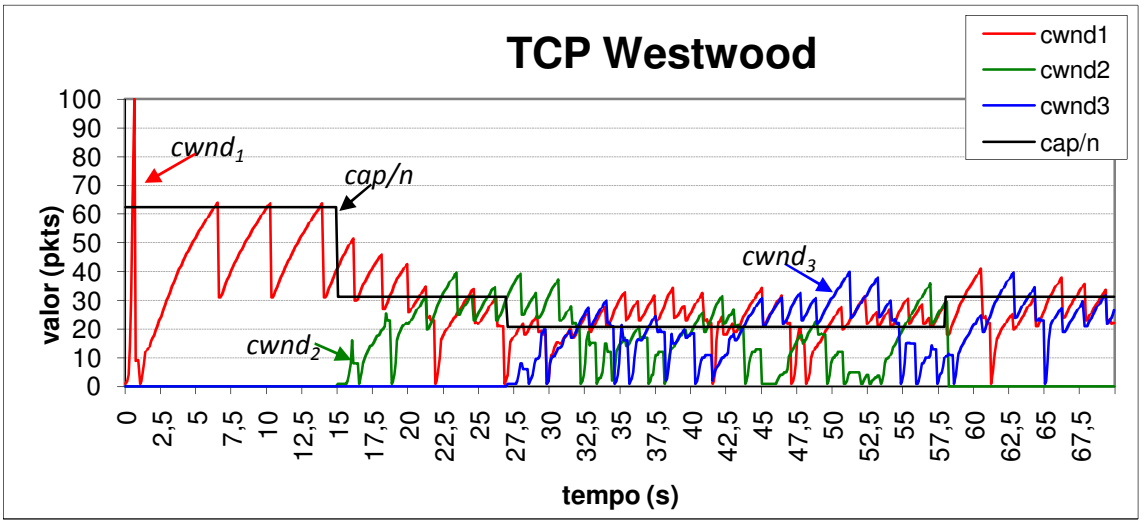


Figura 5.33: Simulação TCP Westwood no NS2 para três usuários transmitindo em tempos diferentes – Usuário 2 para de transmitir os 58s. - Evolução da janela de congestionamento em função do tempo.

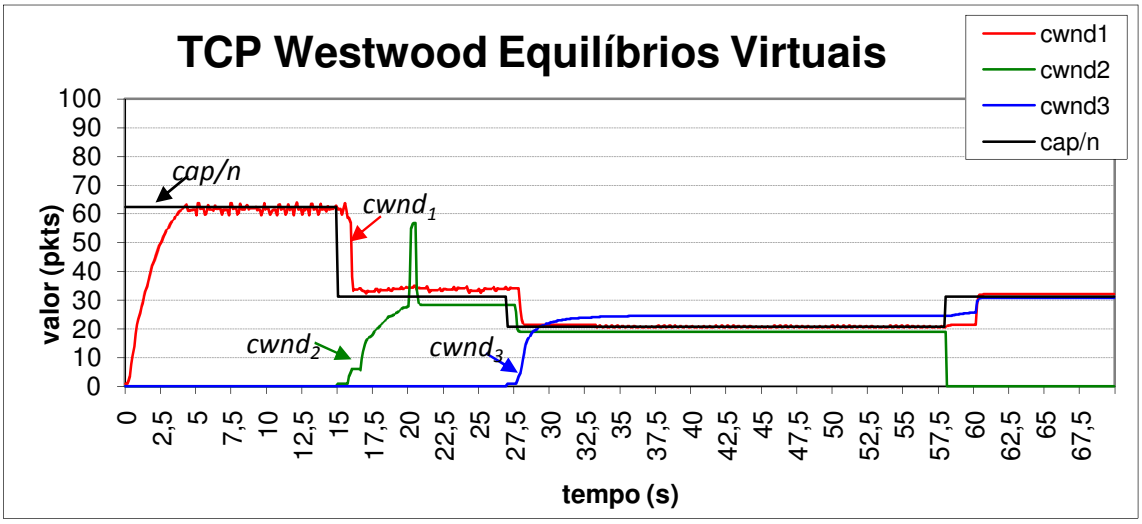


Figura 5.34: Simulação TCP Westwood Equilíbrios Virtuais no NS2 para três usuários transmitindo em tempos diferentes – Usuário 2 para de transmitir os 58s. - Evolução da janela de congestionamento em função do tempo.

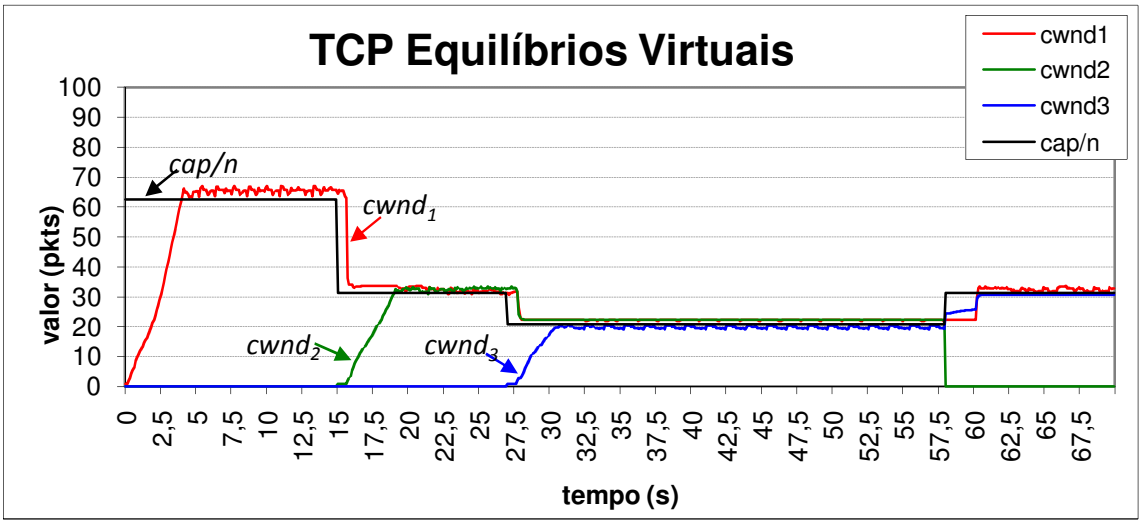


Figura 5.35: Simulação TCP Equilíbrios Virtuais no NS2 para três usuários transmitindo em tempos diferentes – Usuário 2 para de transmitir os 58s. - Evolução da janela de congestionamento em função do tempo.

A Compatibilidade é outra característica desejável para um protocolo de controle de congestionamento, ou seja, ele deve ser compatível com outros protocolos do tipo TCP existentes na Internet. Isto significa que TCP Westwood Equilíbrios Virtuais e TCP Equilíbrios Virtuais devem ser capazes de coexistir com conexões de variantes de TCP, além de possibilitar que todas as conexões progridam satisfatoriamente. As conexões TCP_WVE ou TCP_VE não devem conduzir à inanição (quando algumas conexões possuem acesso ao serviço e outras não) das conexões que funcionam com outras variações de TCP.

Cenário #6.a: Considere três usuários transmitindo: o usuário 1 transmite seus dados utilizando um controle do tipo TCP Westwood Equilíbrios Virtuais no instante 0 segundo, e os usuários 2 e 3 utilizam TCP New Reno aos 15 e 27 segundos respectivamente com um tempo de simulação de 50 segundos.

Cenário #6.b: Considere três usuários transmitindo: o usuário 1 transmite seus dados utilizando um controle do tipo TCP Equilíbrios Virtuais no instante 0 segundo, e os usuários 2 e 3 utilizam TCP New Reno aos 15 e 27 segundos respectivamente com um tempo de simulação de 50 segundos.

No cenário 6 as transmissões do tipo TCP New Reno não são prejudicadas com as transmissões que utilizam os protocolos propostos.

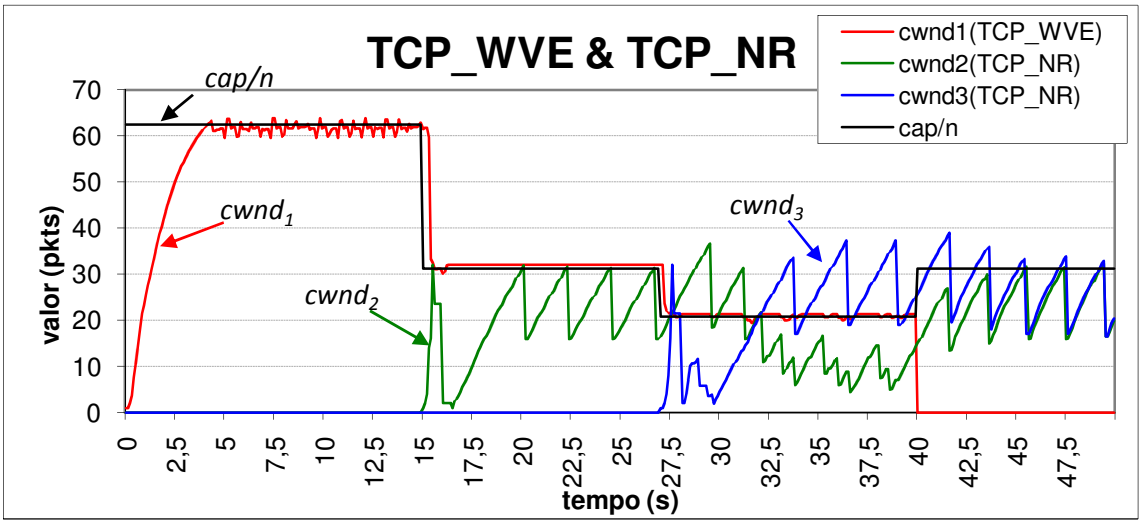


Figura 5.36 Compatibilidade TCP_WVE com TCP_NR – Cenário 6.a – Simulação no NS2.

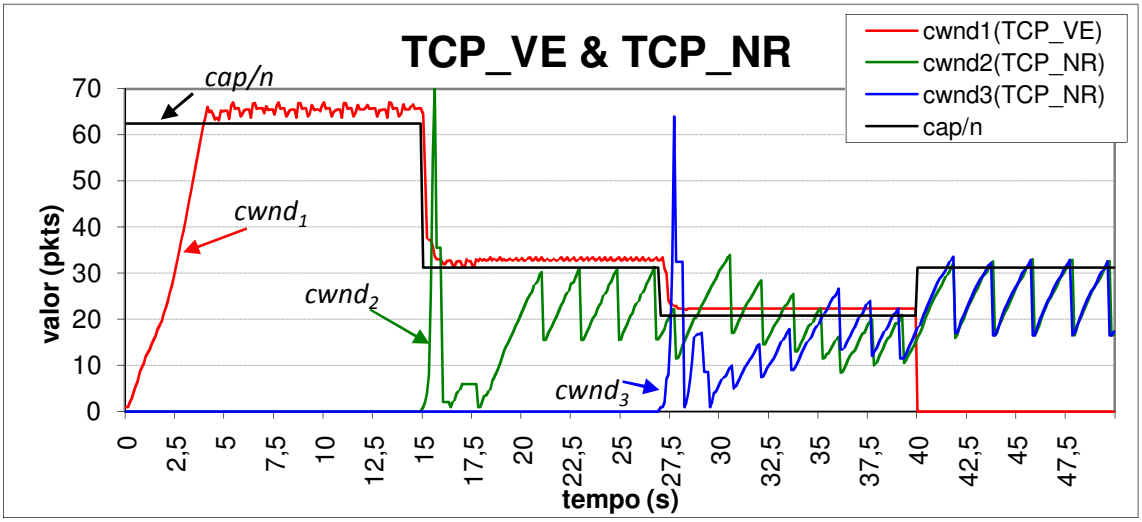


Figura 5.37: Compabilidade TCP_VE com TCP_NR – Cenário 6.b – Simulação no NS2.

Como pode verse nas Figuras 5.36 e 5.37. Para este cenário, tanto no caso do TCP_WVE, quanto no caso do TCP_VE, o comportamento do tamanho da janela de congestionamento se aproxima muito do valor ideal cap/n. No entanto, comparando ambas as figuras o TCP_WVE apresenta melhores resultados. Porém, nem sempre a compatibilidade resulta em bons resultados como veremos a continuação.

Cenário #7.a: Considere três usuários transmitindo: o usuário 1 transmite seus dados utilizando um controle do tipo TCP New Reno no instante 0 segundo, o usuário 2 com controle do tipo TCP Westwood Equilíbrios Virtuais aos 15 segundos e o usuário 3 como controle do tipo TCP New Reno aos 27 segundos.

Cenário #7.b: Considere três usuários transmitindo: o usuário 1 transmite seus dados utilizando um controle do tipo TCP New Reno no instante 0 segundo, o usuário 2 com controle do tipo TCP Equilíbrios Virtuais aos 15 segundos e o usuário 3 como controle do tipo TCP New Reno aos 27 segundos.

As Figuras 5.38 e 5.39 demonstram que o TCP_WVE é mais compatível que o TCP_VE.

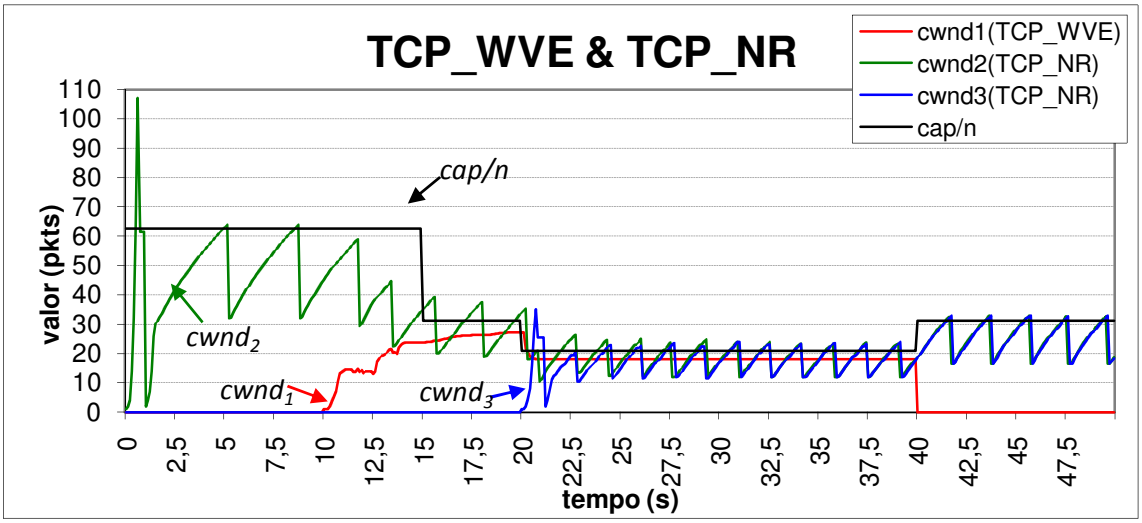


Figura 5.38: Compatibilidade TCP_WVE com TCP_NR – Cenário 7.a – Simulação no NS2.

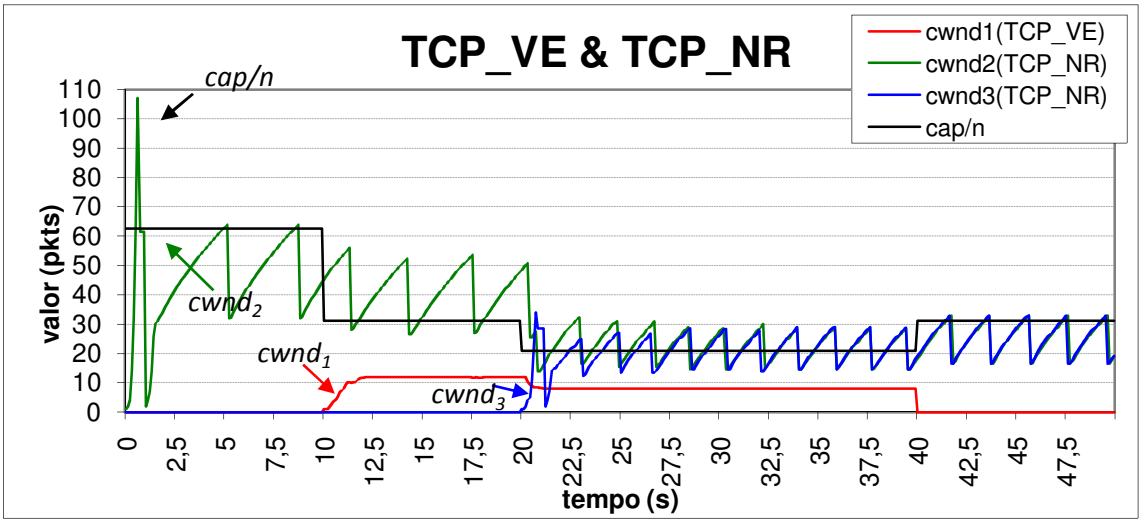


Figura 5.39: Compatibilidade TCP_VE com TCP_NR – Cenário 7.b – Simulação no NS2.

Síntese da sessão

Esta sessão apresentou primeiramente os resultados de simulações sequenciais do algoritmo Equilíbrios Virtuais proposto por BHAYA e KASZKUREWICZ (2007)

frente ao tradicional AIMD. Resultados neste ambiente de simulação demonstraram que a adequada escolha dos parâmetros das dinâmicas de incremento e decremento resulta em melhores resultados.

Logo foram apresentados resultados dos protocolos TCP_WVE e TCP_VE implementados no simulador NS2 e comparados com um dos protocolos de controle de congestionamento mais tradicionais, o TCP New Reno. Cenários de estudo no simulador mostraram que os melhores tempos de convergência, o menor desperdício de largura de banda disponível e as menores oscilações para ambos os protocolos propostos em detrimento de perdas de pacotes.

5.2 Simulações de propostas baseadas em otimização

5.2.1 Técnica de avaliação e objetivos

Para avaliar o desempenho das propostas de algoritmos baseados em otimização o estudo foi realizado de forma lineal programando em linguagem C com a ferramenta *Visual Studio.NET*. Versões síncronas e assíncronas dos algoritmos AIMD, OptQua e OptAbs foram feitas e os resultados comparados para o estudo de caso de dois usuários e dez usuários que compartilham um único canal de capacidade cap; sendo alguns dos resultados experimentais publicados em BENITEZ, et.al. (2008).

O objetivo dos estudos realizados na comparação de AIMD com as novas propostas OptQua e OptAbs é avaliar o melhor comportamento em termos de uma melhor utilização da largura de banda, tempo de resposta, convergência, tamanho das oscilações, e equidade entre taxas de transmissão dos usuários.

5.2.2 Comparação dos algoritmos AIMD, Otimização Quadrática e Otimização Absoluta

Considerando novamente o estudo de caso #1, duas fontes, ou usuários x_1 e x_2, que compartilham um único canal de capacidade cap pacotes/segundo.

O modelo S_AIMD_CJ [13] proposto por CHIU e JAIN (1989) com atrasos homogêneos (ou síncronos) para os seguintes parâmetros $cap = 1$, $a_I = 0.08$, $b_I = 1$,

[13] Lembre se da sessão 2.1 a notação de S_AIMD_CJ para a versão síncrona do AIMD e A_AIMD_G para a versão assíncrona do mesmo.

$a_D = 0$, $b_D = 0.95$ e as condições iniciais $x_1(0) = 0.3$ e $x_2(0) = 0.1$ apresenta como resultado o comportamento ilustrado na Figura 5.40. Pode-se observar que partindo das condições iniciais, com os parâmetros escolhidos, o sistema atinge o ponto de estabilidade ótimo $x_1^* = x_2^* = 0.5$.

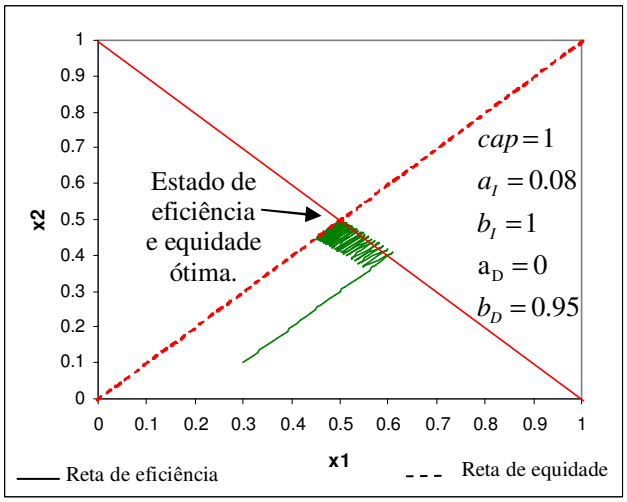


Figura 5.40: Diagrama de estado do S_AIMD_CJ para o estudo de caso #1 – Dados obtidos da simulação sequencial.

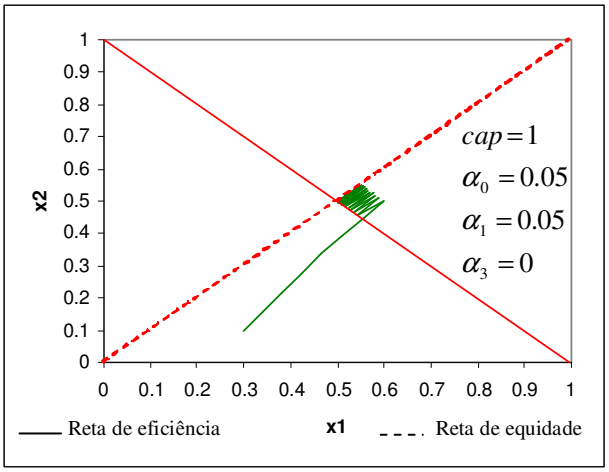


Figura 5.41: Diagrama de estado do OptQua para o estudo de caso #1 – Dados obtidos da simulação sequencial.

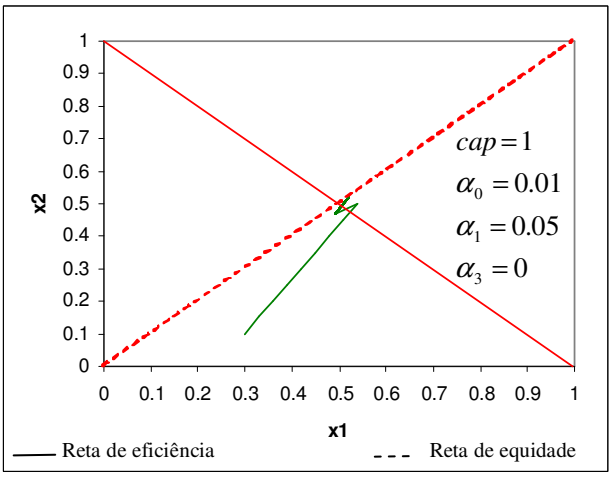


Figura 5.42: Diagrama de estado do OptAbs para o estudo de caso #1 – Dados obtidos da simulação sequencial.

A Figura 5.41 mostra o resultado da simulação do estudo de caso #1 do modelo OptQua com parâmetros de penalidade $\alpha_0 = 0.05$, $\alpha_1 = 0.05$ e $\alpha_3 = 0$, e a Figura 5.42 o resultado do OptAbs com $\alpha_0 = 0.01$, $\alpha_1 = 0.05$ e $\alpha_3 = 0$.

Observando as figuras 5.41, 5.42 e 5.43 pode-se constatar que a pesar dos modelos baseados em otimização propostos neste trabalho apresentarem comportamentos similares OptAbs tem um melhor tempo de resposta que o OptQua (≈ 10 vs. ≈ 40). Tendo em conta que os parâmetros de penalidade escolhidos em ambos os novos modelos são similares em magnitude, este comportamento pode ocorrer devido a escolha da função objetivo do modelo OptAbs, $f(x) = |x_1 - x_2|$, que faz com que a dinâmica do algoritmo seja mais rápida que a dinâmica do modelo OptQua que usa a função objetivo $f(x) = 0.5(x_1 - x_2)^2$. Devê-se isto à inclinação da função $f(x) = |x_1 - x_2|$ é maior que da função $f(x) = 0.5(x_1 - x_2)^2$.

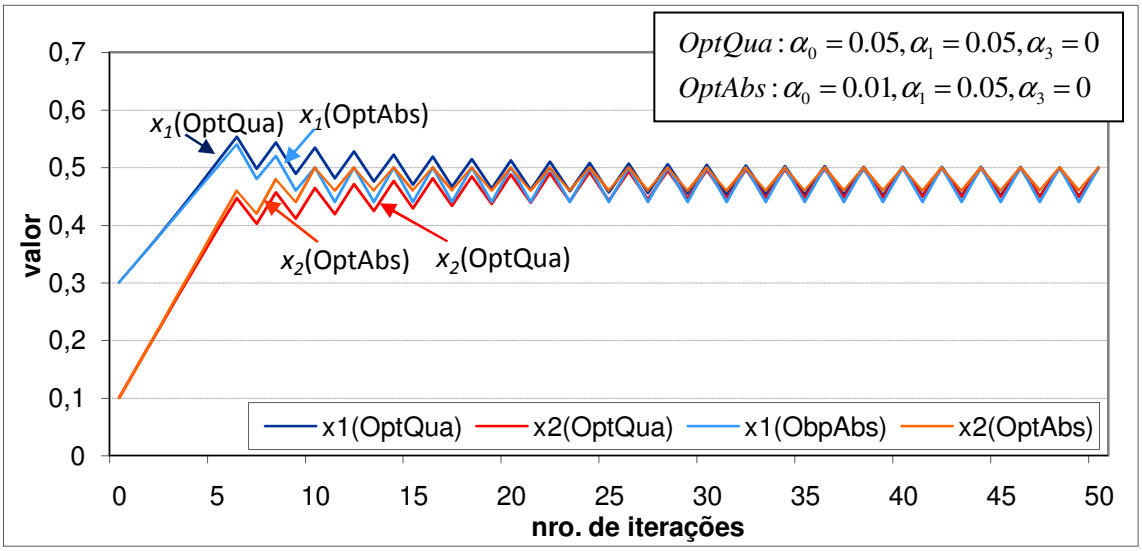


Figura 5.43: Comparação dos modelos OptQua e OptAbs em presença de atrasos homogêneos – Resposta em função do tempo – Dados obtidos da simulação sequencial.

As respostas dos modelos OptQua e OptAbs são consideravelmente melhores que a resposta do AIMD como corrobora-se na comparação da Figura 5.44. Pode-se observar que, na presença de atrasos homogêneos ambos os modelos propostos apresentam menor tempo de convergência ao ponto de eficiência máxima (isto é, melhor tempo de resposta) e amplitude de oscilações menor do que aquela apresentada pelo modelo AIMD (isto é, mais suave).

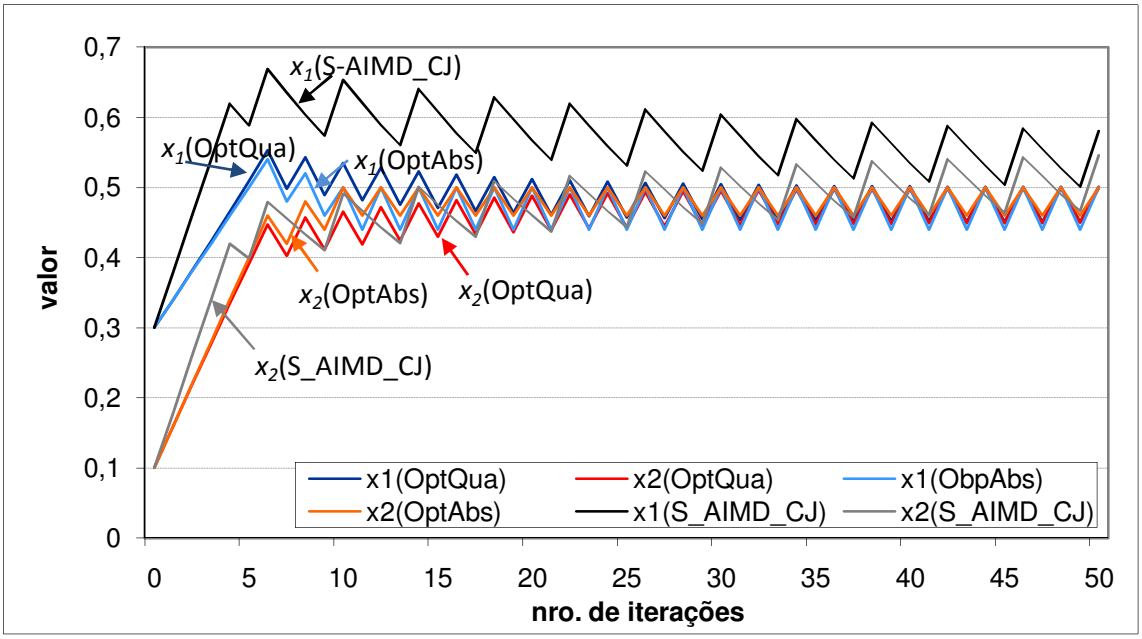


Figura 5.44: Comparação dos modelos AIMD, OptQua e OptAbs em presença de atrasos homogêneos. - Respostas em função do tempo – Dados obtidos da simulação sequencial.

Para verificar se este comportamento ocorre para qualquer condição inicial do estudo de caso #1, os modelos foram simulados com diversas condições iniciais. O resultado é apresentado na Tabela 5.6. Para todos os casos a simulação foi feita com um número máximo de iterações igual a 500 e para achar o tempo de resposta foi utilizada a condição:

$$\left(\left|x_1(k)-x_2(k)\right|<tol\right)\&\&\left(\left|F.O.-x_1(k)\right|<tol\right)\&\&\left(\left|F.O.-x_2(k)\right|<tol\right) \qquad \text{com } tol=0.001$$

Resultados demonstram que o modelo OptAbs tem um tempo de resposta melhor e converge à equidade ótima (*F.O.*) em todos os casos. Também pode se observar que o modelo OptQua apresenta uma melhor convergência que o modelo tradicional AIMD que tem tempos de resposta significativamente mais altos, já que mesmo atingindo o número máximo de iterações o algoritmo não converge com a tolerância solicitada.

Tabela 5.6 – Comparação para diversas condições iniciais – Síncrono.

ALG.	$x_1(0)$	$x_2(0)$	k_{resp}	$x_1(k_{resp})$	$x_2(k_{resp})$
S_AIMC_CJ $a_I=0.08,$ $b_I=1,$ $b_D=0.95$	0.3	0.1	500	0.5329	0.5329
	0.1	0.7	500	0.5329	0.5329
	0.5	0.1	500	0.5329	0.5329
	0.0	0.0	500	0.5609	0.5609
	0.7	0.1	500	0.5329	0.5329
	0.9	0.0	500	0.4809	0.4809
OptQua $\alpha_0=0.05,$ $\alpha_1=0.05,$ $\alpha_3=0$	0.3	0.1	53	0.5004	0.4996
	0.1	0.7	63	0.4996	0.5004
	0.5	0.1	59	0.5004	0.4996
	0.0	0.0	11	0.5	0.5
	0.7	0.1	63	0.5004	0.4996
	0.9	0.0	66	0.5005	0.4995
OptAbs $\alpha_0=0.01,$ $\alpha_1=0.05,$ $\alpha_3=0$	0.3	0.1	**11**	0.5	0.5
	0.1	0.7	**33**	0.5	0.5
	0.5	0.1	**21**	0.5	0.5
	0.0	0.0	**11**	0.5	0.5
	0.7	0.1	**31**	0.5	0.5
	0.9	0.0	**46**	0.5	0.5

Estudo de Caso #4: Dez fontes, ou usuários (x_1 até x_{10}) compartilhando o mesmo canal de capacidade $cap=10$ pacotes/segundo.

As Figuras 5.45, 5.46 e 5.47 apresentam as simulações para dez usuários compartilhando um único canal com capacidade de 10 pacotes/segundo com atrasos homogêneos para S_AIMD_CJ, OptQua e OptAbs, respectivamente.

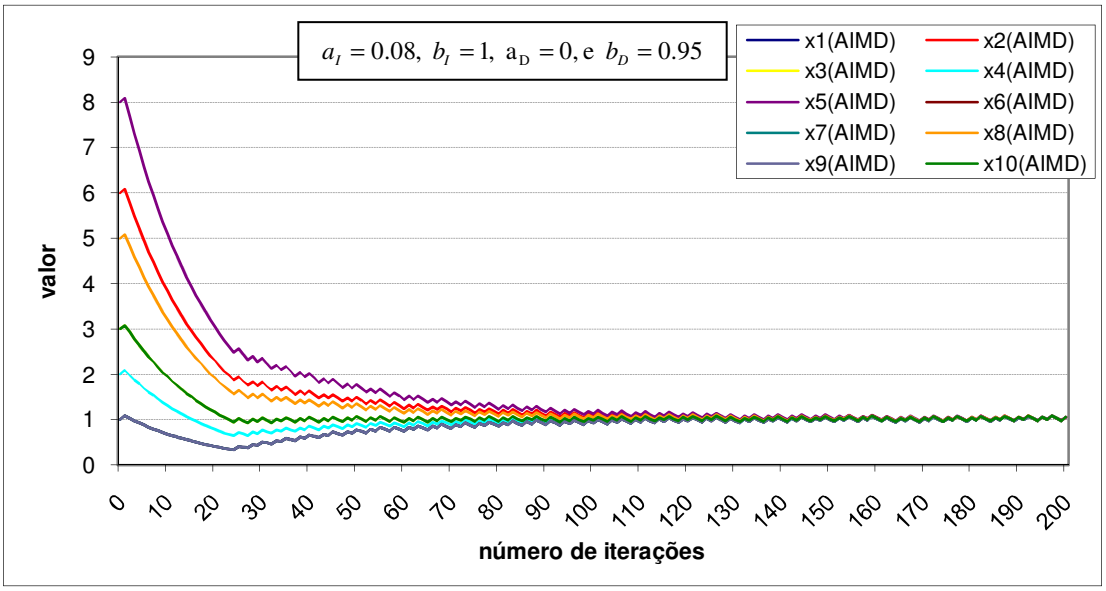


Figura 5.45: Comportamento do S_AIMD_CJ para estudo de caso #4 - Resposta em função do tempo – Dados obtidos da simulação sequencial.

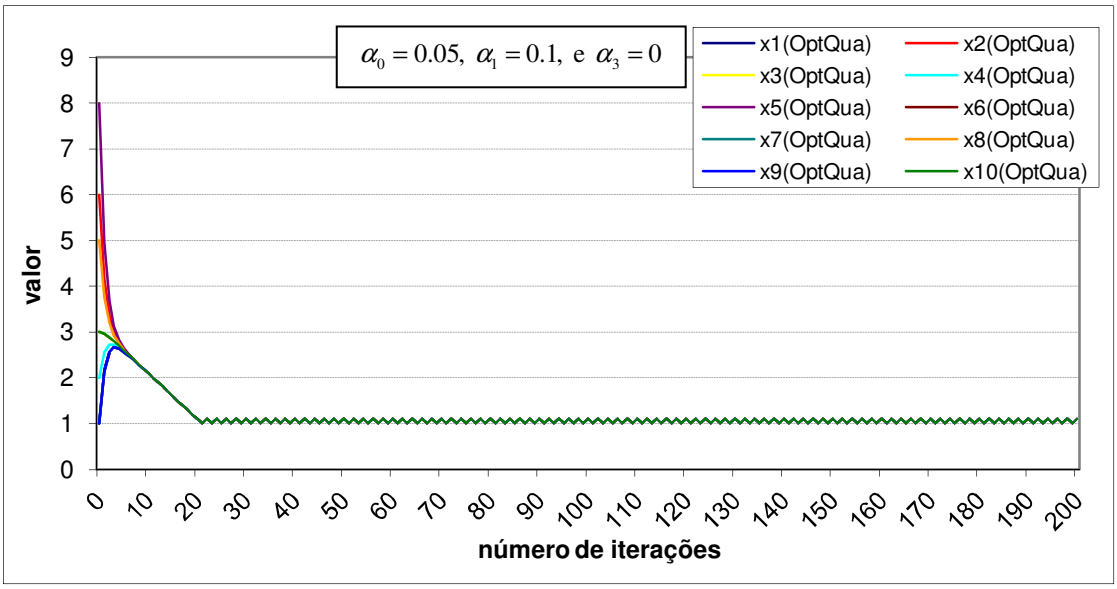


Figura 5.46: Comportamento do OptQua para estudo de caso #4 - Resposta em função do tempo – Dados obtidos da simulação sequencial.

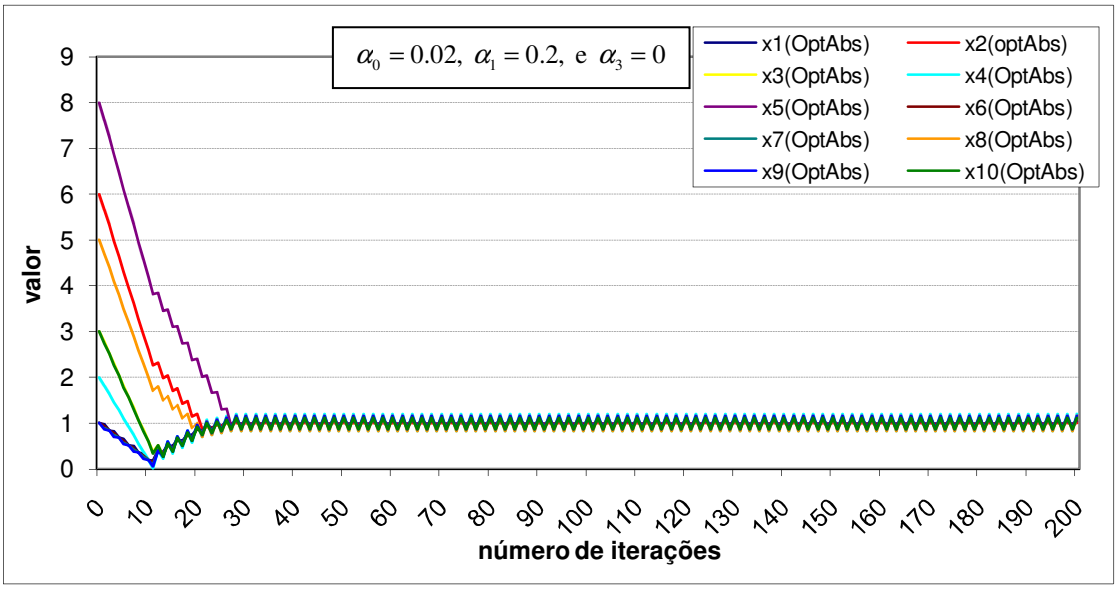


Figura 5.47:Comportamento do algoritmo OptAbs para estudo de caso #4 - Resposta em função do tempo – Dados obtidos da simulação sequencial.

Ambas as propostas melhoram os resultados de Chui e Jain (S_AIMD_CJ). Enquanto a convergência do S_AIMD_CJ acorre em aproximadamente 130 iterações, para o caso do OptAbs a convergência ocorre em aproximadamente 30 iterações para todos os usuários. Nota-se que ainda que o OptQua é melhor que o S_AIMD_CJ em convergência (20 iterações aproximadamente). Existe um tempo em que nenhum usuário está transmitindo dados, como pode ser observado entre as iterações 6 e 20, na qual todas as curvas descendem indicado a perda de pacotes. Por isto, para este estudo de caso o OptAbs tem um melhor comportamento que o OptQua e o S_AIMD_CJ.

Em um ambiente mais realista o numero de usuários que acessam a rede varia no tempo. Isto pode ser simulado variando o parâmetro *cap* para observar o comportamento dos modelos. Um incremento do parâmetro *cap* equivale a usuários deixando de transmitir, resultando em um incremento da largura de banda; por outro lado, um decremento no valor da capacidade *cap* equivale a mais usuários começando a transmitir dados na rede. Espera-se que o algoritmo utilizado responda rapidamente às variações da capacidade da rede. Da mesma forma, é possível ter atrasos nos roteadores da rede, representados por atrasos heterogêneos nas equações.

Estudo de Caso #5: Considere o estudo de caso #1 com uma variação da capacidade da rede gerada aleatoriamente entre os valores [0.7, 1] a cada 10 iterações.

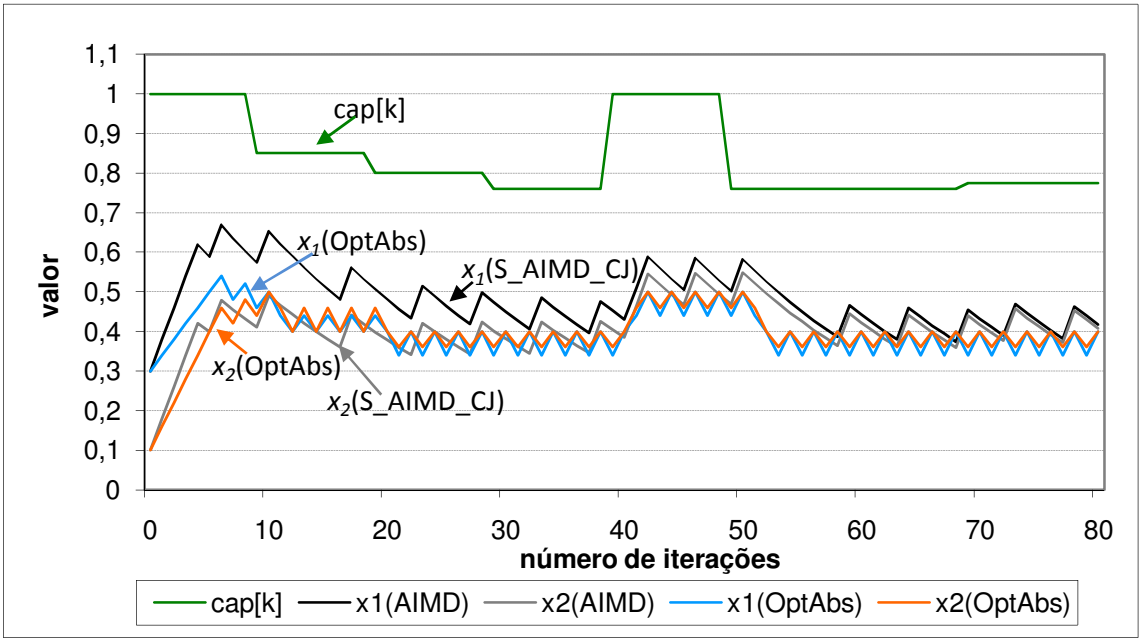


Figura 5.48: Comparação S_AIMD_CJ e OptAbs em presença de atrasos homogêneos variando a *cap* da rede entre [0.7,1] - Resposta em função do tempo – Dados obtidos da simulação sequencial.

Assim a Figura 5.48 mostra os resultados de variação da capacidade *cap* a cada 10 iterações (ou unidades de tempo), com um valor aleatório entre [0.7, 1] para ambos os usuários utilizando o algoritmo S_AIMD_CJ e a proposta OptAbs. Por exemplo, na iteração 50 o parâmetro *cap* varia para o valor 0.73. Conforme esperado o algoritmo AIMD demora mais para reagir. O S_AIMD_CJ demora aproximadamente 8 iterações para reagir e o algoritmo OptAbs aproximadamente 3 iterações. Assim, um algoritmo que reage mais rapidamente as mudanças de largura de banda disponível por usuário é mais efetivo.

Estudo de Caso #6: Considere o estudo de caso #1 mas com atrasos heterogêneos com tempos totais de percurso ou *RTT* de 4 (formado por $d_1^f = d_1^b = 2$) para o primeiro usuário e 10 (consistido de $d_2^f = d_2^b = 5$) para o segundo.

Os parâmetros de simulação utilizados para este estudo de caso são: $a_I = 0.01$, $b_I = 1$, e $b_D = 0.9$ para a versão assíncrona de AIMD (A_AIMD_G de sessão 2.1.2), $\alpha_0 = 0.5$ $\alpha_1 = 0.005$ $\alpha_3 = 0$ para OptQua, e $\alpha_0 = 0.01$, $\alpha_1 = 0.01$, $\alpha_3 = 0$ para OptAbs.

Com as condições iniciais dos dois usuários $x_1(0) = 0.3$, e $x_2(0) = 0.1$, em presença de atrasos heterogêneos (ou diferentes RTT) os algoritmos baseados em otimização OptQua e OptAbs propostos apresentam melhor tempo de resposta e uma melhor suavidade quando comparados com o modelo AIMD de Chiu e Jain em sua versão assíncrona (S_AIMD_G). As Figuras 5.49-5.52 apresentam o resultado destas simulações. O modelo OptAbs apresenta novamente um melhor desempenho que o modelo OptQua e o AIMD.

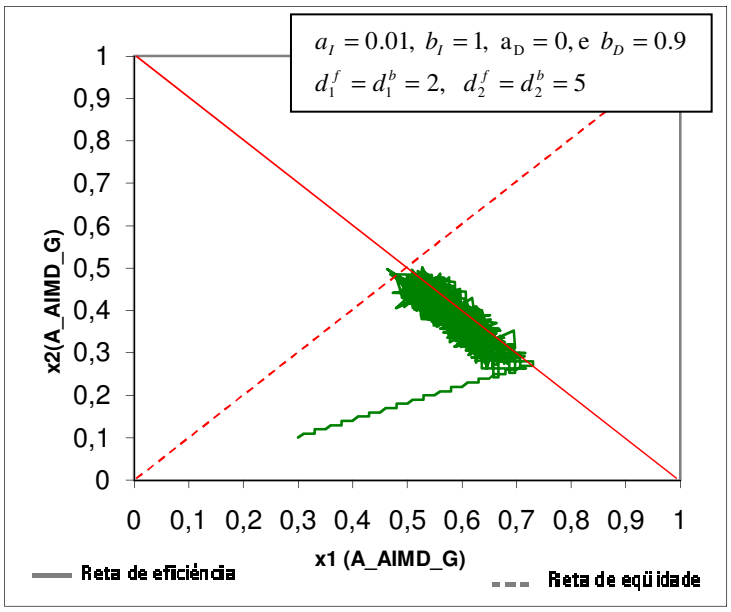


Figura 5.49: Diagrama de estados do A_AIMD_G para o estudo de caso #6 com atrasos heterogêneos – Dados obtidos da simulação sequencial.

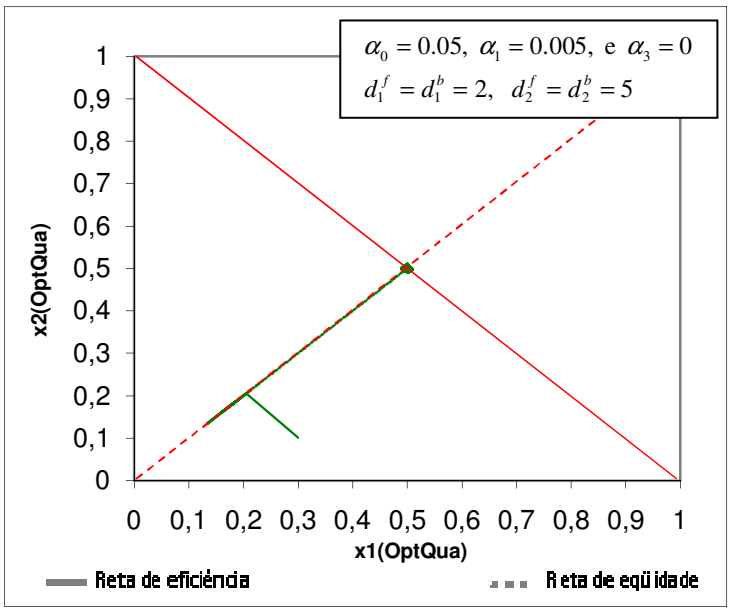


Figura 5.50: Diagrama de estados do OptQua para o estudo de caso #6 com atrasos heterogêneos – Dados obtidos da simulação sequencial.

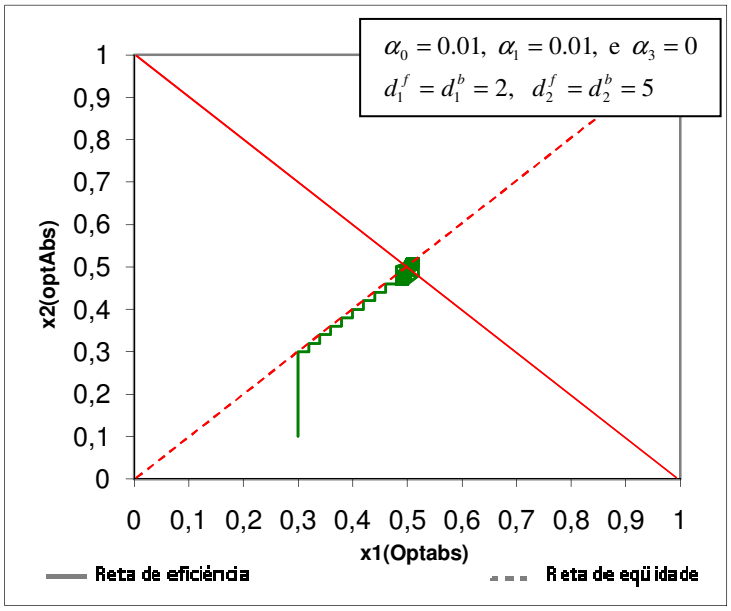


Figura 5.51: Diagrama de estados do OptAbs para o estudo de caso #6 com atrasos heterogêneos – Dados obtidos da simulação sequencial.

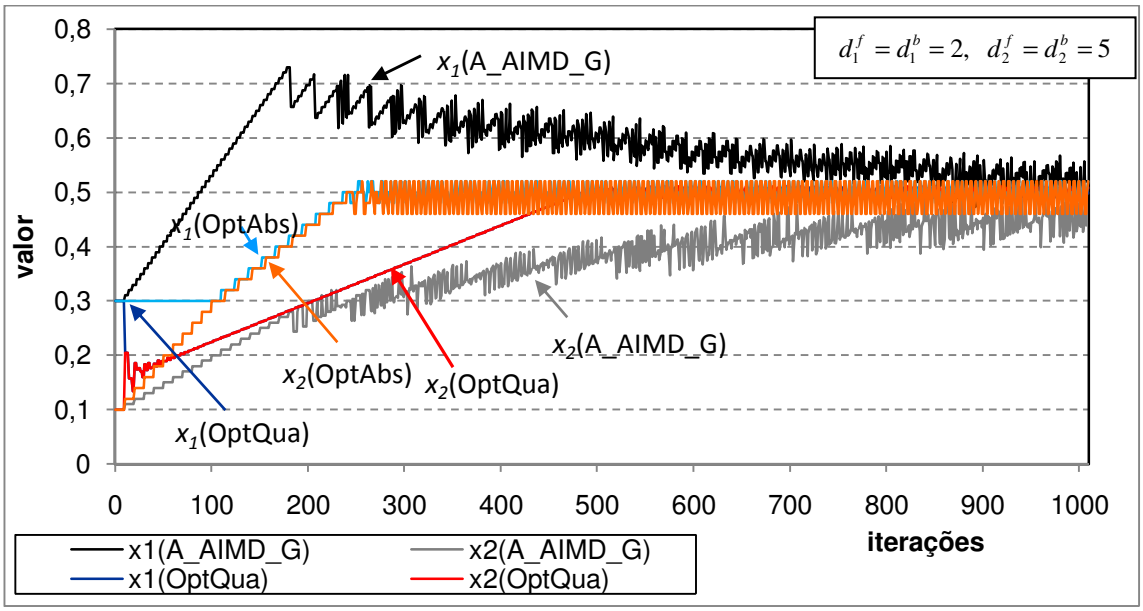


Figura 5.52: Comparação dos modelos A_AIMD_G, OptQua e OptAbs para o estudo de caso #6 com atrasos heterogêneos - Resposta em função do tempo – Dados obtidos da simulação sequencial.

A Tabela 5.7 apresenta uma comparação entre os modelos citados considerando diversas condições iniciais para o estudo de caso #6 com atrasos heterogêneos $d_1^f = d_1^b = 2$, $d_1^f = d_1^b = 5$. Resultados demonstram que o modelo OptQua tem melhor comportamento, já que converge à equidade ótima (F.O.) em todos os casos, seguido pelo modelo OptAbs.

Tabela 5.7 – Comparação para diversas condições iniciais – Assíncrono.					
Modelo.	$x_1(0)$	$x_2(0)$	k_{resp}	$x_1(k_{resp})$	$x_2(k_{resp})$
A_AIMD_G $a_I = 0.01,$ $b_I = 1,$ $b_D = 0.9$	0.3	0.1	1000	0.5069	0.4441
	0.1	0.7	1000	0.4738	0.5053
	0.5	0.1	1000	0.5023	0.4551
	0.0	0.0	1000	0.5476	0.4451
	0.7	0.1	1000	0.5364	0.4365
OptQua $\alpha_0 = 0.5,$ $\alpha_1 = 0.005,$ $\alpha_3 = 0$	0.3	0.1	485	0.4994	0.4994
	0.1	0.7	60	0.5004	0.5004
	0.5	0.1	405	0.4994	0.4994
	0.0	0.0	705	0.4994	0.4994
	0.7	0.1	325	0.4994	0.4994
	0.9	0.0	345	0.4994	0.4994
OptAbs $\alpha_0 = 0.01,$ $\alpha_1 = 0.01,$ $\alpha_3 = 0$	0.3	0.1	**241**	0.5	0.5
	0.1	0.7	**176**	0.5	0.5
	0.5	0.1	**201**	0.5	0.5
	0.0	0.0	**351**	0.5	0.5
	0.7	0.1	**220**	0.5	0.5
	0.9	0.0	**292**	0.5	0.5

Considerando agora o comportamento dos modelos AIMD, OptQua e OptAbs em presença de atrasos heterogêneos com as duas condições iniciais especiais.

Estudo de caso #7: Considere duas fontes, ou usuários, x_1 e x_2, que compartilham um mesmo canal de capacidade $cap = 100$ pacotes/segundo. Os valores dos atrasos heterogêneos utilizados são $d_1^f = d_1^b = 4$ para o usuário 1 e $d_2^f = d_2^b = 1$ para o usuário 2. O comportamento dos modelos AIMD, OptQua e OptAbs em presença de atrasos heterogêneos é estudado primeiro, com condições iniciais justas mas ineficientes (FI: $x_1(0) = 0, x_2(0) = 0$) e enseguida, com condições iniciais não justas mas, eficientes (EU: $x_1(0) = 100, x_2(0) = 0$).

Os parâmetros escolhidos para o caso do algoritmo AIMD são os parâmetros tradicionais do protocolo TCP: $a_I = 1$, $b_I = 1$, $a_D = 0$, e $b_D = 0.5$. Para OptQua são $\alpha_0 = 0.1$, $\alpha_1 = 1$, e $\alpha_3 = 0$; e para OptAbs: $\alpha_0 = 2$, $\alpha_1 = 1$, e $\alpha_3 = 0$.

As Figuras 5.53 e 5.54 demonstram que na presença de atrasos heterogêneos e condições extremas os algoritmos baseados em otimização (OptQua e OptAbs) possuem um melhor comportamento. Os tempos de resposta de ambos os algoritmos são muito menores que o algoritmo AIMD tradicional. A suavidade das oscilações também é significativamente menor. Constata-se que os valores das penalidades dos algoritmos

baseados em otimização propostos devem ser tais que além de aumentar a convergência não devem aumentar a suavidade das oscilações.

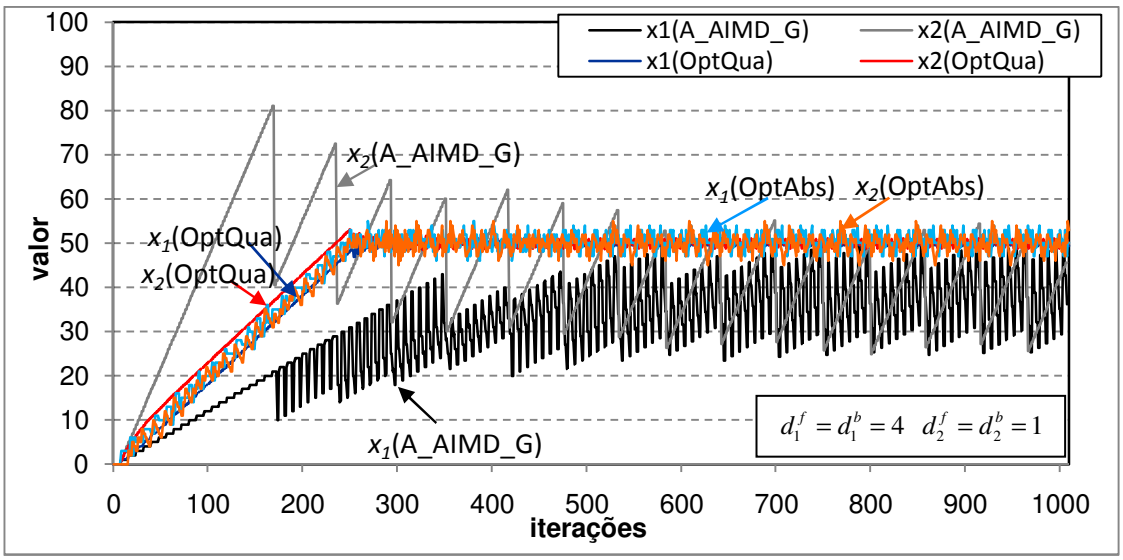


Figura 5.53: Comparação dos modelos A_AIMD_G, OptQua e OptAbs para o estudo de caso #7, condições iniciais FI $x_1(0) = 0, x_2(0) = 0$ e atrasos heterogêneos - Resposta em função do tempo – Dados obtidos da simulação sequencial.

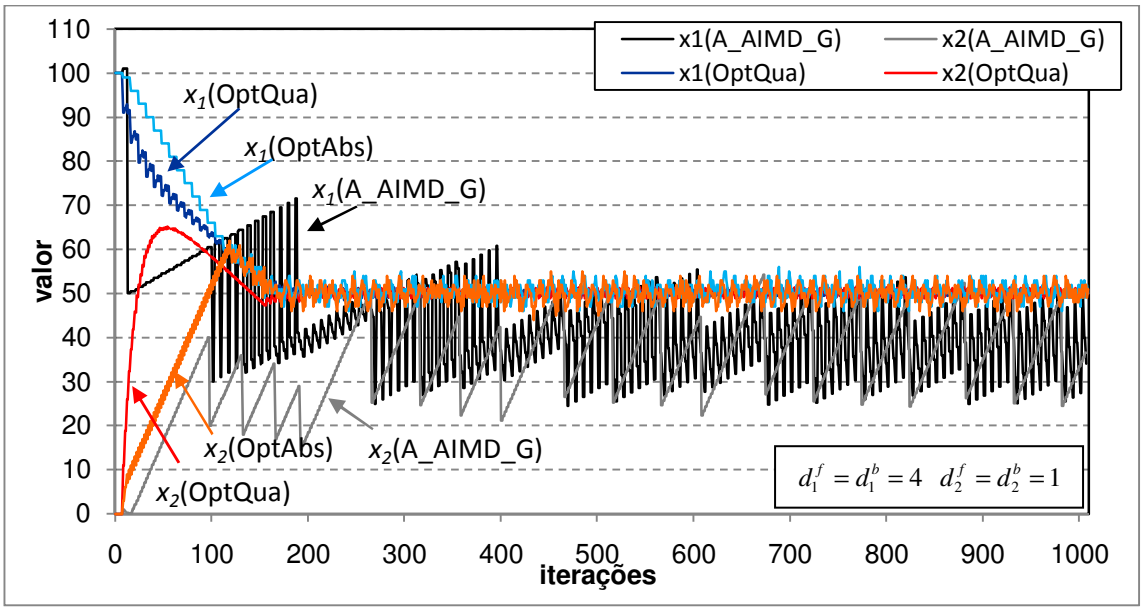


Figura 5.54: Comparação dos modelos A_AIMD_G, OptQua e OptAbs para o estudo de caso #7, condições iniciais EU $x_1(0) = 100, x_2(0) = 0$ e atrasos heterogêneos - Resposta em função do tempo – Dados obtidos da simulação sequencial.

Nas simulações dos estudos de casos #6 e #7 os tempos totais de percurso ou *RTT* foram considerados diferentes para os usuários. No entanto, estes tempos foram mantidos constantes ao longo da simulação. É importante destacar que nem é sempre

esses tempos são mantidos, pois os atrasos podem ser diferentes para cada usuário e para cada transmissão de dados. Logo, para poder representar este fato nas simulações os valores dos atrasos dos usuários podem ser obtidos em forma randômica (entre dois valores) em cada iteração. Assim temos o estudo de caso #8 a continuação.

Estudo de caso #8: Considere o estudo de caso #4, mas com atrasos heterogêneos variáveis escolhidos aleatoriamente entre dois valores Isto é $d_i^f = d_i^b = random[1,3]$. Esta variação representa os atrasos que os pacotes enviados através da rede podem sofrer nos roteadores da rede. Assim o algoritmo a ser utilizado para o controle deverá atuar eficientemente mesmo que a informação que se tem no momento de tomar a decisão de incrementar ou decrementar a taxa de envio esteja desatualizada.

A Figura 5.55 mostra a comparação do algoritmo AIMD com o algoritmo proposto, OptAbs, na presença de atrasos (d_i^f , d_i^b) heterogêneos variáveis aleatórios, entre os valores [1, 3] para cada 10 iterações. Através desta figura pode-se verificar que o algoritmo proposto OptAbs mesmo na presença de atrasos heterogêneos, apresenta um melhor tempo de resposta. Os parâmetros escolhidos para o caso do algoritmo A_AIMD_G são: $a_I = 0.05$, $b_I = 1$, $a_D = 0$, e $b_D = 0.9$ e para OptAbs: $\alpha_0 = 0.01$, $\alpha_1 = 0.03$, e $\alpha_3 = 0$.

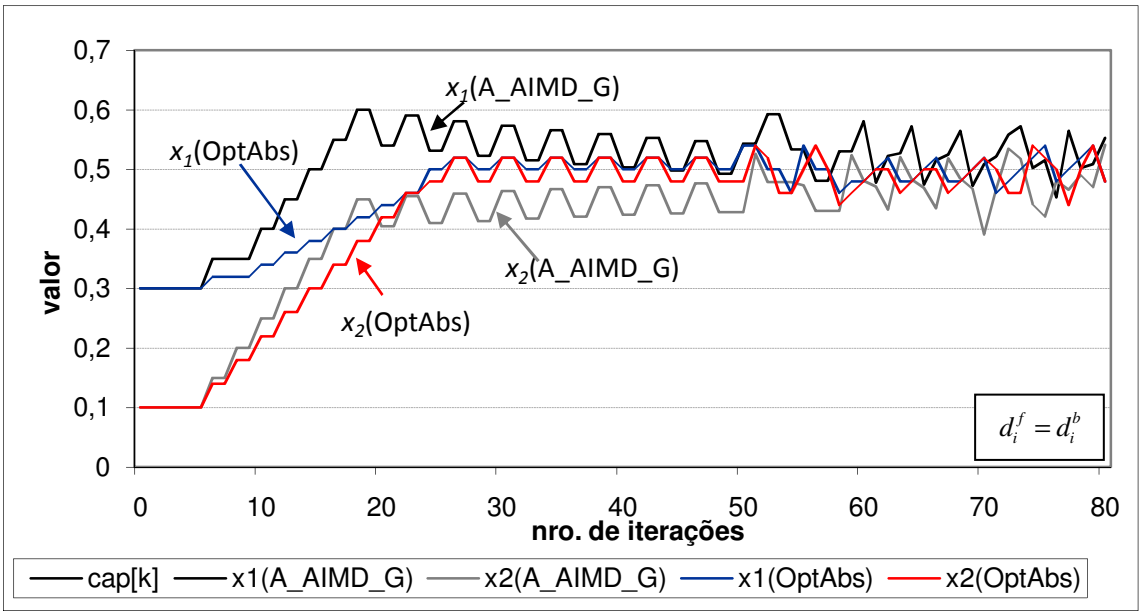


Figura 5.55: Comparação do A_AIMD_G e OptAbs com atrasos heterogêneos variáveis aleatoriamente entre [1, 3] - Resposta em função do tempo – Dados obtidos da simulação sequencial.

Considere-se agora uma combinação da variação de *cap* (variação no tempo de usuários compartilhando a rede) e a variação de atrasos heterogêneos (atrasos nos roteadores da rede).

Estudo de caso #9: Considere duas fontes ou usuários, x_1 e x_2 que compartilham um mesmo canal de capacidade $cap = 1$ pacotes/segundo; uma variação da capacidade da rede gerada aleatoriamente entre os valores [0.7, 1] para cada 10 iterações e atrasos heterogêneos variáveis e diferentes ($d_i^f \neq d_i^b$) escolhidos aleatoriamente entre [1, 3] em cada iterações.

Os parâmetros da simulação escolhidos para o caso do algoritmo A_AIMD_G são: $a_I = 0.05$, $b_I = 1$, $a_D = 0$, e $b_D = 0.9$; e para OptAbs: $\alpha_0 = 0.01$, $\alpha_1 = 0.03$, e $\alpha_3 = 0$.

Através da Figura 5.56 também, se pode confirmar que o algoritmo OptAbs apresenta um melhor comportamento. Por exemplo, na iteração $k = 25$ temos $cap = 0.85$ mesmo com a presença de atrasos aleatórios $x_1 = 0.48$ para o algoritmo OptAbs, enquanto que $x_1 = 0.55$ para o modelo A_AIMD_G. O modelo OptAbs atua eficientemente mesmo que a informação presente no momento de tomar a decisão de incrementar ou decrementar a taxa de envio esteja desatualizada e exista uma variação no número de usuários que utilizam a rede.

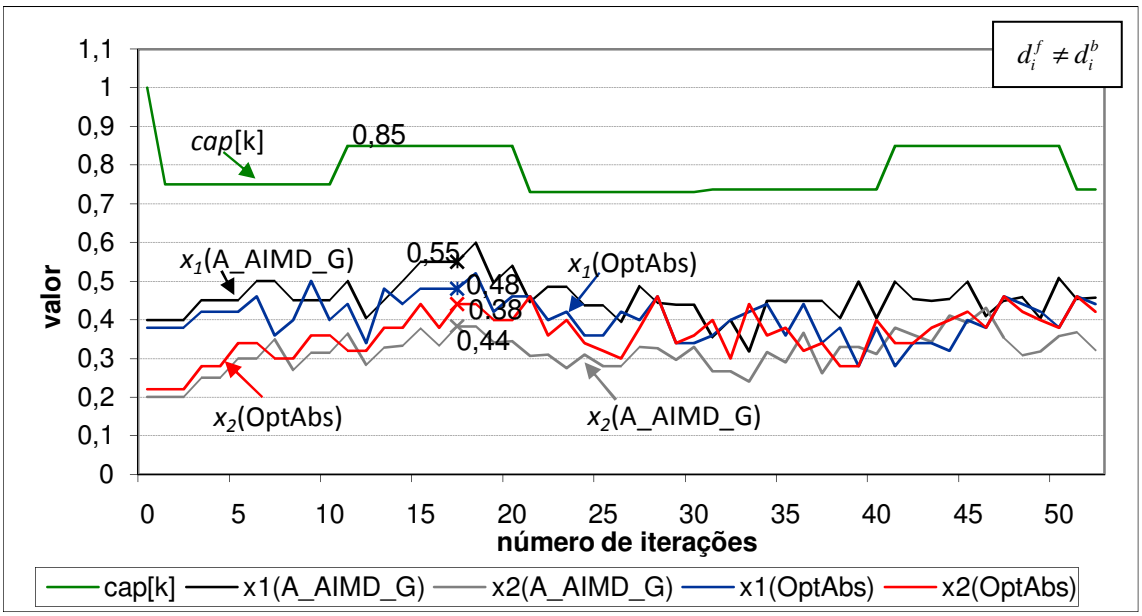


Figura 5.56: Comparação do A_AIMD_G e OptAbs com atrasos heterogêneos variáveis aleatorios entre [1, 3], e variação da capacidade da rede entre [0.7, 1] - Resposta em função do tempo – Dados obtidos da simulação sequencial.

Finalmente verifiquemos o comportamento para um estudo de caso com maior número de usuários.

Estudo de caso #10: Considere dez fontes, ou usuários, x_1 até x_{10} que compartilham um mesmo canal de capacidade $cap = 10$ pacotes/segundo, uma variação da capacidade da rede gerada aleatoriamente entre os valores [5, 10] para cada 20 iterações e atrasos heterogêneos variáveis e diferentes ($d_i^f \neq d_i^b$) escolhidos aleatoriamente entre [1, 3] em cada iteração.

Os parâmetros da simulação escolhidos para o caso do algoritmo A_AIMD_G são: $a_I = 0.08$, $b_I = 1$, $a_D = 0$, e $b_D = 0.95$; e para OptAbs: $\alpha_0 = 0.02$, $\alpha_1 = 0.2$, e $\alpha_3 = 0$.

A Figuras 5.57 e 5.58 mostram o comportamento de dois usuários, dentre os 10 usuários que compartilham um único canal com capacidade $cap = 10$, para os algoritmos A_AIMD_G e OptAbs respectivamente. Nota-se também neste estudo de caso que o algoritmo OptAbs apresenta um melhor comportamento, mesmo com um maior número de usuários. Por exemplo, na iteração $k = 60$ a capacidade diminui do valor 10 para o valor 6 e, mesmo na presença de atrasos aleatórios, o algoritmo OptAbs se adapta à variação de capacidade mais rapidamente do que o modelo AIMD.

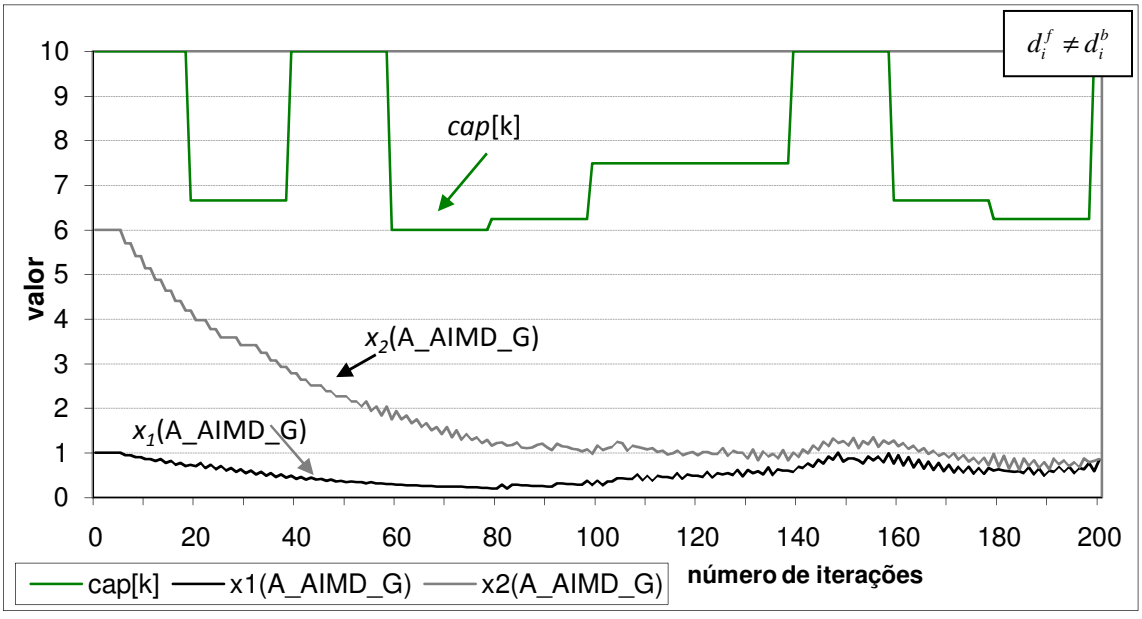


Figura 5.57: Comportamento do algoritmo A_AIMD_G com atrasos heterogêneos, variáveis e aleatórios, entre [1, 3], variando a capacidade da rede aleatoriamente entre [5, 10] - Respostas em função do tempo – Dados obtidos da simulação sequencial.

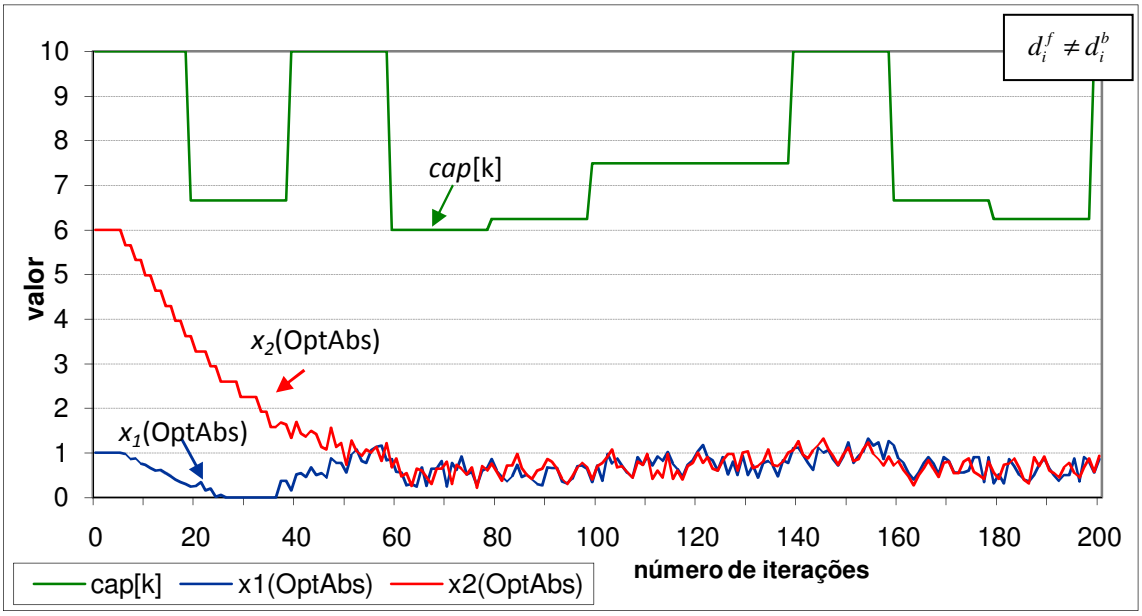


Figura 5.58: Comportamento do OptAbs com atrasos heterogêneos, variáveis, e aleatórios, entre [1, 3], variando-se a capacidade da rede aleatoriamente entre [5, 10] - Respostas em função do tempo – Dados obtidos da simulação sequencial.

Síntese da sessão

Esta sessão apresentou os das simulações dos algoritmos propostos que tratam o problema de controle de congestionamento sob uma perspectiva de otimização e controle. Os resultados das simulações com dois e dez usuários, para diversas condições iniciais, mostram que o tempo de resposta é melhorado, inclusive na presença de atrasos variáveis, de variação da capacidade do canal e no número de usuários. Os algoritmos propostos apresentam melhor convergência, considerando equidade e plena utilização da capacidade do canal.

Estes resultados levam a crer que a informação das taxas de envio dos vizinhos, em situações que tais informações estão disponíveis, pode justificar a melhora alcançada com esse tipo de controle.

Síntese do capitulo

Neste capitulo foram apresentados primeiramente resultados experimentais do Algoritmo Equilíbrios Virtuais (Alg_VE) comparado com o tradicional AIMD., demonstrando que a adequada escolha dos parâmetros das dinâmicas de incremento e decremento resulta em melhores resultados.

Seguidamente, foram apresentados resultados dos protocolos TCP Westwood Equilíbrios Virtuais (TCP_WVE) e TCP Westwood Equilíbrios Virtuais (TCP_VE) implementados no simulador NS2 comparados com o TCP New Reno demonstrando melhores tempos de convergência, o menor desperdício de largura de banda disponível e menores oscilações em detrimento de perdas de pacotes.

Também foram apresentados resultados de simulações das propostas baseados em otimização; o Algoritmo Otimização Quadrática (Alg_OptQua) e o Algoritmo Otimização Absoluta (Alg_OptAbs) que demonstraram melhor convergência, considerando equidade e plena utilização da capacidade do canal.

6. Conclusões e trabalhos futuros

Neste trabalho novas propostas de controle de congestionamento da Internet foram apresentadas, simuladas e analisadas. Duas linhas de investigação foram consideradas para propor as novas aproximações. A primeira linha foi baseada em uma seleção mais adequada dos parâmetros de incremento e decremento das dinâmicas do algoritmo tradicionalmente utilizado segundo equilíbrios virtuais (BHAYA e KASZKUREWICZ, 2007). A segunda linha baseada em otimização e inspirada no trabalho de KELLY et.al (2003).

Uma analise da seleção dos parâmetros de incremento e decremento do algoritmo AIMD proposto por CHIU e JAIN foi apresentado por BHAYA e KASZKUREWICZ (2007). Este trabalho afirma que havendo conhecimento sobre a capacidade da rede e do numero de usuários é possível ajustar os parâmetros de incremento e decremento de forma a obter os melhores tempos de convergência, as menores oscilações e, por conseguinte, uma forma mais eficiente de transmissão de dados. Baseados nesta proposta foram deduzidos dois modelos de controle de congestionamento da Internet, apresentados no Capítulo 3; implementados dois novos protocolos, detalhados no Capitulo 4; e apresentados e discutidos os resultados experimentais para diversos cenários no Capitulo 5.

A primeira proposta TCP Westwood Equilíbrios Virtuais ou TCP_WVE utiliza a estimativa de largura de banda (mais uma pequena sobre estimativa) disponível em uma rede empregada pelo protocolo TCP Westwood para calcular a quantidade de pacotes que a rede pode tratar (cap). Quanto ao número de usuários n compartilhando a rede, uma implementação foi feita nos roteadores da rede para que estes realizassem uma contagem dos identificadores dos fluxos que atravessam o roteador. Assim, com estes dois dados foi possível aplicar as equações que atualizam dinamicamente os parâmetros das dinâmicas de incremento e decremento e calcular o tamanho da janela de congestionamento. O TCP_WVE é eficiente, equitativo e melhora o tamanho das oscilações e o tempo de convergência em comparação ao tradicional TCP New Reno sendo compatível ao mesmo.

Uma segunda proposta inspirada nos equilíbrios virtuais é o protocolo TCP Equilíbrios Virtuais ou TCP_VE que, partindo de um valor inicial do parâmetro aditivo

de incremento e incrementando o mesmo progressivamente, utiliza a estimativa de cap/n no momento de uma perda de pacotes para atualizar dinamicamente o parâmetro aditivo de decremento. Assim ele ajusta os parâmetros de incremento e decremento seguindo as equações dos equilíbrios virtuais (Capítulo 3). Este protocolo apresenta uma boa eficiência, bons tempos de resposta, suavidade das oscilações e uma rápida transmissão de dados. No entanto, quando executada juntamente com outras transmissões governadas por protocolos tipo TCP New Reno, não consegue transmitir adequadamente se a sua transmissão ocorre depois das outras de tipo diferente.

A segunda linha de trabalho apresentou dois algoritmos Otimização Quadrática (OptQua) e Otimização Absoluta (OptAbs) inspirados no modelo de KELLY et.al (1998) cujo trabalho apresenta o problema de controle de congestionamento como um problema de otimização com restrições de capacidade dos canais de comunicação. Os modelos propostos neste trabalho utilizam de forma similar funções a ser otimizadas sujeitas as restrições de capacidade da rede. Assim, o problema de controle de congestionamento é formulado como um problema de otimização com restrições e reformulado como um problema de otimização sem restrições que pode ser resolvido mediante um sistema gradiente. Os novos algoritmos propostos foram expostos no Capitulo 3, no qual foram apresentados para dois usuários, assim como a formulação matemática dos mesmos para n usuários. O Capitulo 5 demonstrou com exemplos simulados linearmente as vantagens em eficiência, grau de resposta e equidade sobre o modelo tradicional AIMD.

6.1 Contribuições

Este trabalho utilizou dois enfoques para resolver o problema de controle de congestionamento da Internet, e mostrou resultados experimentais que justificam que os mencionados métodos e/ou protocolos podem favorecer o controle de congestionamento e melhorar as taxas de transmissão dos usuários que os utilizem. Entre as contribuições desta tese se encontram:

Técnica de estimativa de largura de banda

Uma técnica de estimativa da largura de banda, baseada na estimativa do TCP Westwood, foi utilizada para estabelecer os parâmetros das dinâmicas de incremento e decremento.

O conhecimento (aproximado) da capacidade de transmissão de uma rede, mediante esta técnica de estimativa da largura de banda, e do número de usuários que compartilham a mesma pode ser usado para ajustar dinamicamente os parâmetros do algoritmo de controle de congestionamento obtido se uma recuperação mais rápida das perdas de pacotes (eventos de ACK duplicados e *timeout*); isto é, protocolos de controle de congestionamento mais eficientes e equitativos em demérito das retransmissões.

Técnica de estimativa do número de usuários que compartilham uma rede

Uma técnica simples de estimativa do número de fluxos que compartilham a rede foi apresentada. Ela é implementada nos roteadores que se encontram no caminho da transmissão de dados e leva tempo de RTT para que as fontes obtenham esta informação.

A estimativa ajuda a obter uma rápida equidade, pois esta informação faz com que os fluxos que se encontraram transmitindo em um mesmo momento reduzam suas taxas o mais rapidamente possível adequando-se ao novo número de usuários que transmitem.

Algoritmos e Protocolos baseados em Equilíbrios Virtuais

Neste trabalho são propostos duas técnicas baseadas em equilíbrios virtuais e dois novos protocolos TCP_WVE e TCP_VE para controle de congestionamento de Internet que apresentam uma melhor eficiência e equidade

Nos protocolos TCP_WVE e TCP_VE o ajuste do parâmetro de incremento a_I faz com que a janela de congestionamento seja incrementada com mais que um segmento por vez contribuindo a um tempo de resposta mais efetivo.

No caso do TCP_WVE a utilização da técnica de estimativa da largura de banda do TCP Westwood junto como o ajuste do parâmetro de incremento traz como benéfico uma rápida recuperação dos eventos de ACK duplicados e *timeout*. Quando uma perda de pacotes por ACK duplicados ou timeout acontece os protocolos recalculam o parâmetro de decremento a_D fazendo que a janela de congestionamento se ajuste segundo a capacidade possível para uma conexão. Isto faz com que o decremento não seja muito agressivo como ocorre no caso do New Reno que no caso de ACK duplicados reduz drasticamente a janela de congestionamento e no caso de *timeout* é um segmento.

O TCP_WVE e TCP_VE apresentam uma dinâmica de incremento ou decremento agressiva ou suave dependendo da necessidade. No entanto, o beneficio de ser mais

eficiente é uma maior retransmissão, ou seja, o tamanho da janela de congestionamento oscila ao redor do limite máximo permitido para cada usuário.

Formulação matemática de novos algoritmos baseados em otimização

Este trabalho apresentou uma nova formulação matemática do problema de controle de congestionamento da Internet na qual as funções objetivo escolhidas ($f(x) = 0.5(x_1 - x_2)^2$ para OptQua e $f(x) = |x_1 - x_2|$ para OptAbs) fazem com que a dinâmica tenda para a reta da equidade, e a escolha da restrição $x_1 + x_2 - cap = 0$ faz com que a dinâmica ir para a reta da capacidade máxima.

Também foi apresentada a generalização da formulação matemática para n usuários que compartilham um único canal de comunicação.

Novos algoritmos baseados em Otimização

Este trabalho apresentou duas novas propostas que apresentam melhor convergência, considerando equidade e plena utilização da capacidade do canal, inclusive na presença de atrasos variáveis e de variação da capacidade do canal.

Os modelos baseados em otimização OptQua e OptAbs demonstram que planeando o problema como uma função objetivo com restrições é possível controlar eficientemente e equanimente o tamanho da janela de congestionamento. Assim a dinâmica dos mesmos se dirige ao ponto de equidade ótimo na qual todos os usuários têm igual quantidade de taxa de transmissão e a capacidade do canal de transmissão é eficientemente utilizada.

Os resultados experimentais demonstraram que os modelos OptQua e OptAbs são eficientes e equitativos inclusive na variação do número de usuários (ou variação da capacidade do canal) e/ou até mesmo na presença de atrasos variáveis. Assim, baseados numa proposta de controle é possível modelar e resolver o problema de controle de congestionamento.

O preço que deve ser pago por essa eficiência, equidade, suavidade das oscilações e bom grau de resposta alcançado é o conhecimento das taxas de envio dos vizinhos. Um cenário real no qual esta informação estaria disponível é um cluster de usuários com um servidor para conectar-se a Internet através de um único canal (também denominado canal gargalo). Os resultados prometedores levam a pensar que um novo protocolo pode

ser implementado e colocado no mencionado cluster para que os usuários que compartilham o acesso possam obter equitativa taxa de transmissão de dados.

Estudo de modelos inspirados em sistemas biológicos para Controle de Congestionamento

Estudos de modelos inspirados em sistemas biológicos mostram que o problema do controle de congestionamento pode ser resolvido como um sistema de equações diferenciais do tipo Lotka-Volterra.

Da mesma forma, a pesquisa realizada nesta tese mostra que problema de controle de congestionamento pode ser modelado por um sistema de otimização com restrições, utilizando o método de penalidades pode ser reformulado como um problema de otimização sem restrições, e finalmente resolvido mediante um sistema de gradiente que corresponde a um problema do tipo predador-presa.

6.2 Perspectiva de Trabalhos Futuros

Seguindo as linhas de pesquisa estudada nesta tese e tendo em vista os resultados obtidos, alguns tópicos tornam-se propícios para propostas de trabalhos futuro.

Melhorias na estimativa de largura de banda disponível da rede.

A estimativa da largura de banda do TCP Westwood pode ser utilizada como parâmetro no protocolo TCP_WE. No entanto, existe na literatura protocolos que empregam outras estimativas que podem ser utilizadas para verificar se estas melhoram o comportamento do TCP_WVE.

Outro aspecto a ser estudado é o valor utilizado de sobre estimativa do TCP_WVE. A estimativa do TCP Westwood dá um patamar inferior no momento da perda de pacotes para ajustar a janela de congestionamento. Entretanto, se esta for utilizada pura para ajustar dinamicamente os parâmetros de decremento e incremento, não conduz a uma eficiente utilização do canal compartilhado. É provado que o valor da sobre estimativa depende da topologia e das características da rede. Logo deve ser estudado como estabelecer o valor da sobre estimativa para obter uma convergência ao valor limite de transmissão para cada usuário.

Outro estudo que pode ser feito é a escolha de um patamar baseado na estimativa conseguida para que as oscilações fiquem abaixo do mencionado patamar e evitem perdas e, consequentemente, as retransmissões.

Estudo dos valores dos parâmetros das dinâmicas de incremento e decremento dos protocolos TCP_WVE e TCP_VE.

Neste trabalho os parâmetros aditivos das dinâmicas de incremento a_I e decremento a_D foram deduzidos segundo as equações dos equilíbrios virtuais. Entretanto os valores dos parâmetros multiplicativos de incremento b_I e decremento b_D , assim como os valores para as desigualdades respectivas foram fixos e selecionados de forma empírica para obter bons resultados. Estes valores merecem um estudo mais detalhado.

Melhoria da implementação da estimativa do número de usuários transmitindo na rede.

Este trabalho apresentou uma implementação simplificada do calculo do número de usuários que compartilham a rede. O objetivo era utilizar esta informação para verificar se os protocolos melhoravam seu comportamento com tal conhecimento. Outros métodos mais inteligentes e, por conseqüência, mais eficientes e rápidos podem ser implementados, como tabelas *hash* de um ou dois níveis nos roteadores.

Estudo dos valores dos parâmetros das dinâmicas de incremento e decremento dos algoritmos OptQua e OptAbs.

Os parâmetros utilizados nas simulações dos algoritmos OptQua e OptAbs foram obtidos empíricamente e ate heurística. Assim, sugere-se um estudo mais aprofundado sobre a escolha dos mesmos. Nota-se uma relação entre a seleção dos parâmetros e a convergência dos casos de estudo, o que merece uma maior investigação.

Implementação dos algoritmos OptQua e OptAbs basados em protocolos para controle de congestionamento.

Este trabalho apresentou somente simulações de forma sequencial dos algoritmos baseado em otimização. Uma implementação dos mesmos no simulador NS2 merece especial atenção para verificar se os resultados obtidos se aproximam com uma situação

mais real. Assim se poderá ter uma melhor forma de comparar com protocolos utilizados atualmente como o TCP New Reno.

7. Referências bibliográficas

BANSAL, D., BALAKRISHNAN, H., 2001, "Binomial Congestion Control Algorithms". In; *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM 2001*, v.2, pp.631-640, Anchorage, AK, USA, Abr.

BENÍTEZ, D., KASZKUREWICZ, E., BHAYA, A., 2008, "Controle de congestionamento em redes de comunicação: um modelo baseado em otimização", *XVII Congresso Brasileiro de Automática*, Juiz de Fora, MG, Brasil.

BERTSEKAS, D., GALLAGER, R., 1992, *Data Networks*. 2 ed, New Jersey, Prentice Hall.

BHAYA, A., KASZKUREWICZ, E., 2006, *Control Perspectives on Numerical Algorithms and Matrix Problems*. 1 ed., Philadelphia, PA, USA, Society for Industrial and Applied Mathematic - SIAM.

BHAYA, A., KASZKUREWICZ, E., 2007, "An Improved AIMD-type scheme for congestion control derived using virtual equilibria", In: *46th IEEE Conference on Decision & Control*, pp. 5696-5703, New Orleans, LA, USA, Dec.

BRAKMO, L.S., O'MALLEY, S.W., PETERSON, L.L., 1994, "TCP Vegas: New Techniques for Congestion Detection and Avoidance". In: *Proceedings of the conference on Communications architectures, protocols and applications - SIGCOMM 1994*, v.24, pp.24-35, Londres, UK, Oct.

CASETTI, C., GERLA, M., MASCOLO, S., et.al., 2001, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links". In: *Proceedings of ACM Mobicom*. pp.287-297, Roma, Italia. Jul.

CHEN, C., LI, Z.G., SOH, Y.C., 2007, "Analysis of Monotonic Responsive Functions for Congestion Control". In: *Advanced in Multimedia Modeling*, v.4352/2006, *Lecture Notes in Computer Science*, Springer Berlin/Heildelberg, pp. 383-392.

CHIU, D., JAIN, R., 1989, "Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks", *Journal of Computer Networks and ISDN*, v.17, n.1, (Jun), pp.1-14.

COSTA, M.I.S., KASZKUREWICZ, E., BHAYA, A., 2000, "Achieving global convergence to an equilibrium population in predator–prey systems by the use of

a discontinuous harvesting policy", *Ecological Modeling, 2000*, V.128, n.2-3 (Abr), pp.89-99.

DEB, S., SRIKANT, R., 2002, "Global stability of congestion controllers for the Internet". *IEEE Transactions on Automatic Control, 2002*, v.48, n.6 (Jun), pp.1055-1060.

FAN, X., CHANDRAYANA, K., ARCAK, M. et.al., 2005, "A Two-Time Scale Design for Detection and Rectification of Uncooperative Network Flows". In: *Proceedings of the 44th IEEE Conference on Decision and Control - 2005 European Control Conference - CDC-ECC 2005*. pp.1842-1847, Sevilla, Espanha, Dez.

FAN, X., CHANDRAYANA, K., ARCAK, M., et.al., 2006, "A two-time-scale design for edge-based detection and rectification of uncooperative flows". *IEEE/ACM Transactions on Networking*, 2006. v.14, n.6 (Dez), pp1313-1322.

FLOYD, S., FALL, K., 1999, "Promoting the use of end-to-end congestion control in the Internet", *IEEE/ACM Transactions on Networking - TON 1999*, v.7, n.4(Ago), pp.458-472.

FLOYD, S., HENDERSON, T., 1999, *The New Reno Modification to TCP's Fast Recovery Algorithm*. In: RFC 2582, Network Working Group, USA.

FLOYD, S., *HighSpeed TCP for Large Congestion Windows*,. Disponível em: http://www.ietf.org/rfc/rfc3649.txt. Acesso em: Jun. 2008.

FLOYD, S., JACOBSON, V., 1993, "Random early detection gateways for congestion avoidance", *IEEE/ACM Transactions on Networking - TON 1993*, v.1, n.4 (Ago), pp.397-413.

FLOYD, S., KOHLER, E., 2005, "Internet Research Needs Better Models". *Proceedings of the First Workshop on Hot Topics in Networks (HotNes-I)*, pp., Princeton, NJ, USA, Out.

GEVROS, P., CROWCROFT, J., 2004, "Distributed resource management with heterogeneous linear controls", *Computer Networks – the International Journal of Computer and Telecomunications Networking*, v.45, n.6, (Ago), pp.835-858.

GHUNG, J., CAYPOOL, M., *NS by Example - Tutorial*, Computer Science Department - WPI Worcester Polytechnic Institute, Disponível em: http://nile.wpi.edu/NS/. Acesso em: Mar, 2006.

GLAZOS, M.P., HUI, S., ˙ZAK, S.H., 1998, "Sliding modes in solving convex programming problems", *SIAM Journal of Control and Optimization*, v.36, n.2, pp.680–697.

GOLESTANI, S.J., BHATTACHARYYA, S., 1998, "A Class of End-to-End Congestion Control Algorithms for the Internet". In: *Proceedings of the Sixth International Conference on Network Protocols - ICNP 1998*, pp.137-150, Austin, TX, USA, Out.

GORINSKY, S., 2004, "Feedback Modeling in Internet Congestion Control". In: *Proceedings Next generation Telegraphic and Wired/Wireless Advances Networking - NEW2AN 2004*, pp.231-234, Feb., 2004.

GORINSKY, S., JAIN, S., VIN, H, et.al., 2003, "Robustness to inflated subscription in multicast congestion control". In: *Proceedings of the 2003 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications - SIGCOMM 2003*. pp.87-98, Karlsruhe, Alemanha, Aug.

GORINSKY, S., VIN, H., 2002, *Extended Analysis of Binary Adjustment Algorithm*. In: Report TR2002-39, Department of Computer Sciences, University of Texas, Austin, TX, USA.

GRIECO, L.A., MASCOLO, S., FERORELLI, R., 2002, "Additive Increase Adaptive Decrease Congestion Control; a Mathematical Model and Its Experimental Validation". In: *7th International Symposium on Computers and Communications ISCC'02*, pp. 849-854, Taormina, Italia, Jul.

HORIE, R., AIYOSCHI, E., 1998, "Neural Networks Realization of Searching Models for Nash Equilibrium Points and Their Application to Associative Memories", In: *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1886-1891, San Diego, CA, USA, Oct.

IGUCHI, T., HASEGAWA, G., MURATA, M., 2005, "A new congestion control mechanism of TCP with inline network measurement", In: *Information Networking, Convergence in Broadband and Mobile Networking, International Conference - ICOIN 2005*, pp. 109-121, Jeju Island, Korea, Jan-Fev.

JACOBSON, V., 1998, "Congestion avoidance and control". In: *Symposium proceedings on Communications architectures and protocols - SIGCOMM 1988*, pp.314-329, Stanford, CA, USA, Ago.

JAIN, R., 1989, *A Delay-Based Approach for Congestion Avoidance in Heterogeneous Computer Networks*. In: Technical Report DEC-TR566, Digital Equipment Corporation, Littleton, MA, USA.

JAIN, R., 1991, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. 1ed., USA, Wiley.

JAMALI, S., ANALOUI, M., 2007, "Internet congestion control using nature population", *Ecological Informatics*, v.2, n.4 (Dez), pp.337-352.

KALYANARAMAN, S., JAIN, R., FAHMY, S. et.al., 2000, "The ERICA switch algorithm for ABR traffic management in ATM networks", *IEEE/ACM Transactions on Networking – TON 2000*, v.8, n.1 (Fev), pp.87-98.

KASZKUREWICZ, E., BAHYA, A., 2006, "Congestion control using an improved variant of the AIMD scheme". In: *45th IEEE Conference on Decision and Control*. pp.1894-1899, San Diego, CA, USA, Dez.

KATABI, D., HANDLEY, M., ROHRS, C., 2002, "Congestion control for high bandwidth-delay product networks". *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp.89-102, Pittsburgh, PA, USA, Ago.

KELLY, F., 2003, "Fairness and stability of end-to-end congestion control". In *European Control Conference*, v.9, pp.159-176, Cambridge, UK, Set.

KELLY, F.P., MAULLOO, A.K., TAN, D.K.H., 1998, "Rate Control in Communication Networks: Shadow Prices, Proportional Fairness, and Stability" *Journal of Operational Research Society*, v. 49, pp.237-52.

KELLY, T., 2003, "Scalable TCP: Improving Performance in High-speed Wide Area Networks", *ACM SIGCOMM Computer Communication Review*, v.33, n.2, pp.83-91.

KU, K., TIAN, Y., ANSARI, N., 2004, "TCP-Jersey for Wireless IP Communications", *IEEE Journal on Selected Areas in Communication,* v.22, n.4 (Mai), pp 747-756.

KUMAR, A. MANJUNATH, D. KURI, J., 2004, Communication Networking - An Analytical Approach. 1 ed., USA, Morgan Kaufmann Publishers.

LAHAMAS, A., 2003, *On the Efficiency and Fairness of AIMD-based*. Ph.D. dissertation, Northeastern University, USA.

LAHAMAS, A., TSAOUSSIDIS, V., 2002, "Additive Increase Multiplicative Decrease - Fast Convergence (AIMD-FC)". In: *Proceedings of Networks 2002*. pp.511-522, Atlanta, Georgia, USA, Ago.

LAHAMAS, A., TSAOUSSIDIS, V., 2003, "t-AIMD for Asynchronous Receiver Feedback". In: *Proceedings of the Eighth IEEE International Symposium on Computers and Communications - ISCC 2003*, pp. 735-740, Kemer-Antalya, Turkia, Jun-Jul.

LE, T., 2006, "TCP BaLDE for Improving TCP Performance over Heterogeneous Networks", *IEICE Transaction on Communication*, v.E89-B, n.4(Abr), pp.1127-1135.

LEE, S., LEANEY, J., O'NEIL, T, et.al., 2005, "Performance benchmark of parallel and distributed network simulator". *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*, pp.101-108, Monterey, CA, USA, Jun.

LIAN, F.L., LEE, H.T., FONG, T.C., 2007, "Phase Plane Analysis and Design for TCP Congestion Control". In: *IEEE International Conference on Control Applications - CCA 2007*, pp. 53-58, Singapore, Out.

LIN, D., MORRIS, R., 1997, "Dynamics of random early detection". In: *ACM SIGCOMM Computer Communication Review archive - SIGCOMM 1997*, v.27, n.4, pp.127-137, Canes, França, Set.

LOW, S.H., LAPSLEY, D.E., 1999, "Optimization Flow Control-I: Basic Algorithm and Convergence", *IEEE/ACM Transactions on Networking – TON 1999*, V.7, n.6 (Dez), pp.861-874.

LOW, S.H., PAGANINI, F., DOYLE, J.C., 2002, "Internet Congestion Control", *IEEE Control System Magazine*, v.22, n.1, (Fev), pp.28-43.

MACKIE-MASON, J.K., VARIAN, H.R., 1995, "Pricing congestible Network Resources" *IEEE Journal on Selected Areas in Communications*, v.13, n.7 (Set), pp.1141-1149.

MAMATAS, L., TSAOUSSIDIS, V., ZHANG, C., 1999, *Approaches to Congestion Control in Packets Networks*. In: Technical Report TR-DUTH-EE-2004-10, Dept. of Electrical & Computer Engineering, Demokritos University, Xanthi, Grécia.

MASCOLO, S., CASETTI, C., GERLA, M., et.al., 2000. *TCP Westwood: congestion control with faster recovery*. In: UCLA CSD Technical Report #200017, Computer Science Department, UCLA, CA, USA.

MATHIS, M., MAHDAVI, J., FLOYD, S., et.al. "TCP Selective Acknowledgment Options". Disponível em: http://www.ietf.org/rfc/rfc2018. Acesso em: Jun 2008.

MATHIS, M., MAHDAVI, J., FLOYD, S., et.al., 1996, *TCP Selective Acknowledgment Options*, In: RFC 2018, Network Working Group, USA.

MITRA, D., SEERY, J., 1993, "Dynamic adaptive windows for high speed data networks with multiple paths and propagation delays", *Source Computer Networks and ISDN Systems*, v.25, n.6 (Jan), pp.663-679.

UNIVERSITY OF SOUTHERN CALIFORNIA, *The Network Simulator - NS2*. Information Science Institute. Disponível em: http://www.isi.edu/nsnam/ns/. Acesso em: Mar 2006.

ODLYZKO, A., 199, "Paris metro pricing for the Internet". In: *Proceedings of the 1st ACM conference on Electronic Commerce*. pp.140-147, Denver, CO, USA, Nov.

OSISCHOOL, Dumbell Topology, Disponível em: http://www.osischool.com/concept/communication/dumbbell/. Acesso em: Mai. 2010.

PACHECO, D. M., LEFÈVRE, L., PHAM, C., 2008, "Lightweight Fairness Solutions for XCP and TCP Cohabitation". In: *IFIP/TC6 Networking 2008*, pp.715-726, Singapore, Mai.

PAGANINI, P., 2002, "A global stability result in network flow control", *Systems & Control Letters,* v.46, n.3, pp.165-172.

POSTEL, J., 1981, *Transmission Control Protocol. In: RFC 793*, Information Sciences Institute, University of Southern California, Marina del Rey, CA, USA.

RAMAKRISHNAN, K., FLOYD, S., 1999, *A Proposal to add Explicit Congestion Notification (ECN) to IP*, In: RFC 2481, Network Working Group, AT&T Labs Research e Lawrence Berkeley National Laboratory, USA.

RAMAKRISHNAN, K.K., JAIN, R., 1988, "A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer". In: *Proceedings of the ACM Symposium on Communications Architectures and Protocols - SIGCOMM 1988*, pp.303-313, Stanford, CA, USA, Ago.

RYU, S., RUMP, C., QIAO, C., 2003, "Advances In Internet Congestion Control", *IEEE Communications Surveys - Third Quarter 2003*, v.5, n.1, pp.28-39.

SHENKER, S., 1995, "Fundamental design issues for the future Internet", *IEEE Journal on Selected Areas in Communications*, v.13, n.7 (Set), pp.1176-1188.

SHENKER, S., CLARK, D., ESTIN, D. et.al., 1996, "Pricing in Computer Networks: Reshaping the Research Agenda", *ACM SIGCOMM Computer Communication Review*, v.26, n.2(Abr), pp.19-43.

SRIKANT, R., 2003, *The Mathematics of the Internet Congestion Control*. 1 ed., USA, Birkhäuser.

STOICA, I., SHENKER, S., ZHANG, H., 1998, "Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks". In: *Proceedings of SIGCOMM 1998*, pp.118-130, Vancouver, Canada, Ago.

TANENBAUM, A., 1996, *Computer Networks*. 3 ed., New Jersey, Hall Prentice.

UCLA COMPUTER SCIENCE DEPARTMENT. *TCP WESTWOOD Home Page*. High Performance Internet Lab, UCLA Computer Science Department, University of California, CA, USA. Disponível em: http://www.cs.ucla.edu /NRL/hpi/tcpw. Acesso em: Jun 2008.

VARIAN, H.R., *Economic Issues Facing the Internet*. UC Berkeley School of Information. Disponível em: http://people.ischool.berkeley.edu/~hal/Papers/econ-issues-internet.html, Acesso em: Set 2007.

WALRAND, J., 1998, *Communication Networks: A First Course*. 2 ed. USA, McGraw-Hill.

WANG, R., VALLA, M., SANADIDI, M.Y., et.al., 2002, "Using Adaptive Rate Estimation to Provide Enhanced and Robust Transport over Heterogeneous Networks". In: *Proceedings of the 10th International Conference on Network Protocols*, pp.206-215, Paris, França, Nov.

WEI, D., JIN., C., LOW, S., et.al., 2006, "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Transactions on Networking - TON 2006*, v.14, n.6, (Dez), pp.1246-1259.

WEN, J.T., ARCAK, M., 2004, "A unifying passivity framework for network flow control", *IEEE Transactions on Automatic Control*, v.49, n.2, pp.162-174.

WU-CHANG, F., KANLUR, D.D., SAHA, D., et.al., 2001, "Stochastic fair blue: a queue management algorithm for enforcing fairness". In: *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM 2001*, V.3, pp.1520-1529, Anchorage, AK, USA, Abr.

XU, K., ANSARI, N., 2005, "Stability and Fairness of Rate Estimation-Based AIAD Congestion Control in TCP", *IEEE Communications Letters*, v.9, n.4 (Abr), pp 378-380.

YANG, Y.R., KIM, M.S., ZHANG, X., et.al., 2000, *Two Problems of TCP AIMD Congestion Control*. In: Technical Report TR2000-13, University of Texas, Austin, TX, USA.

Apêndices

A. Síntese do Simulador NS2

O NS ou *network simulator* (também chamado de NS2 em referência a sua geração) é um simulador de redes de computadores popular nos meios acadêmicos por ter o código fonte aberto. O NS fornece a sustentação substancial para a simulação do TCP, do roteamento, e de redes (locais e satélite) *wired* e *wireless* do excesso dos protocolos do *multicast*.

A primeira versão do NS foi desenvolvida em 1989 baseada no *Real Network* Simulator e evoluiu substancialmente nos últimos anos. No NS o desenvolvimento 1995 foi suportado por DARPA com o projeto em LBL, Xerox PARC, UCB, e USC/ISI de VINT. Atualmente o desenvolvimento do NS é sustentação através de DARPA com SAMAN e através do NSF com CONSER, ambos na colaboração com outros investigadores incluindo ACIRI. O NS2 incluiu sempre contribuições substanciais de outros investigadores [8, 55]. A versão atual é a 2 e foi desenvolvida na Universidade de Berkley usando as linguagens C++ e Tcl. Os scripts Tcl são usados para descrever o ambiente a ser simulado, sendo o simulador um interpretador destes scripts que usa as bibliotecas desenvolvidas em C++ as quais possuem os objetos para *scheduling* de eventos e elementos de rede.

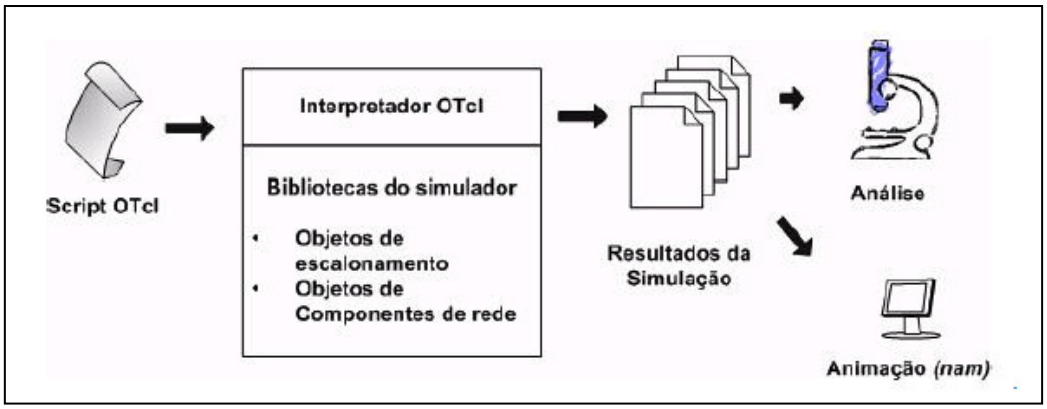


Figura A.1: Vista simplificada de un usuario de NS2.

Como é mostrado na Figura A, na opinião de um usuário simplificado, o NS é o intérprete *object-oriented* do *script* Tcl (OTcl) que tem um *scheduler* ou escalonador de eventos da simulação, umas bibliotecas de objeto componentes da rede, e uns módulos de ajuste de redes. Para usar o NS, se deve um programa na linguagem OTcl, uma linguagem de script onde o usuário dever criar o escalonador de eventos, habilitar o

trace ou rastreio desejado, criar a topologia da rede, estabelecer o roteamento, criar as conexões do transporte, criar geradores de trafego e indicar ao simulador os tempos de inicio e final.

Um usuário comum atua no perímetro “tcl”, escrevendo o scripts em OTcl e executando simulações. Os escalonadores de eventos e os componentes de rede são implementados em C++ e disponibilizados ao interpretador OTcl através de uma replicação feita pela camada tclcl, que recria os objetos C++ em objetos OTcl, e que podem finalmente ser manipulados por esta última (processo denominado *linkage*). Todo o conjunto constitui-se no NS, que é um interpretador de OTcl com bibliotecas de simulação para redes de computadores.

Uso básico do NS2

O uso básico do simulador NS2 (CHUNG e CAYPOOL, 2006) é bastante simples, se creia um arquivo onde se escreve o que se deseja simular (topologia, nós, movimentos, tráfego, etc.) e se passa como parâmetro ao simulador, este gera traços de todos os eventos que foram gerados durante a simulação.

O arquivo de configuração é um script realizado em OTcl e a saída é um (o mais) arquivo de texto (geralmente com extensão “.tr”), adicionalmente se pode indicar que gere um arquivo para o visor gráfico *NAM* ou gerar arquivos adicionais para visualizar em *xgraph*.

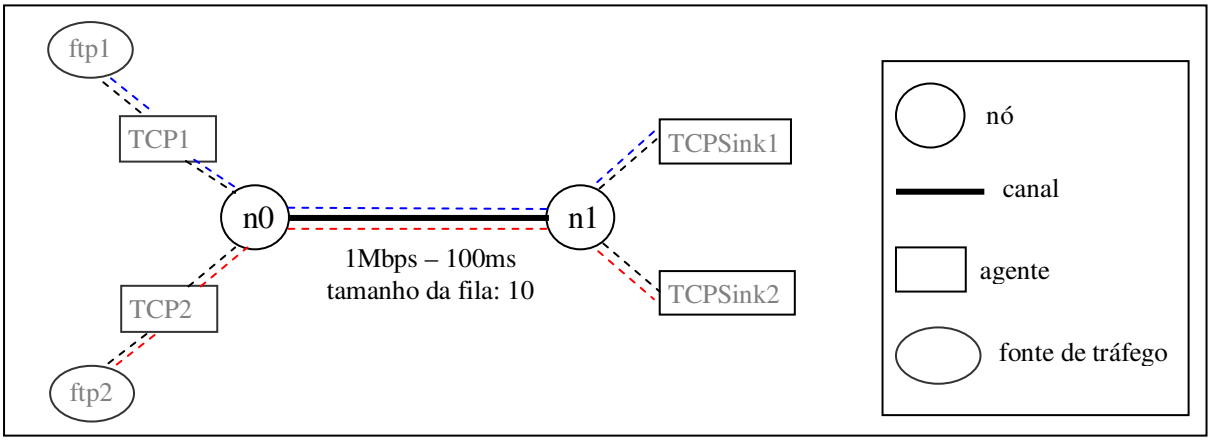


Figura A.2: Topologia de rede do Simple.tcl

Mostrar-se agora um exemplo simples de um script de simulação do NS2 explicando-se cada linha. O Simple.tcl é um script OTcl que cria uma configuração de rede simples e roda o cenário de simulação na Figura 5.2. Esta rede consiste de dois nós (n0, n1) e canal duplex entre eles que têm 1 Mbps de largura de banda e 100 ms de atraso com uma fila DropTail de tamanho máximo é 10. Dois agentes TCP são

anexados no n0, e uma conexão é estabelecida para um agente TCPSink correspondente no n1. Por default, o tamanho máximo do pacote que o agente TCP pode gerar é de 1 KByte. Um agente TCPSink gera e envia pacotes ACK para o remetente (agente TCP) e libera os pacotes recebidos. Geradores de tráfego ftp são anexados para os agentes TCP.

Simple.tcl

```
set ns [new Simulator]              #Criar um objeto simulator

$ns color 1 Blue                    #Definir diferentes colores para fluxos de dados (para NAM)
$ns color 2 Red

set nf [open out.nam w]             #Abrir o arquivo para rastreio do NAM
$ns namtrace-all $nf
set tf [open out6.tr w]             #Abrir o arquivo de TRACE out6.tr  para escritura
$ns trace-all $tf

set n0 [$ns node]                   #Criar dois nós n0 e n1
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 0ms DropTail      #Criar canal entre os dois nós
$ns queue-limit $n0 $n1 10          #Estabelecer o tamanho da fila do canal (n0-n1) a 10
$ns duplex-link-op $n0 $n1 orient right       #Estabelecer posição dos nós para o NAM.
$ns duplex-link-op $n0 $n1 queuePos 0.5       #Monitorar a fila do canal (n0-n1) para NAM

#Estabelecer a primeira conexão TCP
set tcp1 [new Agent/TCP]            #Criar o agente TCP1
$tcp1 set class_ 2
$ns attach-agent $n0 $tcp1          #Anexar o agente TCP1 ao nó 0
set sink1 [new Agent/TCPSink]       #Criar o agente TCPSink1
$ns attach-agent $n1 $sink1         #Anexar o agente TCPSink1 ao nó 1
$ns connect $tcp1 $sink1            #Conectar os agentes TCP1 e TCPSink1
$tcp1 set fid_ 1                    #Dar a cor Blue a primeira conexão (para o NAM)

# Estabelecer a FTP sobre a conexão TCP1
set ftp1 [new Application/FTP]      #Criar uma fonte de trafego tipo FTP
$ftp1 attach-agent $tcp1            #Anexar o a fonte de trago ftp1 ao agente TCP1
$ftp1 set type_ FTP

#Estabelecer a segunda conexão TCP
set tcp2 [new Agent/TCP]            #Criar o agente TCP2
$tcp2 set class_ 2
$ns attach-agent $n0 $tcp2          #Anexar o agente TCP2 ao nó 0
set sink2 [new Agent/TCPSink]       #Criar o agente TCPSink2
$ns attach-agent $n1 $sink2         #Anexar o agente TCPSink2 ao nó 1
$ns connect $tcp2 $sink2            #Conectar os agentes TCP2 e TCPSink2
$tcp2 set fid_ 2                    #Dar a cor Red a segunda conexão (para o NAM)

# Estabelecer a FTP sobre a conexão TCP2
set ftp2 [new Application/FTP]      #Criar uma fonte de trafego tipo FTP
$ftp2 attach-agent $tcp2            #Anexar o a fonte de trago ftp2 ao agente TCP2
$ftp2 set type_ FTP

set tcptrace1 [open tcp-trace1.tr w] #Abrir um arquivo tcp-trace1.tr para rastrear tamanho da
$tcp1 attach-trace [set tcptrace1]   #janela do TCP1 (variável cwnd_)
$tcp1 tracevar cwnd_                 #Rastrear a variável cwnd_

set tcptrace2 [open tcp-trace2.tr w] #Abrir um arquivo tcp-trace2.tr para rastrear tamanho da
$tcp2 attach-trace [set tcptrace2]   #janela do TCP2 (variável cwnd_)
$tcp2 tracevar cwnd_                 #Rastrear a variável cwnd_
```

Simple.tcl – continuação

```
#Eventos para o Schedule dos agentes FTP
$ns at 0.0 "$ftp1 start"            #Iniciar fonte de trafego ftp1 aos 0.0 segundos
$ns at 0.0 "$ftp2 start"            #Iniciar fonte de trafego ftp2 aos 0.0 segundos
$ns at 10.0 "$ftp1 stop"            #Parar fonte de trafego ftp1 aos 10 segundos
$ns at 10.0 "$ftp2 stop"            #Parar fonte de trafego ftp1 aos 10 segundos

#Estabelecer o scheduler para separar os agentes tcp e sink (opcional)
$ns at 10.0 "$ns detach-agent $n0 $tcp1 ; $ns detach-agent $n0 $sink1"
$ns at 10.0 "$ns detach-agent $n0 $tcp2 ; $ns detach-agent $n0 $sink2"

#Chamar ao procedimento finish
$ns at 10.0 "finish"

#Definir um procedimento 'finish'
proc finish {}
    {
    global ns nf tf          #variáveis ns, nf, e tf
    $ns flush-trace
    #Fechar o arquivo TRACE
    close $tf
    #Fechar o arquivo de rastreio de NAM
    close $nf
    #Executar NAM sobre o arquivo de rastreio
    exec nam out.nam &
    exit 0
    }

#Rodar a simulação
$ns run
```

Para simular o usuário deve escrever no *prompt* a sentencia “ns Simple.tcl”. Os arquivos de rastreio ou *tracing* são gerados com registro de cada evento simulado. Um ponto importante a observar é que o NS2 não fornece estatísticas de simulação de modo automático; estas devem ser obtidas através de procedimentos matemáticos no script ou pela manipulação de objetos especiais chamados monitores. Pode-se, ainda, usar ferramentas para análise dos arquivos de *tracing* gerados durante a simulação, que são os verdadeiros resultados da simulação, conforme a Figura A.1. Estes arquivos registram cada evento gerado pelos escalonadores. O animador NAM pode também ser usado para analisar visualmente a simulação e obter algumas estatísticas, mas ele não é apropriado para análises mais profundas. Se nada for feito, o simulador apenas rodará o script, gerará os arquivos de saída (*tracing*) e terminará sem nada mostrar ao usuário.

B. Tabelas varias

Tabela **tvel_total** de resultados de três usuários que iniciam a transmissão ao mesmo e o valor do tempo é guardado cada 100 pkts para o estudo da transferência de pacotes no tempo.

NR			WP			WVE			VE		
time	0.50718	b_trans 136,000	time	0.50718	b_trans 136,000	time	0.91979	B_trans 136,000	time	1.15202	B_trans 136,000
time	1.40918	b_trans 272,000	time	1.62734	b_trans 272,000	time	1.62315	B_trans 272,000	time	1.8061	B_trans 272,000
time	1.88182	b_trans 408,000	time	2.24782	b_trans 408,000	time	2.25259	B_trans 408,000	time	2.43554	B_trans 408,000
time	2.46422	b_trans 544,000	time	2.79438	b_trans 544,000	time	3.21443	B_trans 544,000	time	3.19392	B_trans 544,000
time	3.08981	b_trans 680,000	time	4.13155	b_trans 680,000	time	4.33757	B_trans 680,000	time	4.34331	B_trans 680,000
time	3.71925	b_trans 816,000	time	5.54349	b_trans 816,000	time	5.12307	B_trans 816,000	time	5.13365	B_trans 816,000
time	4.40693	b_trans 952,000	time	9.93029	b_trans 952,000	time	6.29898	B_trans 952,000	time	5.71381	B_trans 952,000
time	5.00949	b_trans 1,088,000	time	11.57918	b_trans 1,088,000	time	7.37856	B_trans 1,088,000	time	6.20499	B_trans 1,088,000
time	5.66805	b_trans 1,224,000	time	15.26226	b_trans 1,224,000	time	8.45752	B_trans 1,224,000	time	6.75442	B_trans 1,224,000
time	6.29525	b_trans 1,360,000	time	16.13424	b_trans 1,360,000	time	9.65358	B_trans 1,360,000	time	7.2568	B_trans 1,360,000
time	6.93141	b_trans 1,496,000	time	16.78446	b_trans 1,496,000	time	10.51238	B_trans 1,496,000	time	7.87118	B_trans 1,496,000
time	7.60565	b_trans 1,632,000	time	17.2459	b_trans 1,632,000	time	11.58525	B_trans 1,632,000	time	8.71469	B_trans 1,632,000
time	8.22165	b_trans 1,768,000	time	17.74157	b_trans 1,768,000	time	12.77011	B_trans 1,768,000	time	9.46085	B_trans 1,768,000
time	8.92725	b_trans 1,904,000	time	18.23437	b_trans 1,904,000	time	13.88144	B_trans 1,904,000	time	10.35102	B_trans 1,904,000
time	9.52085	b_trans 2,040,000	time	18.73389	b_trans 2,040,000	time	15.06854	B_trans 2,040,000	time	10.8259	B_trans 2,040,000
time	10.25109	b_trans 2,176,000	time	19.34989	b_trans 2,176,000	time	15.84957	B_trans 2,176,000	time	11.3187	B_trans 2,176,000
time	10.90741	b_trans 2,312,000	time	19.98443	b_trans 2,312,000	time	16.92019	B_trans 2,312,000	time	11.74654	B_trans 2,312,000
time	11.67573	b_trans 2,448,000				time	18.1073	B_trans 2,448,000	time	12.24606	B_trans 2,448,000
time	12.32981	b_trans 2,584,000	time	1.35381	b_trans 136,000	time	19.19522	B_trans 2,584,000	time	12.73214	B_trans 2,584,000
time	12.95701	b_trans 2,720,000	time	2.07086	b_trans 272,000				time	13.26526	B_trans 2,720,000
time	13.62677	b_trans 2,856,000	time	2.68686	b_trans 408,000	time	1.03851	B_trans 136,000	time	13.75358	B_trans 2,856,000
time	14.29653	b_trans 2,992,000	time	3.81795	b_trans 544,000	time	1.67243	B_trans 272,000	time	14.21054	B_trans 2,992,000
time	14.93493	b_trans 3,128,000	time	5.20301	b_trans 680,000	time	2.30187	B_trans 408,000	time	14.63838	B_trans 3,128,000
time	15.60469	b_trans 3,264,000	time	6.05931	b_trans 816,000	time	3.27043	B_trans 544,000	time	15.11102	B_trans 3,264,000
time	16.26549	b_trans 3,400,000	time	6.65963	b_trans 952,000	time	4.33981	B_trans 680,000	time	15.61054	B_trans 3,400,000
time	16.88149	b_trans 3,536,000	time	7.13675	b_trans 1,088,000	time	5.5253	B_trans 816,000	time	16.15262	B_trans 3,536,000
time	17.58485	b_trans 3,672,000	time	7.52938	b_trans 1,224,000	time	6.33706	B_trans 952,000	time	16.62302	B_trans 3,672,000
time	18.26805	b_trans 3,808,000	time	7.99082	b_trans 1,360,000	time	7.4144	B_trans 1,088,000	time	17.07326	B_trans 3,808,000
time	18.91093	b_trans 3,944,000	time	8.4372	b_trans 1,496,000	time	8.60374	B_trans 1,224,000	time	17.56382	B_trans 3,944,000
time	19.56501	b_trans 4,080,000	time	8.874	b_trans 1,632,000	time	9.69229	B_trans 1,360,000	time	18.06782	B_trans 4,080,000
			time	9.29288	b_trans 1,768,000	time	10.88325	B_trans 1,496,000	time	18.52478	B_trans 4,216,000
time	1.12794	b_trans 136,000	time	9.82886	b_trans 1,904,000	time	11.74205	B_trans 1,632,000	time	19.0467	B_trans 4,352,000
time	1.88854	b_trans 272,000	time	10.11136	b_trans 2,040,000	time	12.80371	B_trans 1,768,000	time	19.50142	B_trans 4,488,000
time	2.52694	b_trans 408,000	time	10.40928	b_trans 2,176,000	time	13.99978	B_trans 1,904,000	time	19.99198	B_trans 4,624,000
time	3.21077	b_trans 544,000	time	10.87072	b_trans 2,312,000	time	15.07078	B_trans 2,040,000			
time	3.90741	b_trans 680,000	time	11.28574	b_trans 2,448,000	time	16.26909	B_trans 2,176,000	time	1.1789	B_trans 136,000
time	4.60853	b_trans 816,000	time	11.63966	b_trans 2,584,000	time	17.09267	B_trans 2,312,000	time	1.84418	B_trans 272,000
time	5.19989	b_trans 952,000	time	12.10334	b_trans 2,720,000	time	18.2641	B_trans 2,448,000	time	2.48034	B_trans 408,000
time	6.01525	b_trans 1,088,000	time	12.39678	b_trans 2,856,000	time	19.34144	B_trans 2,584,000	time	3.12222	B_trans 544,000
time	6.59541	b_trans 1,224,000	time	12.7619	b_trans 2,992,000				time	3.63968	B_trans 680,000
time	7.27637	b_trans 1,360,000	time	13.21277	b_trans 3,128,000	time	1.07659	B_trans 136,000	time	4.25432	B_trans 816,000
time	7.97301	b_trans 1,496,000	time	13.48792	b_trans 3,264,000	time	1.71723	B_trans 272,000	time	4.92085	B_trans 952,000
time	8.61365	b_trans 1,632,000	time	13.72723	b_trans 3,400,000	time	2.48107	B_trans 408,000	time	6.05267	B_trans 1,088,000
time	9.27893	b_trans 1,768,000	time	14.01395	b_trans 3,536,000	time	2.99976	B_trans 544,000	time	7.39346	B_trans 1,224,000
time	9.96437	b_trans 1,904,000	time	14.35443	b_trans 3,672,000	time	3.54371	B_trans 680,000	time	8.1915	B_trans 1,360,000
time	10.60501	b_trans 2,040,000	time	14.78451	b_trans 3,808,000	time	4.09949	B_trans 816,000	time	9.29733	B_trans 1,496,000
time	11.24117	b_trans 2,176,000	time	15.21522	b_trans 3,944,000	time	4.8015	B_trans 952,000	time	9.66469	B_trans 1,632,000
time	11.91989	b_trans 2,312,000	time	15.7848	b_trans 4,080,000	time	5.29555	B_trans 1,088,000	time	10.02048	B_trans 1,768,000
time	12.53589	b_trans 2,448,000	time	16.40366	b_trans 4,216,000	time	5.99533	B_trans 1,224,000	time	11.25822	B_trans 1,904,000
time	13.23925	b_trans 2,584,000	time	16.83822	b_trans 4,352,000	time	6.50954	B_trans 1,360,000	time	12.43422	B_trans 2,040,000
time	13.92245	b_trans 2,720,000	time	17.21678	b_trans 4,488,000	time	7.26146	B_trans 1,496,000	time	13.85438	B_trans 2,176,000
time	14.66165	b_trans 2,856,000	time	17.77293	b_trans 4,624,000	time	7.73982	B_trans 1,632,000	time	15.30814	B_trans 2,312,000
time	15.34485	b_trans 2,992,000	time	18.18957	b_trans 4,760,000	time	8.4508	B_trans 1,768,000	time	16.59614	B_trans 2,448,000
time	15.99445	b_trans 3,128,000	time	18.59053	b_trans 4,896,000	time	8.95157	B_trans 1,904,000	time	17.78782	B_trans 2,584,000
time	16.63509	b_trans 3,264,000	time	19.01837	b_trans 5,032,000	time	9.64686	B_trans 2,040,000	time	18.9571	B_trans 2,720,000
time	17.32501	b_trans 3,400,000	time	19.34093	b_trans 5,168,000	time	10.15211	B_trans 2,176,000			
time	17.93877	b_trans 3,536,000	time	19.85613	b_trans 5,304,000	time	10.83621	B_trans 2,312,000	time	1.20578	B_trans 136,000
time	18.68469	b_trans 3,672,000				time	11.36386	B_trans 2,448,000	time	1.88226	B_trans 272,000
time	19.33877	b_trans 3,808,000	time	1.37173	b_trans 136,000	time	12.11578	B_trans 2,584,000	time	2.53858	B_trans 408,000
time	19.90101	b_trans 3,944,000	time	2.10446	b_trans 272,000	time	12.5919	B_trans 2,720,000	time	3.48512	B_trans 544,000
			time	2.73166	b_trans 408,000	time	13.31856	B_trans 2,856,000	time	3.94109	B_trans 680,000
time	1.14138	b_trans 136,000	time	3.34605	b_trans 544,000	time	13.80365	B_trans 2,992,000	time	4.40888	B_trans 816,000
time	1.93782	b_trans 272,000	time	3.68293	b_trans 680,000	time	14.52358	B_trans 3,128,000	time	4.83062	B_trans 952,000
time	2.71734	b_trans 408,000	time	4.04867	b_trans 816,000	time	15.00419	B_trans 3,264,000	time	5.30837	B_trans 1,088,000
time	3.44597	b_trans 544,000	time	4.53986	b_trans 952,000	time	15.79581	B_trans 3,400,000	time	5.83987	B_trans 1,224,000
time	4.10229	b_trans 680,000	time	4.84413	b_trans 1,088,000	time	16.24344	B_trans 3,536,000	time	6.35794	B_trans 1,360,000
time	4.76533	b_trans 816,000	time	5.11565	b_trans 1,224,000	time	17.00755	B_trans 3,672,000	time	6.90674	B_trans 1,496,000
time	5.48661	b_trans 952,000	time	5.48749	b_trans 1,360,000	time	17.44398	B_trans 3,808,000	time	7.39344	B_trans 1,632,000
time	6.16309	b_trans 1,088,000	time	5.88621	b_trans 1,496,000	time	18.2081	B_trans 3,944,000	time	7.92718	B_trans 1,768,000
time	6.77909	b_trans 1,224,000	time	6.31019	b_trans 1,632,000	time	18.65573	B_trans 4,080,000	time	8.42509	B_trans 1,904,000
time	7.56309	b_trans 1,360,000	time	6.70891	b_trans 1,768,000	time	19.48704	B_trans 4,216,000	time	8.81224	B_trans 2,040,000
time	8.17909	b_trans 1,496,000	time	7.20234	b_trans 1,904,000	time	19.88378	B_trans 4,352,000	time	9.26373	B_trans 2,176,000
time	8.98773	b_trans 1,632,000	time	7.6705	b_trans 2,040,000				time	10.2272	B_trans 2,312,000
time	9.60149	b_trans 1,768,000	time	8.20362	b_trans 2,176,000				time	10.77662	B_trans 2,448,000
time	10.31157	b_trans 1,904,000	time	8.69704	b_trans 2,312,000				time	11.2515	B_trans 2,584,000
time	10.89173	b_trans 2,040,000	time	9.09352	b_trans 2,448,000				time	11.8227	B_trans 2,720,000
time	11.58165	b_trans 2,176,000	time	9.56616	b_trans 2,584,000				time	12.3043	B_trans 2,856,000
time	12.19765	b_trans 2,312,000	time	10.79456	b_trans 2,720,000				time	12.86206	B_trans 2,992,000
time	12.99509	b_trans 2,448,000	time	11.43582	b_trans 2,856,000				time	13.41758	B_trans 3,128,000
time	13.59765	b_trans 2,584,000	time	11.93086	b_trans 2,992,000				time	13.93054	B_trans 3,264,000
time	14.26069	b_trans 2,720,000	time	12.5715	b_trans 3,128,000				time	14.45918	B_trans 3,400,000
time	14.90805	b_trans 2,856,000	time	13.82355	b_trans 3,264,000				time	14.94974	B_trans 3,536,000
time	15.54421	b_trans 2,992,000	time	14.55155	b_trans 3,400,000				time	15.51422	B_trans 3,672,000
time	16.22293	b_trans 3,128,000	time	15.19506	b_trans 3,536,000				time	15.99582	B_trans 3,808,000
time	16.93301	b_trans 3,264,000	time	15.90128	b_trans 3,672,000				time	16.54014	B_trans 3,944,000
time	17.64533	b_trans 3,400,000	time	17.40942	b_trans 3,808,000				time	17.0419	B_trans 4,080,000
time	18.26133	b_trans 3,536,000							time	17.60414	B_trans 4,216,000
time	18.96917	b_trans 3,672,000							time	18.12606	B_trans 4,352,000
time	19.63893	b_trans 3,808,000							time	18.67038	B_trans 4,488,000
									time	19.23262	B_trans 4,624,000
									time	19.73438	B_trans 4,760,000

Tabela **tvel_parcial** de transmissão até o menor valor alcançado por algum TCPs (indicado pelos quadros laranjas na tabela tvel_total)

NR			WP			WVE			VE		
time	0.50718	b_trans 136,000	time	0.50718	b_trans 136,000	time	0.91979	B_trans 136,000	time	1.15202	B_trans 136,000
time	1.40918	b_trans 272,000	time	1.62734	b_trans 272,000	time	1.62315	B_trans 272,000	time	1.8061	B_trans 272,000
time	1.88182	b_trans 408,000	time	2.24782	b_trans 408,000	time	2.25259	B_trans 408,000	time	2.43554	B_trans 408,000
time	2.46422	b_trans 544,000	time	2.79438	b_trans 544,000	time	3.21443	B_trans 544,000	time	3.19392	B_trans 544,000
time	3.08981	b_trans 680,000	time	4.13155	b_trans 680,000	time	4.33757	B_trans 680,000	time	4.34331	B_trans 680,000
time	3.71925	b_trans 816,000	time	5.54349	b_trans 816,000	time	5.12307	B_trans 816,000	time	5.13365	B_trans 816,000
time	4.40693	b_trans 952,000	time	9.93029	b_trans 952,000	time	6.29898	B_trans 952,000	time	5.71381	B_trans 952,000
time	5.00949	b_trans 1,088,000	time	11.57918	b_trans 1,088,000	time	7.37856	B_trans 1,088,000	time	6.20499	B_trans 1,088,000
time	5.66805	b_trans 1,224,000	time	15.26226	b_trans 1,224,000	time	8.45752	B_trans 1,224,000	time	6.75442	B_trans 1,224,000
time	6.29525	b_trans 1,360,000	time	16.13424	b_trans 1,360,000	time	9.65358	B_trans 1,360,000	time	7.2568	B_trans 1,360,000
time	6.93141	b_trans 1,496,000	time	16.78446	b_trans 1,496,000	time	10.51238	B_trans 1,496,000	time	7.87118	B_trans 1,496,000
time	7.60565	b_trans 1,632,000	time	17.2459	b_trans 1,632,000	time	11.58525	B_trans 1,632,000	time	8.71469	B_trans 1,632,000
time	8.22165	b_trans 1,768,000	time	17.74157	b_trans 1,768,000	time	12.77011	B_trans 1,768,000	time	9.46085	B_trans 1,768,000
time	8.92725	b_trans 1,904,000	time	18.23437	b_trans 1,904,000	time	13.88144	B_trans 1,904,000	time	10.35102	B_trans 1,904,000
time	9.52085	b_trans 2,040,000	time	18.73389	b_trans 2,040,000	time	15.06854	B_trans 2,040,000	time	10.8259	B_trans 2,040,000
time	10.25109	b_trans 2,176,000	time	19.34989	b_trans 2,176,000	time	15.84957	B_trans 2,176,000	time	11.3187	B_trans 2,176,000
time	10.90741	b_trans 2,312,000	time	19.98443	b_trans 2,312,000	time	16.92019	B_trans 2,312,000	time	11.74654	B_trans 2,312,000
time	1.12794	b_trans 136,000	time	1.35381	b_trans 136,000	time	1.03851	B_trans 136,000	time	1.1789	B_trans 136,000
time	1.88854	b_trans 272,000	time	2.07086	b_trans 272,000	time	1.67243	B_trans 272,000	time	1.84418	B_trans 272,000
time	2.52694	b_trans 408,000	time	2.68686	b_trans 408,000	time	2.30187	B_trans 408,000	time	2.48034	B_trans 408,000
time	3.21077	b_trans 544,000	time	3.81795	b_trans 544,000	time	3.27043	B_trans 544,000	time	3.12222	B_trans 544,000
time	3.90741	b_trans 680,000	time	5.20301	b_trans 680,000	time	4.33981	B_trans 680,000	time	3.63968	B_trans 680,000
time	4.60853	b_trans 816,000	time	6.05931	b_trans 816,000	time	5.5253	B_trans 816,000	time	4.25432	B_trans 816,000
time	5.19989	b_trans 952,000	time	6.65963	b_trans 952,000	time	6.33706	B_trans 952,000	time	4.92085	B_trans 952,000
time	6.01525	b_trans 1,088,000	time	7.13675	b_trans 1,088,000	time	7.4144	B_trans 1,088,000	time	6.05267	B_trans 1,088,000
time	6.59541	b_trans 1,224,000	time	7.52938	b_trans 1,224,000	time	8.60374	B_trans 1,224,000	time	7.39346	B_trans 1,224,000
time	7.27637	b_trans 1,360,000	time	7.99082	b_trans 1,360,000	time	9.69229	B_trans 1,360,000	time	8.1915	B_trans 1,360,000
time	7.97301	b_trans 1,496,000	time	8.4372	b_trans 1,496,000	time	10.88325	B_trans 1,496,000	time	9.29733	B_trans 1,496,000
time	8.61365	b_trans 1,632,000	time	8.874	b_trans 1,632,000	time	11.74205	B_trans 1,632,000	time	9.66469	B_trans 1,632,000
time	9.27893	b_trans 1,768,000	time	9.29288	b_trans 1,768,000	time	12.80371	B_trans 1,768,000	time	10.02048	B_trans 1,768,000
time	9.96437	b_trans 1,904,000	time	9.82886	b_trans 1,904,000	time	13.99978	B_trans 1,904,000	time	11.25822	B_trans 1,904,000
time	10.60501	b_trans 2,040,000	time	10.11136	b_trans 2,040,000	time	15.07078	B_trans 2,040,000	time	12.43422	B_trans 2,040,000
time	11.24117	b_trans 2,176,000	time	10.40928	b_trans 2,176,000	time	16.26909	B_trans 2,176,000	time	13.85438	B_trans 2,176,000
time	11.91989	b_trans 2,312,000	time	10.87072	b_trans 2,312,000	time	17.09267	B_trans 2,312,000	time	15.30814	B_trans 2,312,000
time	12.53589	b_trans 2,448,000	time	11.28574	b_trans 2,448,000	time	18.2641	B_trans 2,448,000	time	16.59614	B_trans 2,448,000
time	13.23925	b_trans 2,584,000	time	11.63966	b_trans 2,584,000	time	19.34144	B_trans 2,584,000	time	17.78782	B_trans 2,584,000
time	1.14138	b_trans 136,000	time	1.37173	b_trans 136,000	time	1.07659	B_trans 136,000	time	1.20578	B_trans 136,000
time	1.93782	b_trans 272,000	time	2.10446	b_trans 272,000	time	1.71723	B_trans 272,000	time	1.88226	B_trans 272,000
time	2.71734	b_trans 408,000	time	2.73166	b_trans 408,000	time	2.48107	B_trans 408,000	time	2.53858	B_trans 408,000
time	3.44597	b_trans 544,000	time	3.34605	b_trans 544,000	time	2.99976	B_trans 544,000	time	3.48512	B_trans 544,000
time	4.10229	b_trans 680,000	time	3.68293	b_trans 680,000	time	3.54371	B_trans 680,000	time	3.94109	B_trans 680,000
time	4.76533	b_trans 816,000	time	4.04867	b_trans 816,000	time	4.09949	B_trans 816,000	time	4.40888	B_trans 816,000
time	5.48661	b_trans 952,000	time	4.53986	b_trans 952,000	time	4.8015	B_trans 952,000	time	4.83062	B_trans 952,000
time	6.16309	b_trans 1,088,000	time	4.84413	b_trans 1,088,000	time	5.29555	B_trans 1,088,000	time	5.30837	B_trans 1,088,000
time	6.77909	b_trans 1,224,000	time	5.11565	b_trans 1,224,000	time	5.99533	B_trans 1,224,000	time	5.83987	B_trans 1,224,000
time	7.56309	b_trans 1,360,000	time	5.48749	b_trans 1,360,000	time	6.50954	B_trans 1,360,000	time	6.35794	B_trans 1,360,000
time	8.17909	b_trans 1,496,000	time	5.88621	b_trans 1,496,000	time	7.26146	B_trans 1,496,000	time	6.90674	B_trans 1,496,000
time	8.98773	b_trans 1,632,000	time	6.31019	b_trans 1,632,000	time	7.73982	B_trans 1,632,000	time	7.39344	B_trans 1,632,000
time	9.60149	b_trans 1,768,000	time	6.70891	b_trans 1,768,000	time	8.4508	B_trans 1,768,000	time	7.92718	B_trans 1,768,000
time	10.31157	b_trans 1,904,000	time	7.20234	b_trans 1,904,000	time	8.95157	B_trans 1,904,000	time	8.42509	B_trans 1,904,000
time	10.89173	b_trans 2,040,000	time	7.6705	b_trans 2,040,000	time	9.64686	B_trans 2,040,000	time	8.81224	B_trans 2,040,000
time	11.58165	b_trans 2,176,000	time	8.20362	b_trans 2,176,000	time	10.15211	B_trans 2,176,000	time	9.26373	B_trans 2,176,000
time	12.19765	b_trans 2,312,000	time	8.69704	b_trans 2,312,000	time	10.83621	B_trans 2,312,000	time	10.2272	B_trans 2,312,000
time	12.99509	b_trans 2,448,000	time	9.09352	b_trans 2,448,000	time	11.36386	B_trans 2,448,000	time	10.77662	B_trans 2,448,000
time	13.59765	b_trans 2,584,000	time	9.56616	b_trans 2,584,000	time	12.11578	B_trans 2,584,000	time	11.2515	B_trans 2,584,000
time	14.26069	b_trans 2,720,000	time	10.79456	b_trans 2,720,000	time	12.5919	B_trans 2,720,000	time	11.8227	B_trans 2,720,000
time	14.90805	b_trans 2,856,000	time	11.43582	b_trans 2,856,000	time	13.31856	B_trans 2,856,000	time	12.3043	B_trans 2,856,000
time	15.54421	b_trans 2,992,000	time	11.93086	b_trans 2,992,000	time	13.80365	B_trans 2,992,000	time	12.86206	B_trans 2,992,000
time	16.22293	b_trans 3,128,000	time	12.5715	b_trans 3,128,000	time	14.52358	B_trans 3,128,000	time	13.41758	B_trans 3,128,000
time	16.93301	b_trans 3,264,000	time	13.82355	b_trans 3,264,000	time	15.00419	B_trans 3,264,000	time	13.93054	B_trans 3,264,000
time	17.64533	b_trans 3,400,000	time	14.55155	b_trans 3,400,000	time	15.79581	B_trans 3,400,000	time	14.45918	B_trans 3,400,000
time	18.26133	b_trans 3,536,000	time	15.19506	b_trans 3,536,000	time	16.24344	B_trans 3,536,000	time	14.94974	B_trans 3,536,000
time	18.96917	b_trans 3,672,000	time	15.90128	b_trans 3,672,000	time	17.00755	B_trans 3,672,000	time	15.51422	B_trans 3,672,000
time	19.63893	b_trans 3,808,000	time	17.40942	b_trans 3,808,000	time	17.44398	B_trans 3,808,000	time	15.99582	B_trans 3,808,000

Resultados de rodadas para a Tabela 5.3

	TCPNR- 3 usuarios - inicio: 0s. - fin: 20s.		
	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	11,927,200.00	231,200.00	587,738.69
Rodada 2	11,934,000.00	231,299.00	588,075.43
Rodada 3	11,927,200.00	231,200.00	587,738.69
Rodada 4	11,927,200.00	231,200.00	587,738.69
Rodada 5	11,934,000.00	231,299.00	588,075.43
Rodada 6	11,927,200.00	231,200.00	587,738.69
Rodada 7	11,927,200.00	231,200.00	587,738.69
Rodada 8	11,927,200.00	231,200.00	587,738.69
Rodada 9	11,927,200.00	231,200.00	587,738.69
Rodada 10	11,927,200.00	231,200.00	587,738.69
	11,928,560.00	231,219.80	587,806.04

	TCPW - 3 usuarios - inicio: 0s. - fin: 20s.		
	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	11,527,360.00	425,680.00	557,873.37
Rodada 2	11,483,840.00	414,800.00	556,233.17
Rodada 3	11,234,960.00	452,880.00	541,813.07
Rodada 4	11,490,640.00	425,680.00	556,028.14
Rodada 5	11,259,440.00	454,240.00	542,974.87
Rodada 6	11,486,560.00	414,800.00	556,369.85
Rodada 7	11,527,360.00	425,600.00	557,877.39
Rodada 8	11,259,440.00	454,240.00	542,974.87
Rodada 9	11,331,520.00	428,400.00	547,895.48
Rodada 10	11,500,160.00	414,800.00	557,053.27
	11,410,128.00	431,112.00	551,709.35

	TCPWVE - 3 usuarios - inicio: 0s. - fin: 20s.		
	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	9,658,720.00	2,006,000.00	384,558.79
Rodada 2	12,008,800.00	784,720.00	564,024.12
Rodada 3	12,148,880.00	563,040.00	582,203.02
Rodada 4	12,203,280.00	705,840.00	577,760.80
Rodada 5	1,212,800.00	690,880.00	26,227.14
Rodada 6	11,396,800.00	1,185,920.00	513,109.55
Rodada 7	12,117,600.00	783,360.00	569,559.80
Rodada 8	12,116,240.00	761,600.00	570,584.92
Rodada 9	12,117,600.00	794,240.00	569,013.07
Rodada 10	12,114,880.00	761,600.00	570,516.58
	10,709,560.00	903,720.00	492,755.78

TCPVE - 3 usuarios - inicio: 0s. - fin: 20s.

	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	12,112,160.00	552,160.00	580,904.52
Rodada 2	12,178,800.00	456,960.00	589,037.19
Rodada 3	12,087,680.00	427,040.00	585,961.81
Rodada 4	12,102,640.00	825,520.00	566,689.45
Rodada 5	12,116,240.00	446,080.00	586,440.20
Rodada 6	12,195,120.00	456,960.00	589,857.29
Rodada 7	12,102,640.00	825,520.00	566,689.45
Rodada 8	11,906,800.00	1,260,720.00	534,978.89
Rodada 9	12,044,160.00	455,600.00	582,339.70
Rodada 10	12,269,920.00	923,440.00	570,174.87
	12,111,616.00	663,000.00	575,307.34

Resultados de simulações para a Tabela 5.5

				TCPNR- 3 usuarios - inicio: aleatorio - fin: 20s.		
	t(0)tcp1	t(0)tcp2	t(0)tcp3	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	0	0	0	1,193,400.00	231,200.00	48,351.76
Rodada 2	0	0.05	0.1	1,193,260.00	228,480.00	48,481.41
Rodada 3	0	0.1	0.2	11,905,440.00	197,200.00	588,353.77
Rodada 4	0	5	5	11,696,000.00	206,720.00	577,350.75
Rodada 5	0.5	1	10	11,436,240.00	300,560.00	559,581.91
Rodada 6	0	4.5	11	11,720,480.00	320,960.00	572,840.20
Rodada 8	0	4.3	9.1	11,739,520.00	212,160.00	579,264.32
Rodada 9	0	4.3	4.3	11,728,640.00	218,960.00	578,375.88
				9,076,622.50	239,530.00	444,075.00

				TCPWVE - 3 usuarios - inicio:aleatorio - fin: 20s.		
	t(0)tcp1	t(0)tcp2	t(0)tcp3	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	0	0	0	9,658,720.00	2,006,000.00	384,558.79
Rodada 2	0	0.05	0.1	11,114,880.00	578,000.00	529,491.46
Rodada 3	0	0.1	0.2	12,050,960.00	579,360.00	576,462.31
Rodada 4	0	5	5	11,599,440.00	833,680.00	540,992.96
Rodada 5	0.5	1	10	11,034,000.00	614,720.00	523,581.91
Rodada 6	0	4.5	11	11,111,200.00	489,600.00	533,748.74
Rodada 8	0	4.3	9.1	11,297,520.00	235,280.00	555,891.46
Rodada 9	0	4.3	4.3	11,618,400.00	734,400.00	546,934.67
				11,185,640.00	758,880.00	523,957.79

				TCPVE - 3 usuarios - inicio: aleatorio - fin: 20s.		
	t(0)tcp1	t(0)tcp2	t(0)tcp3	TRANS EFETIVA	RETANS	GOODPUT
Rodada 1	0	0	0	12,112,160.00	552,160.00	580,904.52
Rodada 2	0	0.05	0.1	11,458,000.00	1,959,760.00	477,298.49
Rodada 3	0	0.1	0.2	12,117,600.00	874,480.00	564,980.90
Rodada 4	0	5	5	11,148,640.00	1,021,360.00	508,908.54
Rodada 5	0.5	1	10	10,805,200.00	1,109,400.00	487,226.13
Rodada 6	0	4.5	11	10,537,280.00	688,160.00	494,930.65
Rodada 8	0	4.3	9.1	10,760,320.00	856,800.00	497,664.32
Rodada 9	0	4.3	4.3	11,051,360.00	973,760.00	506,412.06
				11,248,820.00	1,004,485.00	514,790.70