Uma Ferramenta para Comparação dos Mecanismos de Controle de Congestionamento em Redes TCP/IP Utilizando o NS

Luiz Antonio Rodrigues[1], Michele Mara de A. E. Lima[1]

[1]Colegiado de Informática – Universidade Estadual do Oeste do Paraná (UNIOESTE)
Caixa Postal 711 – 85.810-110 – Cascavel – PR – Brasil

{larodrig,michele}@unioeste.br

Resumo. *Neste trabalho é apresentada uma ferramenta que através de simulações permite ao usuário a avaliação conjunta de implementações TCP, com e sem suporte a ECN e políticas de AQM, em diferentes cenários de rede, nos quais, podem ser feitas variações de tipo de tráfego e escolha das métricas de eficiência a serem utilizadas nas avaliações, tais como: throughput, goodput, taxa de perda, atraso, jitter, etc. A ferramenta foi desenvolvida de forma a permitir que a execução das simulações e a recuperação dos resultados gerados pudessem ser feitas via Web. Para tanto, foram utilizadas as seguintes tecnologias: JSP (Java Server Pages), o servidor Tomcat 4.1.10 e o simulador de redes NS (Network Simulator).*

1 Introdução

O congestionamento em redes de computadores ocorre quando há insuficiência de recursos para acomodar a carga presente na rede ou quando ocorre um desbalanceamento do tráfego nos nós 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. Além disto, os recursos utilizados pelos pacotes que foram descartados são desperdiçados e quando estes pacotes são retransmitidos, novos recursos precisam ser alocados.

A implementação atual mais popular do TCP, denominada *TCP Reno*, possui um mecanismo de controle de congestionamento composto de quatro algoritmos: *Slow Start, Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery* [Allman, 1999]. Os dois primeiros são utilizados pelo TCP emissor para controlar a quantidade de dados injetada na rede, enquanto os outros dois são utilizados para a recuperação de perdas sem a necessidade de se esperar que um *timeout* aconteça. Desta forma, o TCP ajusta a sua taxa de transmissão de acordo com a estimativa de banda passante disponível. Tal adequação é governada pelo recebimento de ACKs enviados pelo receptor. Se três ou mais reconhecimentos forem recebidos para o mesmo segmento, o segmento em questão é considerado perdido e o tamanho da janela de transmissão é reduzido à metade. Quando uma perda é detectada pela ocorrência de um *timeout*, a janela de transmissão é reduzida drasticamente para um segmento e o TCP emissor entra na fase de *Slow Start*.

A principal deficiência do TCP Reno é que seu desempenho é degrado quando múltiplos pacotes são perdidos em uma mesma janela. Quando múltiplas perdas ocorrem, o tamanho da janela é reduzido à metade ou a um segmento diversas vezes e o retorno ao tamanho original da janela ocorre após um considerável intervalo de tempo. Isto acontece porque apenas o primeiro pacote perdido é recuperado *pelo Fast Retransmit* e os demais somente após a ocorrência do *timeout*. Visando suprir esta

deficiência três novas soluções foram propostas: o TCP SACK ameniza os problemas de desempenho do TCP Reno provendo informações de quais e quantos pacotes foram recebidos corretamente pelo TCP receptor [Mathis, 1996]; o TCP *NewReno* aprimora o mecanismo de recuperação de perdas do TCP Reno através da avaliação e diferenciação dos tipos de reconhecimentos recebidos [Floyd, 1999]; finalmente o TCP *Limited Transmit* foi desenvolvido para melhorar o desempenho do mecanismo de recuperação de perdas do TCP Reno para conexões com tamanho pequeno de janela [Allman, 2001].

A notificação de congestionamento é feita no TCP através do descarte de segmentos nos roteadores congestionados. O Gerenciamento Ativo de Filas (*Active Queue Management* - AQM) foi introduzido a fim de se obter o gerenciamento mais eficiente nas filas dos roteadores. A idéia central de AQM é a notificação antecipada da existência de congestionamento incipiente através da marcação/descarte de pacotes, de forma a permitir que os emissores TCP possam reduzir sua taxa de transmissão antes que as filas fiquem cheias, evitando assim, a degradação do desempenho do TCP.

RED (*Random Early Detection*), o algoritmo de AQM recomendado pelo IETF para a Internet, faz a marcação/descarte dos pacotes utilizando propriedades estatísticas sobre o tamanho médio da fila. Primeiramente ele estima o valor do tamanho médio da fila, e o compara com dois limiares, que representam os limiares mínimo (*minth*) e máximo (*maxth*). O valor da probabilidade de marcação/descarte depende de qual das três regiões, definidas pelos dois limiares, o valor do tamanho estimado da fila encontra-se. Se o valor calculado for menor que *minth*, nenhum pacote será marcado/descartado. Se o valor calculado estiver entre *minth* e *maxth*, o algoritmo está na zona de prevenção e os pacotes são marcados/descartados com uma probabilidade *p* que cresce linearmente até *maxp*. Caso o tamanho médio da fila esteja acima de *maxth*, o algoritmo inicia o controle de congestionamento e descarta todos os pacotes que chegam [Floyd, 1993].

Determinar corretamente os parâmetros de RED é um desafio. Quando os valores destes limiares não são corretamente definidos, RED pode vir a se comportar até pior que a política *tail drop*, que descarta pacotes de forma reativa e não preventiva como em RED. Devido aos problemas apresentados por RED outras políticas de AQM foram propostas para superar suas deficiências, entre elas pode-se citar: ARED Floyd, 2001], FRED [Lin, 1997] e *Blue* [Lima, 2003].

No mecanismo de AQM, a notificação de congestionamento é feita geralmente aos nós finais através do descarte de pacotes. Entretanto, esta não é a única forma de notificação de congestionamento possível de ser utilizada por AQM. O mecanismo de ECN substitui o descarte de pacotes utilizando um bit de cabeçalho do pacote para sinalizar o congestionamento, caso o protocolo de transporte entenda tal notificação. [Ramakrishnan, 2001].

Evitar a ocorrência de congestionamento e controlá-lo são de capital importância para a operação da rede. Em redes *best-effort* o congestionamento pode degradar ainda mais os baixos níveis de qualidade de serviço. Em redes com suporte a QoS (*Quality of Service*), o congestionamento pode inviabilizar o compromisso de oferecimento de tal suporte. Assim sendo, a prevenção e o controle de congestionamento são fatores essenciais para aprimorar o desempenho das aplicações e a otimização de recursos da rede. Desta forma, é importante estudar os mecanismos de controle de congestionamento, visando verificar quais deles é o mais eficaz em prevenir o congestionamento, ou ainda fazer o controle eficiente quando ele ocorre, minimizando assim, os seus danos.

Um estudo sobre todos estes mecanismos pode ser mais bem desenvolvido utilizando simuladores. A vantagem de se usar simuladores é que eles permitem obter resultados bastante satisfatórios sem causar impactos indesejados no mundo real, evitando desperdícios de recursos. Além disto, podem ser realizados testes em larga escala que podem ser controlados e reproduzidos. O uso de simuladores é uma maneira bastante apropriada de se determinar, com resultados muito próximos da realidade, quais são os mecanismos de controle de congestionamento mais adequados para determinadas situações e a viabilidade de implantação dos mesmos.

O NS (*Network Simulator*) é hoje o simulador de redes padrão utilizado no meio acadêmico. Entretanto, para utilizá-lo de forma eficiente exige-se do usuário não apenas o conhecimento do simulador, bem como da linguagem TCL para gerar os scripts de simulação. Além, disto, faz-se necessário o uso de ferramentas específicas para pós-processar a enorme quantidade de dados gerados durante as simulações.

Neste trabalho é apresentada uma ferramenta que através de simulações permite ao usuário a avaliação conjunta de implementações TCP, com e sem suporte a ECN e políticas de AQM, em diferentes cenários de rede, nos quais, podem ser feitas variações de tipo de tráfego e escolha das métricas de eficiência a serem utilizadas nas avaliações, tais como: *throughput, goodput*, taxa de perda, atraso, *jitter*, etc. A ferramenta é bastante simples, e não se exige do usuário conhecimentos aprofundados no que diz respeito ao simulador NS, nem tampouco da linguagem TCL. Além disto, ela foi desenvolvida de forma a permitir que a execução das simulações e a recuperação dos resultados gerados pudessem ser feitas via Web. Desta forma, pode-se instalar a ferramenta e o simulador em uma máquina com grande capacidade de processamento e possibilitar que a simulação e o processamento dos dados gerados possam ser feitos a partir de qualquer máquina que possua um *browser* Web.

Este artigo está organizado da seguinte forma. A seção 2 apresenta um breve resumo das diferentes tecnologias utilizadas no desenvolvimento desta ferramenta. A seção 3 descreve a estrutura e o funcionamento da ferramenta. Finalmente, na seção 4, as conclusões são delineadas.

2 Tecnologias Utilizadas

As tecnologias empregadas no desenvolvimento do simulador incluem JSP (*Java Server Pages*), que é uma tecnologia para desenvolvimento de aplicações Web, o servidor Tomcat em sua versão 4.1.10, para suporte a tecnologia JSP e o simulador de redes NS (*Network Simulator*), utilizado para execução das simulações.

Servlets são uma tecnologia Java para o desenvolvimento de aplicações do tipo cliente-servidor. São programas executados no servidor Web, funcionando como uma camada intermediária entre as requisições recebidas do Web *browser* ou outro cliente HTTP e bancos de dados ou aplicações disponíveis no servidor HTTP. *Java Server Pages* (JSP) é a tecnologia Java para desenvolvimento de aplicações Web que permite combinar código HTML estático com conteúdo gerado dinamicamente através de *servlets*. A plataforma JSP necessita de um servidor que converta automaticamente qualquer página JSP em um *servlet* equivalente, isto é, que gere código fonte Java a partir de um documento HTML. Existem diversos servidores atualmente trabalhando com JSP. Um deles é o servidor *Tomcat* da Apache. O *Tomcat*, mais especificamente a sua versão 4.1.10, é uma referência oficial na implementação de *servlet* 2.6 e JSP 1.2. A principal vantagem da utilização do *Tomcat* é o fato dele ser gratuito.

O NS trata-se se um simulador de redes dirigido por eventos discretos que simula vários tipos de redes IP. Ele pode trabalhar simulando protocolos de transporte como o TCP e o UDP; tráfegos de aplicações como FTP, Telnet, Web, CBR (Constant Bit Rate) e VBR (*Variable Bit Rate*); políticas de gerenciamento de filas e políticas de escalonamento como *Tail Drop*, RED e suas variações; etc. O NS pode ainda gerar diversos arquivos de *log* de todos os eventos ocorridos durante o processo de simulação, permitindo que os dados desejados sejam obtidos com grande facilidade. Uma das grandes vantagens do NS reside no fato de ele ser totalmente gratuito e com código fonte aberto.

3 Estrutura da Ferramenta

A ferramenta proposta, nomeada como "*TeCePe*" possui três etapas principais. A primeira etapa é a configuração dos parâmetros da simulação, que deve ser feita pelo usuário através de um formulário em HTML. Em uma segunda etapa é realizada a simulação através do NS, gerando um conjunto de arquivos com os *traces* da simulação. Na terceira etapa estes arquivos são submetidos a um pós-processamento e os resultados gerados são coletadas pelo sistema e disponibilizados via navegador *Web*.

A estrutura de dados do *TeCePe* é composta basicamente por quatro classes principais:

- A classe *BeanUsuario*, que contém os atributos e métodos para cadastro, alteração e exclusão de usuários do sistema;
- A classe *BeanSimulacao*, que implementa os atributos e métodos da simulação requisitada, tais como solicitar uma simulação e recuperar os dados gerados por uma simulação. Os parâmetros são configurados por meio de um objeto da classe *BeanParam* e os resultados são acessados através da classe *BeanResultado*;
- A classe *BeanParam*, que contém os atributos e métodos referentes aos parâmetros da simulação requisitada por um usuário devidamente cadastrado;
- A classe *BeanResultado*, que possui atributos e métodos para recuperar os resultados de uma simulação previamente requisitada e completada, de acordo com solicitação da classe *BeanSimulacao*.

Estas classes são escritas em Java e permitem a execução das principais tarefas realizadas pelo simulador, como cadastro de usuários, pedido de simulação e visualização dos resultados gerados. A interface que interage com estas classes é construída através de arquivos JSP. Estes arquivos possuem os formulários e as chamadas aos métodos das classes descritas acima. Assim, quando o usuário deseja realizar uma simulação, ele acessa a página de configuração da simulação, que por sua vez instancia um objeto da classe *BeanSimulacao*. Os parâmetros configurados através do formulário são repassados ao objeto da classe *BeanParam* pertencente a classe *BeanSimulacao* e a simulação é iniciada. Este e os demais fluxos de execução da ferramenta podem ser vistos na Figura 1.

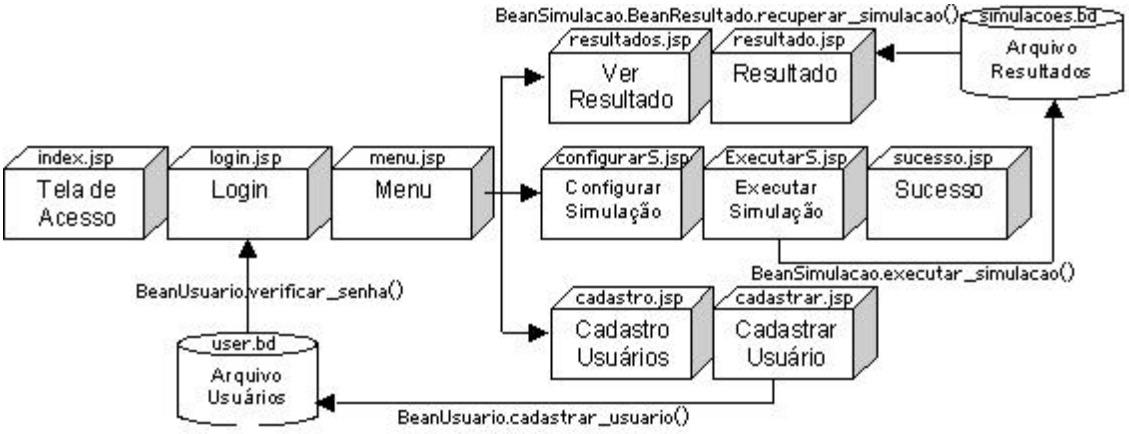

Figura 1 **Fluxo de Execução da Ferramenta.**

Para ter acesso ao simulador basta informar o endereço "*http://enderedo_do_host:8080/Simulador"* no navegador, onde o endereço do *host* pode ser endereço IP ou nome da máquina onde o *Tomcat* e a Ferramenta estão instalados.

Para utilizar a ferramenta, um usuário previamente cadastrado deve informar seu nome e senha de acesso na tela de acesso. Quando submete o pedido de acesso, o arquivo de login faz a verificação dos dados informados acessando o arquivo de cadastro (*user.bd*). Feito o acesso é disponibilizado ao usuário a tela de menu. Esta tela possui as opções disponíveis de acordo com a categoria do usuário ("*Administrador*" ou "*Restrito*"). Para usuários restritos não é disponibilizado o cadastro de usuários. Caso a opção de simulação seja escolhida, a tela de configuração da simulação é exibida contendo um formulário com os parâmetros a serem escolhidos. Quando a simulação é requisitada o arquivo *executarS.jsp* processa o pedido enviando-o a um objeto da classe *BeanSimulacao*, juntamente com os parâmetros já atribuídos ao objeto da classe *BeanParam* instanciado na própria classe *BeanSimulacao*. Se todos os parâmetros estiverem corretos uma tela de sucesso é exibida. Para ver o resultado da simulação o usuário deve acessar a tela de resultados a partir do menu, onde é exibida uma lista com as simulações requisitadas. Escolhida uma delas e desde que a mesma já tenha sido processada, serão exibidos os resultados obtidos. O tempo que uma simulação necessita para ser processada pode variar consideravelmente dependendo dos parâmetros escolhidos. Desta forma, dificilmente um resultado será fornecido imediatamente após a requisição. Caso a simulação ainda não tenha sido concluída, uma mensagem "*não processado*" aparecerá nos campos média e desvio padrão da tela de resultados.

O cálculo dos resultados é feito através de um *script Shell* que efetua um pós-processamento sobre os arquivos gerados pelo *script* TCL executado pelo NS. Destes arquivos são retirados os seguintes valores:

- *Arquivo TA*: arquivo que registra todos os eventos ocorridos durante a simulação. Dele são retirados os valores de *jitter* e *delay*, quando solicitado explicitamente;
- *Arquivo PQ*: arquivo que monitora a variação no tamanho da fila e na probabilidade de descarte. Fornece os valores da variação instantânea da probabilidade de perda e variação instantânea do tamanho da fila;

- *Arquivo MQ*: efetua o monitoramento da fila em intervalos de um segundo. Informa o tamanho da fila, taxa de utilização do enlace e taxa de perda do enlace;
- *Arquivo TT*: monitora cada fluxo gerado pela simulação. Dele é possível retirar os valores médios do *throughput*, *goodput*, RTT, *timeout* e *cwnd*.

Todos os arquivos descritos acima e os demais arquivos gerados pelo pós-processamento são guardados no diretório criado para o usuário, seguindo o padrão */TeCePe/scripts/projeto/usuario/cod_simulacao*, onde o projeto é informado no cadastro do usuário e o código da simulação é gerado automaticamente

4 Considerações Finais

A ferramenta proposta permite o estudo em conjunto dos mecanismos de gerenciamento de filas, das diversas implementações TCP e do suporte a ECN de forma bastante simples e sem exigir conhecimentos aprofundados do usuário no que diz respeito ao simulador NS, que por sua vez exige outros conhecimentos como a linguagem TCL, necessária para geração dos scripts. Além disso, como a ferramenta foi desenvolvida em JSP, é possível utilizá-la através da Internet, possibilitando o acesso remoto a qualquer funcionalidade da mesma. Os resultados gerados pelas simulações, principalmente os gráficos, oferecem uma importante ajuda no estudo dos mecanismos de rede implementados, bem como nos possíveis mecanismos implementados futuramente.

Como trabalhos futuros propõe-se a automatização da ferramenta para inserção de novos mecanismos e o aumento da flexibilidade da topologia, permitindo assim a variação no número de nós, inserção de mais enlaces gargalo, etc.

Referências

Allman, M.; Balakrishnan, H.; Floyd, S. “Enhancing TCP's Loss Recovery Using Limited Transmit”. RFC 3042, 2001.

Allman, M.; Paxson, V.; Stevens, W. “TCP Congestion Control”. RFC 2581, 1999.

Athuraliya, S. at al. “REM: Active Queue Management”. IEEE Network, Volume:15 Issue:3 2001. pp 48-53.

Fall, Kevin; Varadhan, Kannan. “The ns Manual”. Disponível em <http://www.isi.edu/nsnam/ns/doc/index.html >.

Floyd, S. at al. “Adaptive RED: An Algorithm for Increasing the Robustness of RED’s Active Queue Management”, 2001.

Floyd, S.; Henderson, T. “The NewReno Modification to TCP's Fast Recovery Algorithm”. RFC 2582, 1999.

Floyd, S.; Jacobson, V. “Random Early Detection gateways for Congestion Avoidance”. IEEE/ACM Transactions on Networking, V.1 N.4, 1993, p. 397-413.

Lima, Michele. M. A. E.; FONSECA, Nelson. L. S. “Controle de Tráfego Internet”. In: Congresso da Sociedade Brasileira de Computação, XXII, 2002, Florianópolis.

Lin, D and Morris, R. “Dynamics of Random Early Detection”. In the Proceedings of SIGCOMM'97, 1997.

Mathis, M. at al. “TCP Selective Acknowledgment Options”. RFC 2018, 1996.

Ramakrishnan, K.; Floyd, S.; Black, D. “The Addition of Explicit Congestion Notification (ECN) to IP”. RFC 3168, 2001.