Capítulo 2

Simulando a Internet: Aplicações na Pesquisa e no Ensino

Carlos Alberto Kamienski, Djamel Sadok, Dave Alberto T. Cavalcanti, Dênio Mariz T. de Sousa, Kelvin Lopes Dias

Abstract

Traditionally, there have been considerable complexity and costs associated to understanding how the Internet works and to the undertaking of performance evaluations based on real experiments, due to its high heterogeneity and planetary dimensions. Simulation presents itself as a flexible approach that has been frequently used in order to evaluate a variety of scenarios, including the behaviour of newly developed protocols and technologies under the effects of different topologies. The main goal of this course is introducing students to the topic of Internet simulation, giving especial attention to its application in research and education. The subject will be approached both in a theoretical and practical way, with relevant concepts, useful hints and comments about some known pitfalls (even if sometimes subtle) that may invalidate conclusions taken by simulation.

Resumo

Compreender o funcionamento da Internet e realizar estudos de avaliação de desempenho baseados em experimentos reais são atividades de grande complexidade e alto custo, devido a sua grande heterogeneidade e dimensões planetárias. Por esse motivo, simulação é uma técnica utilizada com muita freqüência, pela flexibilidade em testar cenários variados, incluindo o comportamento de protocolos e novas tecnologias e efeito de diferentes topologias. O objetivo principal deste minicurso é introduzir o aluno em simulação da Internet, enfatizando suas aplicações em pesquisa e ensino. O tema será abordado sob um enfoque teórico e prático, apresentando conceitos necessários, dicas de utilidade prática e comentários sobre possíveis armadilhas (às vezes sutis) que podem invalidar as conclusões sobre os resultados de simulação.

2.1. Introdução

Simulação é uma ferramenta importante e amplamente utilizada para testar o comportamento de componentes de redes (por exemplo, protocolos). O objetivo principal deste trabalho é introduzir o aluno em simulação da Internet, enfatizando suas aplicações na pesquisa e no ensino. O tema é abordado sob um enfoque teórico e prático, apresentando os conceitos necessários e dicas de grande utilidade prática.

Atualmente não existe muito material didático de conteúdo específico sobre simulação de redes e o simulador *ns*. A grande motivação para este trabalho é tentar transmitir um pouco da experiência que os autores acumularam nos últimos anos, trabalhando com simulação de redes e da Internet. Existem vários problemas, dificuldades e dúvidas simples que são muito comuns aos iniciantes em simulação da Internet. Muitas vezes, eles são devidos à falta de embasamento teórico, em probabilidade e estatística. Outras, apenas falta de algumas informações que geralmente não se encontra em lugar nenhum, exceto naquelas pessoas que já enfrentaram os mesmos desafios e encontraram as respostas na sua experiência prática. Os aspectos formais de probabilidade e estatística, comum em cursos mais tradicionais de simulação, são abordados somente na medida em que forem necessários para explicar os requisitos de simulação na Internet.

- Apresentar aspectos importantes que devem ser considerados quando forem realizadas simulações envolvendo a Internet.
- Apresentar dificuldades e erros comuns em parametrizar corretamente um ambiente de simulação para avaliar aspectos da Internet.
- Introduzir o simulador de redes *ns* e apresentar exemplos práticos de simulações que podem ser realizadas com ele.
- Despertar o interesse da comunidade acadêmica para a simulação como uma ferramenta de ensino de redes de computadores e da Internet.
- Realizar um estudo de caso de pesquisa usando simulação sobre Qualidade de Serviço (QoS) na Internet com o *ns*.

Este trabalho tem como foco alunos, professores, pesquisadores e profissionais interessados em realizar simulações envolvendo a Internet. Ele pressupõe algum conhecimento sobre redes de computadores, incluindo noções sobre protocolos, arquitetura TCP/IP e Internet. Uma disciplina introdutória da graduação de redes de computadores geralmente apresenta um conteúdo suficiente para acompanhar o texto. Conceitos de probabilidade e estatística, geralmente abordados em disciplinas básicas de graduação, também são de grande utilidade (no entanto, ter cursado ambas as disciplinas não é um requisito do minicurso). Adicionalmente, o envolvimento com o estudo e a pesquisa avançados sobre protocolos na Internet e com projetos prévios de simulação é útil para motivar e auxiliar na compreensão do tema.

As seções foram organizadas da seguinte maneira. A seção 2.2 apresenta conceitos básicos sobre estatística, probabilidade, TCP/IP e Internet. Na seção 2.3 são abordados os principais aspectos envolvidos com a simulação de sistemas em geral e na seção 2.4, aspectos específicos na Internet. A seção 2.5 apresenta o simulador de redes *ns*. Exemplos práticos de utilização do *ns* visando sua utilização no ensino são apresentados na seção 2.6, enquanto a seção 2.7 mostra a elaboração e os resultados de um estudo de caso baseado em simulação. Finalmente, algumas considerações finais são apresentadas na seção 2.8.

2.2. Conceitos básicos

Esta seção apresenta conceitos básicos necessários para auxiliar a compreensão das demais seções. A revisão de bibliografia é feita de maneira bastante superficial, apenas para apresentar os pontos mais importantes de cada assunto.

2.2.1. Internet e TCP/IP

A Internet é formada pelo agrupamento de uma grande quantidade de redes ao redor do mundo interconectadas pelo protocolo IP. Também é chamada de "rede mundial de computadores", ou "a rede das redes".

2.2.1.1. TCP/IP

O termo "TCP/IP" refere-se ao uma família de protocolos de comunicação, projetados com o objetivo de construir interconexões de redes. O conjunto de protocolos recebeu o nome TCP/IP em homenagem a dois dos seus mais importantes protocolos: o Transmission Control Protocol (TCP) e Internet Protocol (IP). A Internet é formada pelo agrupamento de uma grande quantidade de redes ao redor do mundo interconectadas pelo protocolo IP. Também é chamada de "rede mundial de computadores", ou "a rede das redes".

Uma vez que cada rede local tem sua própria interface de comunicação, dependente de tecnologia (ex: ethernet, token ring), um dos objetivos do TCP/IP é estabelecer uma camada padrão de serviços de comunicação sobre a camada de hardware, de maneira que o usuário não tome conhecimento da arquitetura de rede física. Portanto, esses serviços de comunicação oferecidos são acessados pelas aplicações de maneira uniforme e independente da tecnologia da rede física subjacente.

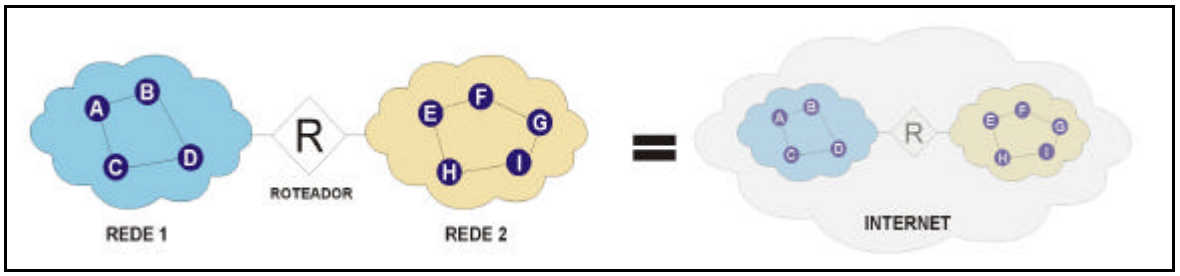


Figura 2.1 – Duas redes interconectadas formando uma inter-rede

Outro objetivo do TCP/IP é permitir a interconexão de redes físicas diferentes, para formar uma internet, que aparece para o usuário como uma grande rede, tal como a Internet que conhecemos hoje. E para interconectar duas redes precisamos de um host[1] que esteja conectado a ambas as redes e que possa enviar pacotes de uma rede para outra, ou seja, "rotear" pacotes. Esses *hosts*, chamados de roteadores, são invisíveis do ponto de vista do usuário.

[1] O termo "host" é usado para denominar qualquer computador conectado a rede.

Endereçamento IP e roteamento

Para ser possível identificar cada *host* na inter-rede (internet), cada servidor recebe um endereço IP[2], que consiste de um número de 32 bits[3] dividido logicamente em duas partes: endereço da rede e endereço do *host*. O endereço IP é representado com números separados por pontos, por exemplo, "200.125.34.27".

A parte responsável pelo roteamento de pacotes é o protocolo IP. Quando uma aplicação envia um pacote de dados para outro *host*, o IP determina para qual rede o pacote deve ser encaminhado e, se necessário, roteia o pacote de uma rede para outra através dos roteadores. O IP sabe para qual rede encaminhar o pacote baseado no endereço IP do destinatário, que contém a identificação da rede de destino e do host de destino, dentro daquela rede.

Arquitetura TCP/IP e sua família de protocolos

As funcionalidades do TCP/IP são distribuídas em 4 camadas diferentes e bem definidas. A distribuição em camadas leva ao termo "pilha de protocolos", em função dos vários protocolos responsáveis pelos serviços atribuídos a cada camada. A Figura 2.2 mostra a organização das camadas do TCP/IP e alguns protocolos da família posicionados na camada correspondente aos serviços que executa. Embora apenas alguns protocolos sejam mostrados na figura, atualmente existem mais de 100 protocolos definidos que podem interoperar no modelo TCP/IP.

O nível de enlace-hardware corresponde ao hardware da rede incluindo o *driver* (software) e a interface (placa de rede) e equivale aos níveis de enlace e físico do modelo OSI. Observe que o Ethernet não é exatamente um protocolo, mas um padrão definido pelo IEEE para redes locais. Da mesma forma, PPP, SLIP, Token Ring, ATM etc, não são parte do TCP/IP mas dão suporte a ele.

O nível inter-redes gerencia o movimento de pacotes ao redor da rede e inclui o protocolo IP, junto com outros protocolos auxiliares tais como ICMP, IGMP.

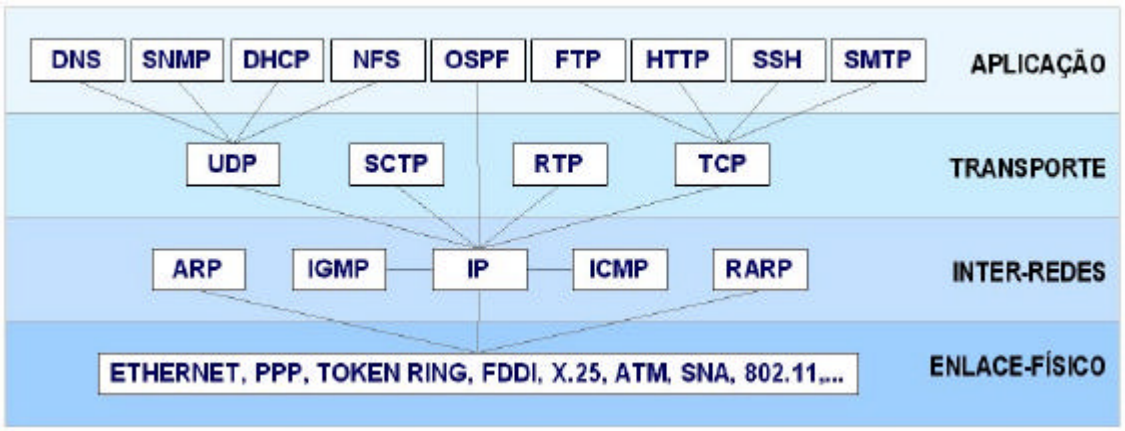


Figura 2.2 – O modelo de camadas e a pilha de protocolos do TCP/IP

[2] Quando um host tem mais de uma interface, cada interface tem um endereço IP diferente.
[3] O IPv4 usa 32 bits para o endereço e o IPv6 usa 128 bits. Neste texto assume-se o IPv4.

O nível de transporte provê mecanismos para que aplicações executando em dois *hosts* diferentes troquem dados entre si. Existem vários protocolos definidos para a execução desta tarefa e a escolha deles irá depender do tipo de serviço desejado. Os protocolos mais usados são TCP e UDP.

O nível de aplicação se refere aos protocolos de alto nível que suportam os vários serviços oferecidos tais como FTP, HTTP, DNS, SSH, SNMP, DHCP, NFS entre outros. Entretanto, um programa escrito pelo usuário pode definir sua própria estrutura de comunicação entre o cliente e o servidor e usar os protocolos do nível de transporte (e até mesmo o próprio IP diretamente) para enviar dados.

2.2.2. Probabilidade e Estatística para Simulação

Nesta seção são descritos resumidamente os principais conceitos de probabilidade e estatística necessários para a realização e avaliação adequada de simulações.

2.2.2.1. Variável Aleatória

Uma variável aleatória é um mapeamento do resultado de um evento[4] em um número. O estado de um servidor, por exemplo, pode ser modelado como uma variável aleatória que pode assumir o valor 1 quando o servidor estiver funcionando ou zero caso contrário. Neste exemplo, a variável mapeia o evento estado do servidor (desligado ou funcionando) em um número (0 ou 1). Uma variável aleatória pode ser discreta ou contínua. Uma variável é discreta quando o conjunto de possíveis valores é enumerável, como no exemplo acima. Outro exemplo de variável discreta é o número de pacotes IP descartados por um roteador em um intervalo de tempo. Já uma variável contínua pode assumir qualquer valor dentro de um determinado intervalo, ou seja, o conjunto dos possíveis valores não é enumerável. O atraso observado no estabelecimento de uma conexão TELNET entre um usuário e um servidor em uma rede pode ser modelado como uma variável contínua.

2.2.2.2. Eventos Independentes

Dois eventos são chamados independentes se a ocorrência de um evento não afeta a probabilidade de ocorrência do outro. Durante a realização de uma simulação é extremamente importante identificar a relação de dependência entre as variáveis envolvidas. O número de pacotes descartados por um roteador e o número de pacotes que chegam em um intervalo de tempo são variáveis dependentes. Já o número de chamadas telefônicas que chegam a uma central num intervalo de tempo e o tempo de duração das chamadas são variáveis independentes.

2.2.2.3. Distribuição de Probabilidade e Função densidade

Considere o exemplo da variável X que modela o estado de um servidor, e que $P[X = 1] = p1 \quad 0 \quad P[X = 1] = p2$. O conjunto $\{p1, p2\}$ é chamado de distribuição de probabilidade da variável discreta X. A distribuição de probabilidade determina completamente o comportamento da variável, uma vez todos os possíveis valores da variável têm suas probabilidades especificadas.

---

[4] Um evento é o resultado de um experimento aleatório.

No caso de uma variável contínua X, define-se uma função de distribuição acumulada $F_X(x)$ que determina a probabilidade da variável assumir um valor menor ou igual a um determinado valor x:

$$F_X(x) = P(X \le x) = \int_{-\infty}^{x} f_X(u)du,$$

onde $f_X(x)$ é a função densidade de probabilidade (ou *probability density function - PDF*) de X. Assim, como a distribuição de probabilidade define completamente uma variável discreta, a função densidade define o comportamento de uma variável contínua.

A Tabela 2.1 mostra três tipos de variáveis aleatórias bastante utilizadas para modelar eventos em simulações.

Tabela 2.1 – Exemplos de variáveis aleatórias comuns.

Variável	PDF	Alguns eventos que é capaz de modelar
Uniforme	$\begin{cases} f_X(x) = \dfrac{1}{b-a}, & a \le x \le b \\ 0, & \textit{outros casos} \end{cases}$	Direção de movimentação de um usuário em uma rede celular; Probabilidde de um determinado pacote conter um erro dentre um conjunto de pacotes enviados.
Exponencial	$\begin{cases} f_X(x) = ae^{-ax}, & a > 0, x \ge 0 \\ 0, & \textit{outros casos} \end{cases}$	Tempo entre chegadas de requisições de conexões TELNET em um servidor; Tempo entre chamadas realizadas por um usuário em uma rede celular.
Poisson	$P[X = k] = \dfrac{e^{-lt}(lt)^k}{k!}, \; l > 0$	Número de chamadas telefônicas, de consultas a um banco de dados ou de conexões TELNET em um servidor dentro de um intervalo de tempo.

2.2.2.4. Média ou Valor Esperado

A média ou valor esperado de uma variável aleatória X é dado por:

$$E(X) = \sum_{i=1}^{n} p_i x_i = \int_{-\infty}^{+\infty} x f_X(x)dx.$$

O somatório é utilizado para o caso de uma variável discreta. A média é um parâmetro muito utilizado na prática para obter estimativas do comportamento de um sistema. Por exemplo, em uma rede de computadores parâmetros comumente utilizados para avaliação de desempenho são vazão média e atraso médio. Aspectos de como obter esse parâmetro e qual a significância do seu valor são abordados mais adiante na seção sobre estimação de parâmetros.

2.2.2.5. Variância e Desvio Padrão

Em determinadas situações a descrição do comportamento do sistema utilizando apenas a média não é suficiente. Na realidade, a média $E[X]$, não contém nenhuma informação sobre o espalhamento dos dados, ou seja, quão distante as amostras estão do valor médio. A média dos dois conjuntos A={5,10,15} e B={0,10,20} é a mesma, mas o espalhamento é diferente.

No caso de aplicações multimídia na Internet, por exemplo, medir o atraso médio em uma conexão não é suficiente para avaliar a qualidade do serviço, pois um valor médio razoável pode ser resultado da combinação de atrasos bastante elevados e atrasos pequenos. Essa grande variação do atraso pode ser mais prejudicial para aplicação do que um atraso médio elevado. Nesse caso, um parâmetro para medir a dispersão dos dados em relação ao valor médio é extremamente importante.

A variância e o desvio padrão são parâmetros que medem a dispersão dos dados em relação à média. A quantidade $\left(X - E(X)^2\right)$ representa o quadrado da distância entre X e sua média e o valor médio dessa quantidade é chamado variância de X:

$$Var(X) = E[(X - E(X))^2] = \sum_{i=1}^{n} p_i (x_i - E(X))^2 = \int_{-\infty}^{+\infty} (x_i - E(X))^2 f_X(x) dx$$

.

O quadrado da distância é utilizado para podermos computar a dispersão tanto de valores acima, como abaixo da média. A variância é tradicionalmente denotada por s^2. A raiz quadrada da variância é chamada de desvio padrão e é representado por s.

2.2.2.6. Amostragem de dados e estimação de parâmetros

As características do conjunto de dados obtidos através das ocorrências de uma variável aleatória são representadas por parâmetros como valor esperado, variância e desvio padrão, por exemplo. Na maioria dos problemas reais, estes parâmetros são desconhecidos e para obtê-los, precisaríamos repetir o experimento um número infinitamente grande de vezes, o que é impossível. Na prática, estes parâmetros são aproximados por valores obtidos através da amostragem de alguns valores entre o conjunto das ocorrências da variável. Estas estimativas são chamadas de estatísticas. A média aritmética (média amostral) de um conjunto $\{x_1, x_2, \ldots, x_n\}$ de ocorrências de uma variável X, representada por $\bar{x}$, é uma estimativa (estatística) para o valor esperado $E(X)$. É importante a distinção entre um parâmetro e uma estatística, porque o parâmetro é um valor fixo, enquanto uma estatística é uma variável aleatória.

Considere o exemplo de uma variável aleatória T que representa o tamanho dos pacotes IP que trafegaram em um segmento de rede em um determinado intervalo de tempo. Pela especificação do protocolo IP, sabe-se que um pacote tem um tamanho mínimo de 48 bytes e um máximo de 65535 bytes, logo T pode assumir qualquer valor inteiro no intervalo {48, ..., 65535}. No entanto, não sabemos a distribuição de T, nem muito menos o seu valor médio $E[T]$. Mas, coletando alguns pacotes e medindo seus tamanhos formamos uma amostra, por exemplo: {100, 124, 64, 75, 340, 240, 294}. Através dessa amostra podemos calcular a média amostral que serve de estimativa para $E[T]$.

Do exemplo acima, podemos concluir que se obtivermos k amostras de tamanho n da variável aleatória T, teremos k médias amostrais diferentes que são estimativas para a média de T. O próximo passo é obter uma única estimativa para a média a partir das k estimativas.

2.2.2.7. Intervalo de confiança

Na realidade não é possível encontrar uma estimativa perfeita para a média a partir de um número finito de amostras de tamanho finito. Neste caso, a melhor opção é obter limites probabilísticos, por exemplo, c_1 e c_2, de forma que seja alta a probabilidade $(1-a)$ de a média exata pertencer ao intervalo (c_1, c_2):

$$P(c_1 \leq E(X) \leq c_2) = 1 - a$$

O intervalo (c_1, c_2) é chamado intervalo de confiança para a média, α é chamado nível de significância e $1-a$ é chamado coeficiente de confidência. Tipicamente, o intervalo de confiança é expresso em termo de um percentual próximo de 100%, por exemplo, 90% ou 95%; enquanto o nível de significância a α é expresso como uma fração e é geralmente próximo de zero, por exemplo, 0,05 ou 0,1.

2.3. Simulação de Sistemas

Simulação é uma técnica muito utilizada para avaliação de desempenho de sistemas em geral. Quando o sistema a ser avaliado não está disponível, o que é o caso em muitas situações, uma simulação é o caminho mais fácil para prever o desempenho ou comparar alternativas. No caso de redes de computadores, como a Internet, por exemplo, apesar do sistema já estar implementado e funcionando, uma alteração em um crítico pode gerar resultados indesejados, além de muitos transtornos. Logo, a realização de simulações é o melhor caminho para obter boas estimativas do comportamento do sistema após uma possível modificação de algum protocolo, por exemplo.

2.3.1. O papel da simulação

O comportamento da Internet tem sido explorado, avaliado e compreendido através de quatro técnicas, que são: ***medição***, ***experimentação***, ***análise*** (modelagem analítica) e ***simulação*** [30]. Três dessas técnicas (medição, análise e simulação) são tradicionalmente utilizados em avaliações de desempenho em várias áreas da computação [19]. Todas são necessárias, embora cada uma delas tenha os seus benefícios e problemas. Portanto, elas devem ser utilizadas cuidadosamente, para não comprometer as conclusões geradas a partir dos resultados coletados.

Medição é uma técnica fundamental para compreender o comportamento atual da Internet, com relação a uma série de diferentes pontos de vista. Medições podem ser usadas, por exemplo, para tentar inferir o tamanho da Internet (redes e usuários), assim como padrões de tráfego, protocolos mais utilizados e volume de tráfego. É possível também obter informações sobre o tamanho dos pacotes, quantidade de roteadores entre pontos distintos na Internet, além de parâmetros de desempenho, como atraso fim a fim, perda de pacotes e vazão. Nos últimos anos tem havido muito interesse por medições da Internet "global" ou mesmo parte dela. Existem hoje vários sites que publicam estatísticas da Internet, como Internet Weather Report, Internet Traffic Report [18] e Internet Health Report [10]. Existem ferramentas de medição (simples e complexas, comerciais e de domínio público) disponíveis, como RIPE TTM [31] e Surveyor [1]. A organização CAIDA [6] mantém informações atualizadas sobre várias ferramentas de medição.

A realização de experimentos também é uma técnica importante para conhecer e avaliar questões de implementação de aplicações e protocolos. A experimentação pode ser realizada com novos protocolos/aplicações na Internet atual, ou então em ambientes de teste (*testbed*), onde se pode ter um maior controle sobre os elementos a serem estudados. No entanto, medição e experimentação são técnicas limitadas somente a explorar a Internet atual. Elas podem ser usadas para construir ambientes que tentam imitar algum possível cenário futuro, mas certamente de maneira muito limitada. Além disso, nem sempre é possível ter acesso a redes e equipamentos, e essas experiências práticas geralmente têm um custo alto.

A técnica de análise possibilita explorar um modelo da Internet sobre o qual se tem controle completo. Através de modelos matemáticos imensamente simplificados, podem-se obter resultados rapidamente. Por isso, é também uma técnica barata, pois em princípio não necessita de equipamentos especiais. O risco de confiar em resultados analíticos é que os pressupostos e simplificações são tantos que o comportamento original da Internet pode ser perdido. Obviamente, trabalhar com análise requer um forte embasamento matemático.

Enquanto medição e experimentação permitem descobrir o "mundo real", simulação e análise estão restritas a explorar um modelo abstrato do mundo. Simulação se diferencia de medição e experimentação por permitir maior flexibilidade ao estudo e de análise por permitir a construção de modelos mais complexos e representativos do mundo real. Um importante papel da simulação é desenvolver a intuição das pessoas sobre o comportamento de aplicações, protocolos e tecnologias de rede. Isso é muito útil para pesquisadores, professores, alunos e até para profissionais que trabalham com o projeto de novas redes (podem convencer seus clientes, mostrando a eles como a nova rede vai se comportar).

Como simulação é uma técnica que se baseia na construção e execução de programas, ela geralmente é a preferida pelo pessoal da área de computação. Vários projetos de pesquisa são baseados inteiramente em resultados de simulação, o que pode trazer alguns riscos. Costuma-se fazer extrapolações a partir de resultados de simulação que freqüentemente se mostram incorretas. Por exemplo, dadas as limitações de topologias e cenários avaliados em simulações, não é correto utilizar simulação para concluir que determinada técnica *A* tem um desempenho X% melhor que a técnica *B* quando ambas são utilizadas na Internet. O motivo é que pequenas modificações na topologia, nos cenários, nos parâmetros ou mesmo no nível de detalhe que o modelo está implementado no simulador podem produzir resultados surpreendentemente contrários a essa afirmação. Um erro comum e muito freqüente é "escolher" as situações em que proposta que está sendo defendida apresenta um desempenho melhor que as outras propostas e omitir as outras situações. Por isso, é importante fazer um bom planejamento de simulação (seção 2.3.5) e realizar análises estatísticas sólidas (seção 2.2.2).

2.3.2. Erros comuns em simulação

Apesar de ser uma técnica bastante poderosa para a avaliação de sistemas, um modelo de simulação freqüentemente produz resultados não confiáveis ou errados. A seguir apresentamos os erros mais comuns cometidos em simulações. Algumas técnicas para evitar esses erros são descritas nas próximas seções.

- **Nível de detalhes não apropriado:** A simulação permite o estudo de um sistema de forma mais detalhada que a modelagem analítica, que requer geralmente várias

considerações para tornar o problema tratável matematicamente. Em uma simulação o nível de detalhes é limitado pelo tempo de desenvolvimento disponível. Além disso, quanto mais detalhes, maior a probabilidade de ocorrerem erros e mais difícil é encontrá-los. Outro aspecto a ser considerado é o tempo de execução necessário e o hardware disponível. Quanto maior o detalhamento, maior a quantidade de parâmetros de entrada e muitas vezes os valores desses parâmetros não são conhecidos. Por isso, um maior nível de detalhes, não produz necessariamente uma simulação melhor.

- **Linguagem de programação inadequada:** a escolha da linguagem de programação tem um impacto considerável no tempo de desenvolvimento do modelo. Linguagens de propósito geral são portáveis e proporcionam maior controle sobre a eficiência e o tempo de execução da simulação. Por outro lado, linguagens específicas para simulação requerem menor tempo de desenvolvimento e incluem ferramentas para tarefas comuns como análise estatística.
- **Modelos incorretos:** Os modelos de simulações são geralmente extensos programas, nos quais erros de programação ou *bugs* são bastante comuns, o que pode resultar em resultados comprometidos. Mesmo que o programa não contenha erro, este pode não representar o sistema de forma adequada, devido a considerações incorretas sobre o comportamento do sistema.
- **Inexistência ou má definição de objetivos:** Uma simulação é projeto complexo e exige uma especificação clara dos objetivos. Os objetivos devem ser escritos e estarem claros para todos os envolvidos no processo de desenvolvimento.
- **Tratamento incorreto das condições iniciais, finais e tempo de simulação:** Na maioria dos sistemas reais existe um estado transitório antes do sistema atingir o equilíbrio. Numa simulação também existe um estado inicial que não representa o comportamento do sistema em estado estacionário. Esta parte inicial da simulação não deve ser considerada. Em muitos casos, ao tentar reduzir o tempo da simulação os analistas comprometem os resultados, que são extremamente depende das condições iniciais do utilizadas na simulação.
- **Geradores de números aleatórios:** Modelos de simulação utilizam variáveis aleatórias para representar o comportamento de um determinado parâmetro do sistema, como por exemplo, o tamanho dos pacotes gerados por uma determinada aplicação. Esses valores aleatórios são gerados a partir de procedimentos chamados geradores de números aleatórios. É mais seguro utilizar um gerador bastante conhecido e analisado do que implementar o próprio gerador. No entanto, mesmo geradores conhecidos podem apresentar problemas.
- **Escolha inadequada de sementes:** Os geradores de números aleatórios geram cada valor a partir de um valor anterior. Desta forma, para gerar uma seqüência de números aleatórios o programa deve receber como entrada um valor inicial, chamado de semente. As sementes para diferentes seqüências aleatórias devem ser escolhidas cuidadosamente, de modo que seja mantida a independência entre as seqüências. Utilizar a mesma semente ou compartilhar uma seqüência de números aleatórios entre diferentes processos são erros bastante comuns que levam a conclusões que podem representar de forma incorreta o comportamento do sistema.

- **Escolha de métricas incorretas:** As métricas se referem aos critérios utilizados para quantificar o desempenho do sistema. Por exemplo, vazão e atraso são métricas bastante comuns em redes. A escolha da melhor métrica depende do serviço provido pelo sistema que está sendo avaliado. Em muitos casos as métricas mais fáceis de serem computadas são selecionadas no lugar das métricas mais relevantes. Métricas difíceis de serem obtidas são muitas vezes ignoradas. Através da manipulação das métricas também é possível modificar as conclusões do estudo.
- **Carga de trabalho não representativa:** A carga de trabalho utilizada para comparar dois sistemas deve ser representativa da real utilização do sistema. Por exemplo, se os pacotes em uma rede real são gerados como uma mistura de dois tamanhos, pequeno e longo, a carga de trabalho para comparar duas redes deve consistir de pacotes de ambos os tamanhos. Uma carga de trabalho inadequada poderá gerar conclusões erradas.
- **Topologias inadequadas e insuficientemente reais:** Uma simulação sempre contém abstrações e simplificações do sistema estudado para viabilizar a própria implementação, uma vez que é impossível criar um modelo idêntico ao real. No entanto, uma precaução especial deve ser tomada no caso de simulação de redes, para que a topologia utilizada seja adequada para o problema a ser estudado. Muitas vezes a topologia escolhida não é adequada, sendo muito simplificada ou muito complexa, o que invalida os resultados obtidos nas simulações (seção 2.4.2.3).

2.3.3. Tipos de simulação

Antes de iniciar uma simulação, deve-se decidir qual o tipo mais adequado para o problema. Dentre os vários tipos de simulação que podem ser encontrados na literatura, os mais importantes para aplicações em redes de computadores são: simulação de Monte Carlo, simulação baseada em traces, e simulação baseada em eventos discretos.

2.3.3.1. Simulação de Monte Carlo

A simulação de Monte Carlo é uma poderosa técnica para obtenção de soluções aproximadas para problemas complexos que envolvem componentes aleatórios. Este é um tipo de simulação que serve para modelar fenômenos probabilísticos invariantes no tempo. O comportamento de um sistema pode ser caracterizado por distribuições de probabilidades e seus parâmetros. No entanto, na maioria das vezes os parâmetros são desconhecidos e simulações de Monte Carlo podem ser aplicadas na obtenção de estimativas através da realização de várias replicações de um experimento. Por exemplo, suponha que o tamanho dos pacotes gerados por uma determinada aplicação segue uma distribuição qualquer $f_X(x)$. Para determinar o tamanho médio dos pacotes pode-se utilizar uma simulação de Monte Carlo para gerar várias ocorrências da variável X e depois calcular a média amostral que serve de estimativa para a média E(X).

2.3.3.2. Simulação baseada em Traces

Uma simulação baseada em *traces* é a que tem como entrada um registro que contém eventos, ordenados no tempo, observados em um sistema real. Esses registros são chamados de *traces*. Este tipo de simulação é geralmente utilizado na análise de algoritmos de alocação de recursos. Neste caso, um trace, contendo a demanda por um determinado recurso, é

utilizado como entrada da simulação, a qual pode incluir diferentes algoritmos para serem avaliados sob as mesmas condições de demanda. Uma característica importante é a credibilidade. Um trace contendo os acessos feitos a um determinado serviço na Internet, tem uma maior credibilidade do que informações geradas randomicamente através de alguma distribuição. Um dos principais problemas dos traces é o tamanho. Os traces são geralmente seqüências longas e exigem um considerável tempo computacional para serem processados. Outro problema é a dificuldade de variação da carga de trabalho aplicada.

2.3.3.3. Simulação discreta baseada em eventos

Uma simulação discreta baseada em eventos utiliza um modelo de estados discretos para o sistema. Diferente de um modelo contínuo, no qual os estados que o sistema pode assumir variam continuamente, em um modelo discreto o sistema só pode assumir um número discreto de valores. É importante notar que, mesmo em uma simulação discreta, o tempo da simulação pode assumir valores discretos ou contínuos. Qualquer simulação discreta, independente de aplicação, deve conter os seguintes componentes: um escalonador de eventos; um mecanismo de controle do tempo (*clock*); variáveis globais que descrevem os estados do sistema; rotinas para simular os eventos; rotinas para entrada de parâmetros; rotinas para coletar resultados; rotinas de inicialização; rotinas para gerenciamento dinâmico de memória e um programa principal.

2.3.4. Plataformas/linguagens de simulação

A escolha de uma linguagem adequada é um dos passos mais importantes no desenvolvimento de uma simulação. Uma escolha inadequada pode gerar problemas relacionados com o tempo de desenvolvimento e até falhas no modelo de simulação. Algumas diferenças entre linguagem são: interface amigável; curva de aprendizagem; tempo de desenvolvimento; desempenho (tempo de simulação); escalabilidade para um grande número de eventos (simulação distribuída e paralela). De um modo geral, duas alternativas podem ser consideradas: linguagem de propósito geral e linguagem de simulação. A seguir são apresentadas algumas vantagens e desvantagens em relação às esses dois tipos de linguagens.

Um dos primeiros motivos para a escolha de uma linguagem de propósito geral como Pascal ou C é a familiaridade do analista com essas linguagens mais gerais. Neste caso, não será necessário um tempo de aprendizado antes de iniciar o desenvolvimento, o que é necessário para linguagens específicas de simulação. Por outro lado, escolhendo uma linguagem geral, o usuário terá que desenvolver ou adaptar rotinas para tratamento de eventos, geração de números aleatórios etc, que já estão disponíveis em linguagens específicas para simulação como SIMSCRIPT, por exemplo. Desta forma, escolhendo uma linguagem de propósito geral, o usuário deve ficar atento para não desperdiçar tempo tentando resolver um problema ou implementar uma rotina já conhecida ou que pode ser reutilizada.

No entanto, alguns fatores mostram que nem sempre uma linguagem de simulação é a melhor escolha. Um modelo desenvolvido em uma linguagem de propósito geral é normalmente mais eficiente requer um menor tempo de CPU. Outro fator é a portabilidade das linguagens de propósito geral. Um modelo desenvolvido nesse tipo de linguagem pode facilmente ser convertido para execução em outros sistemas de computação.

Para fazer uma escolha objetiva e coerente entre uma linguagem de simulação e de propósito geral, é sugerido em [20] que o analista aprenda pelo menos uma linguagem de simulação para que outros fatores, além apenas da familiaridade com a linguagem, sejam considerados na escolha. Algumas das linguagens de simulação mais comuns são: SIMULA, SIMSCRIPT, GASP e GPSS. No caso específico de simulação da Internet, destaca-se o simulador *ns* que pode ser considerado um pacote específico para simulação de redes. O *ns*, que será descrito com maiores detalhes na seção 2.5.

2.3.5. Planejamento da Simulação

Todo projeto de simulação possui características próprias. No entanto, existem alguns passos que devem ser seguidos no planejamento de qualquer projeto a fim de evitar os erros mais comuns citados na Seção 2.3.1.

2.3.5.1. Definição clara e precisa dos objetivos

O primeiro passo em qualquer estudo de avaliação de desempenho, de um modo geral, é definir claramente os objetivos e o sistema que será modelado. O conhecimento profundo do sistema em estudo é imprescindível por parte do coordenador da simulação. Muitos sistemas são bastante complexos e, portanto, deve ser definido um escopo para o modelo a ser simulado, o que evita a inclusão exagerada de detalhes no modelo, o que aumentaria a complexidade do sistema e atrasaria o desenvolvimento do projeto.

Cada sistema possui geralmente uma lista de serviços disponíveis. Numa rede de computadores cada protocolo pode incluir um disponibilizar específico de serviços como correção de erros ou sincronização, por exemplo. Todos os serviços disponíveis por um determinado sistema devem ser identificados no início do projeto. Dessa forma, os serviços mais importantes podem ser incluídos na simulação enquanto outros, menos importantes, podem ser desconsiderados.

2.3.5.2. Escolha das métricas adequadas

As métricas são os critérios a serem utilizados para avaliar o desempenho do sistema. Para selecionar as métricas adequadas é preciso definir todos os serviços disponíveis (vide seção 2.3.5.1). Para cada requisição de serviço existem vários resultados possíveis, os quais podem ser geralmente classificados em três categorias: o serviço pode ser executado corretamente, incorretamente ou pode não ser executado pelo sistema. Por exemplo, um *gateway* pode oferecer o serviço de encaminhamento de pacotes para alguns destinos específicos na Internet. Quando um pacote chega ao *gateway*, este pode encaminhar o pacote corretamente para o destino, encaminhar para o destino errado ou descartar o pacote.

Quando o serviço é executado corretamente, o tempo de execução, a taxa e os recursos consumidos podem ser métricas para o desempenho do sistema. No caso do *gateway*, essas três métricas são respectivamente tempo de resposta, vazão e percentual do tempo que todos os recursos do *gateway* estão sendo utilizados. No caso do serviço ser executado incorretamente, identifica-se que ocorreu um erro no sistema. Em uma rede de computadores uma métrica importante para medir erros é a probabilidade de um pacote ser descartado devido a algum erro. Para a terceira possibilidade, o pacote não ser enviado, uma

métrica importante a ser medida é a disponibilidade do *gateway*, ou seja, o percentual do tempo que o serviço fica indisponível.

De um modo geral, as métricas relacionadas com as três possíveis respostas do sistema a uma requisição de serviço são chamadas métricas de velocidade, de confiabilidade e de disponibilidade.

Para a maioria das métricas os mais importantes são valores médios. Entretanto, em algumas situações a observação da variabilidade precisa ser considerada. Por exemplo, existem casos em que a alta variabilidade do tempo de resposta (ou seja, menor previsibilidade) também degrada a qualidade de serviço, mesmo que tempo médio de resposta seja baixo. Nestes casos, ambas as métricas devem ser consideradas.

2.3.5.3. Escolha correta de parâmetros, fatores e níveis

Uma listagem de todos os parâmetros que influenciam o desempenho do sistema deve ser feita. Durante o desenvolvimento da simulação, outros parâmetros podem surgir os quais devem ser incorporados à listagem inicial.

A lista de parâmetros pode ser dividida em duas partes: parâmetros serão variados durante a simulação e parâmetros que serão fixos. Os parâmetros variados são chamados de *fatores* e seus possíveis valores são chamados de *níveis*. Na maioria dos modelos, a quantidade de fatores e níveis de variação é maior do que os recursos disponíveis para considerá-los. A melhor opção é iniciar com uma lista reduzida e adicionar outros fatores gradualmente de acordo com a disponibilidade de recursos.

Os fatores devem ser escolhidos entre os parâmetros que exercem um maior impacto sobre o desempenho do sistema.

2.3.5.4. Quantidade de replicações

Uma vez que os fatores e seus níveis estão especificados, é preciso definir a seqüência de experimentos a ser realizada, que gera a quantidade de informação necessária com o menor esforço possível. Ou seja, é necessário identificar quantas replicações da simulação são necessárias para a obtenção de resultados confiáveis. A resposta a esta questão vai depender da precisão desejada e pode ser respondida utilizando conceitos de amostragem de dados e intervalo de confiança (vide seção 2.2). O nível de confiança das conclusões baseadas em um conjunto de dados depende do tamanho deste conjunto de dados. Quanto maior o conjunto, maior a precisão. Dessa forma, o número de replicações exigidas pode ser calculado utilizando-se a relação entre intervalo de confiança e tamanho da amostra.

2.3.6. Análise dos resultados

O valor dos resultados obtidos depende da implementação correta da simulação e da representatividade desses resultados em relação ao sistema real. Estes dois aspectos podem ser garantidos através da validação e verificação do modelo, do estudo das condições iniciais e tempo de simulação utilizado.

2.3.6.1. Validação e verificação do modelo

A qualidade de um modelo de simulação pode ser medida através da proximidade dos resultados obtidos em relação aos resultados do sistema real. A qualidade do modelo pode

ser medida identificando se as considerações utilizadas são aceitáveis e em seguida, verificando se o modelo implementa essas considerações corretamente. Essas duas etapas são chamadas de **validação** e **verificação**. A validação trata da representatividade das simplificações ou considerações feitas, ou seja, se o comportamento do sistema não será comprometido com as simplificações utilizadas. Enquanto isso, a verificação está relacionada com a implementação correta do modelo.

As técnicas para verificação de modelos de simulação são as mesmas utilizadas em programação de um modo geral, como: desenvolvimento modular e *top-down*; inclusão de pontos de verificação e saídas para identificar erros no programa e documentação e explicação detalhada do código; testes com exemplos simples para facilitar a análise dos resultados; traces para informar o andamento da simulação ao longo do tempo; teste de continuidade, ou seja, executar a simulação várias vezes com parâmetros de entrada bem diferentes; testes de consistência para verificar se o modelo produz resultados similares para parâmetros de entrada que têm efeitos similares; e utilização de sementes nos geradores de números aleatórios que não influenciem no resultado final das simulações.

As técnicas de validação por sua vez dependem das considerações feitas e no sistema em estudo. Diferente das técnicas de verificação, as técnicas de validação não são aplicáveis a qualquer sistema. A validação de um modelo de simulação consiste em validar três importantes aspectos do modelo: 1) considerações; 2) valores dos parâmetros de entrada e distribuições; 3) valores de saída e conclusões.

Os três aspectos acima podem ser validados comparando-os com os resultados obtidos a partir das seguintes fontes: intuição de um especialista, medição e modelagem analítica. Um especialista pode identificar erros facilmente apenas observando os resultados de uma simulação. Uma possibilidade é apresentar a um especialista os resultados da simulação e resultados de medições obtidas do sistema real e verificar se este pode distinguir entre os resultados. Uma comparação com o sistema real é a forma mais confiável de validar um modelo. Infelizmente, a realização de medições no sistema pode não ser possível por não estar ainda disponível ou pelo alto custo de tais medições.

Algumas vezes é possível modelar o sistema analiticamente a partir de simplificações do modelo. Neste caso é pode ser possível determinar analiticamente as distribuições das entradas do sistema. A comparação dos resultados teóricos com os obtidos por simulação também pode validar o modelo, mas deve ser utilizada com cuidado, uma vez que ambos podem apresentar erros no sentido de que nenhum dos dois representa corretamente o sistema. Apesar disso, a modelagem analítica é uma técnica muito utilizada, pois possibilita a validação de casos bastante complexos e não depende da intuição de um especialista.

Os resultados de simulação somente são confiáveis se puderem ser reproduzidos novamente. Por este motivo, simulação usa geradores de números pseudo-aleatórios. Caso fosse utilizado um gerador completamente aleatório, a simulação não poderia ser reproduzida. É importante ressaltar que não existe um modelo completamente validado. Pode-se mostrar apenas que o modelo não é inválido para algumas situações.

2.3.6.2. Estudo das condições iniciais e finais da simulação

Na maioria das simulações o que interessa é a análise do sistema no estado estacionário, isto é, o estado em que o sistema se torna estável. Nesses casos os resultados obtidos na parte inicial da simulação até o sistema atingir a estabilidade devem ser removidos. Esta parte inicial é chamada de estado transitório (ou transiente) e o principal problema é identificar o final desse estágio. A identificação do ponto exato do fim do estado transitório é praticamente impossível.

Um método utilizado para minimizar os efeitos do estado transitório é executar a simulação por um tempo relativamente longo, para garantir que as condições iniciais não vão afetar os resultados. No entanto este método desperdiça recursos e outro fator é que o fim da simulação também deve ser controlado. Este método não é aconselhado. Outro método usado para reduzir o período transiente é iniciar a simulação com valores de parâmetros próximos aos valores no estado estacionário.

Alguns métodos são baseados no fato de que a variabilidade dos parâmetros é menor no estado estacionário que no estado transiente. Dada uma amostra com n observações, o truncamento consiste em ignorar a l primeiras observações e então calcular o mínimo e o máximo do restante das observações. Isto é repetido para $l = 1,2, \ldots n-1$ até que a $(l+1)$-*ésima* observação não seja nem a menor nem a maior das observações restantes. O valor de l neste ponto é igual ao comprimento do estado transitório.

Nos casos em que o estado estacionário não interessa, podem-se utilizar as mesmas técnicas aplicadas para eliminar o estado transiente. O problema de determinar o tempo necessário de simulação está relacionado com o intervalo de confiança e o número de replicações como descrito anteriormente. O principal objetivo é obter o máximo de informação com o mínimo esforço. Técnicas que possibilitam a finalização da simulação assim que um determinado nível de confiança é atingido são baseadas na estimação da variância de parâmetros do sistema e são descritas detalhadamente em [20].

2.3.7. Geradores de números aleatórios

O processo de geração de números aleatórios é muito importante em uma simulação. Essa rotina é responsável por gerar valores para variáveis de acordo com suas distribuições, por exemplo, exponencial ou normal. Isto é feito em duas etapas: primeiro a geração de números aleatórios uniformemente distribuídos entre 0 e 1, em seguida, uma transformação é aplicada em cada número obtido para produzir ocorrências da distribuição desejada.

O método mais comum de gerar números aleatórios é usar a recursividade da seguinte forma: $x_n = f(x_{n-1}, x_{n-2}, \ldots)$. Conhecendo a função f, podemos gerar uma seqüência de números a partir de um valor inicial x_0, que é chamado de semente. Mas é importante observar que como a função f é determinística, a seqüência de números será a mesma para uma determinada semente. Esse gerador não é completamente aleatório e esses números são chamados *pseudo-aleatórios*.

Em alguns casos é preciso implementar estas rotinas, em outras situações o usuário pode utilizar geradores disponíveis na plataforma de simulação ou já implementados por outras

pessoas. Em ambos os casos é importante conhecer as propriedades de um bom gerador de números aleatórios, assim como formas de testar a qualidade de um gerador.

As principais características de um bom gerador de números aleatórios são: eficiência computacional, período longo e os valores sucessivos independentes e uniformemente distribuídos. Como algumas simulações requerem milhares ou milhões de números aleatórios em cada execução, o tempo para gerar esses valores deve ser pequeno. Cada gerador possui um período que determina após quantos números a seqüência se repete. Um período pequeno resulta em vários ciclos dentro da simulação, ou seja, numa seqüência de eventos repetida, com perda na característica aleatória.

Alguns tipos de geradores mais comuns são: geradores congruenciais lineares (*Linear Congruential Generator – LCG*), geradores de registro de deslocamento e geradores Fibonnacci. No entanto, testes realizados com estes geradores [25] mostram que para algumas aplicações como: a) sofisticadas simulações de Monte Carlo; b) estimação de funções de distribuições; e c) geração e teste de números primos para utilização em algoritmos de criptografia; estes geradores apresentam deficiências significativas. Nessas situações, a combinação de mais de um gerador resulta em um gerador com melhores características, como os geradores: COMBO, NCOMBO e Super-Duper [25].

Como destacado anteriormente, a escolha da semente não deve afetar os resultados da simulação. Se o gerador tem um período longo e a simulação envolve apenas uma variável aleatória, então o valor da semente não tem muita importância. Já para simulações envolvendo mais de uma variável, algumas precauções podem ser tomadas como: não utilizar o zero como semente; não utilizar números pares (especialmente para os LCG); não utilizar a mesma seqüência para variáveis diferentes; não utilizar seqüências de números com partes em comum; reutilizar sementes em replicações sucessivas sem reinicializar a semente.

A fim de testar a eficiência de um gerador, ou a transformação aplicada para gerar uma variável aleatória o primeiro passo é observar um histograma e verificar a distribuição de freqüência dos dados obtidos. Um segundo passo é aplicar tantos testes quanto possíveis entre os disponíveis para verificar a qualidade do gerador. Entre os testes mais conhecidos destacam-se os testes Qui-quadrado e Kolmogorov-Smirnov [20]. De um modo geral, estes testes determinam se uma seqüência de dados satisfaz uma determinada distribuição. É importante observar que passar em um teste é uma condição necessária e não suficiente para garantir que o gerador é bom.

2.4. Simulando a Internet

Nesta seção o tema ***simulação*** é abordado sob o ponto de vista da Internet, enfocando os principais problemas e características próprias que fazem dessa área um grande desafio para os pesquisadores.

2.4.1. Dificuldades para simular a Internet

Simular a Internet representa um grande desafio devido às suas características únicas, que tornam difícil obter uma caracterização precisa a seu respeito. Principalmente, se for considerada a Internet "global" e não apenas pequenos trechos dela como a maioria dos trabalhos aborda. Por exemplo, algumas conclusões sobre comportamento de tráfego na

Internet obtidas há alguns anos *não* são mais uma realidade atualmente, em função das mudanças de perfil do usuário, novos protocolos, novas aplicações etc, o que demonstra claramente que a Internet é um "alvo móvel" [14].

2.4.1.1. Dimensões

A Internet é muito grande, sob qualquer parâmetro que se queira observar. Seu tamanho pode ser medido em número de usuários, redes, computadores, interconexões, tráfego, mensagens de correio eletrônico etc. A Figura 2.3 mostra o crescimento do número de computadores na Internet no período de março de 2001 a março de 2002, de acordo com estimativas do NetSizer [33]. Nesse período, o número de computadores teve um crescimento de cerca de 58%, pulando de ±108 milhões ±171 milhões. No momento em que esse documento está sendo escrito, o NetSizer apresenta o número de computadores em ±188 milhões e o número de usuários em ±761 milhões.

O tamanho da Internet traz duas dificuldades. A primeira é que se apenas uma pequena parcela dos computadores apresentar um comportamento atípico, difícil de ser capturado em simulações, mesmo assim a Internet pode ter milhares desses computadores. A segunda dificuldade se refere ao problema de escala. Alguns protocolos ou mecanismos de rede que funcionam bem em redes pequenas, médias e até grandes (milhares de computadores), podem se tornar impraticáveis quando a rede for várias ordens de magnitude maior que isso (a Internet atual). Isso significa que mesmo estudos de simulação bem planejados e com topologias grandes, podem produzir conclusões incorretas, quando extrapoladas para a Internet.

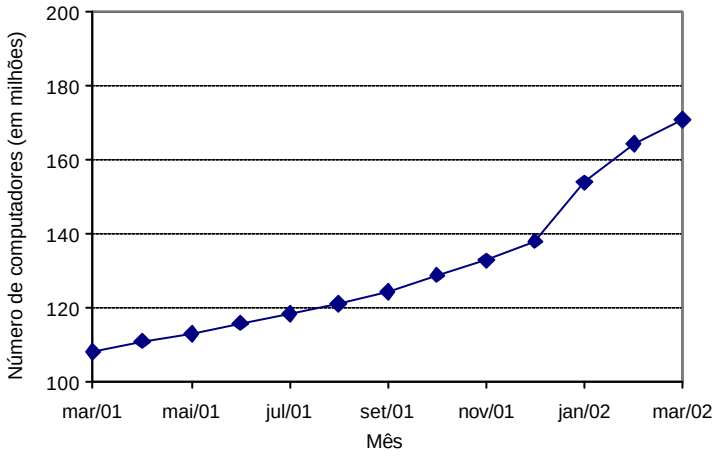


Figura 2.3 – Crescimento do número de computadores na Internet

2.4.1.2. Heterogeneidade

Um dos fatores do grande sucesso da Internet é o protocolo IP, que foi concebido originalmente para interconectar redes, assumindo quase nada a respeito da rede subjacente. Essa característica facilita imensamente a inclusão de várias tecnologias diferentes de rede, o que vem possibilitando que praticamente todas as redes (que desejarem) existentes no mundo sejam conectadas à Internet. No entanto, o seu sucesso em proporcionar conectividade, causa uma grande dificuldade em compreender o seu comportamento [13], porque cada tecnologia de rede apresenta um comportamento que lhe é totalmente peculiar.

Como não existe uma topologia "típica" da Internet, a simulação de protocolos e tecnologias cujo desempenho depende de questões topológicas pode no máximo chegar a conclusões locais. Não é possível extrapolar esses resultados para a Internet global. Outra diferença diz respeito às características dos enlaces, que podem variar de linhas antigas de 9600 Kbps para até 10 Gbps em algumas redes avançadas de pesquisa. As características de atraso e perda de pacotes também variam consideravelmente.

Outras dificuldades dizem respeito às diferenças de protocolos e padrões de tráfego. Caracterizar corretamente o algoritmo de controle de congestionamento do protocolo TCP é um exemplo importante (ver exemplo na seção 2.6.4). A geração de tráfego também representa um dos grandes problemas para modelar e simular a Internet. Em primeiro lugar, deve-se utilizar a proporção correta de tráfego de cada tipo de protocolo. Esses dados podem ser obtidos através de medições, mas são sempre específicos de alguma localidade [34]. Além disso, cada aplicação e/ou protocolo gera tráfego de uma maneira diferente [8]. Com relação a isso, ainda não existem conclusões sobre quais os modelos de tráfego e distribuições que melhor caracterizam os vários tipos de aplicações existentes na Internet.

2.4.1.3. Mudanças drásticas

Outra grande dificuldade para simular a Internet são mudanças drásticas que ocorrem de maneira rápida e imprevisível. Muitos estudos revelam resultados surpreendentes, como crescimento súbito do tráfego em determinada situação e depois uma diminuição também expressiva a níveis semelhantes aos do início do crescimento. Um exemplo de crescimento imprevisível foi o aparecimento de aplicações de transferência de arquivos MP3 (como Napster e Gnutela), ocorrido em 1999. Em alguns casos, o crescimento do tráfego foi tão intenso que muitos administradores de rede limitaram a quantidade de tráfego gerado por essas aplicações. Atualmente, essas aplicações já não parecem estar com o mesmo nível de crescimento. Até mesmo houve uma certa queda, com os problemas enfrentados pelo Napster.

Algumas áreas que podem apresentar mudanças imprevisíveis são: estruturas de tarifação, escalonamento de pacotes, tecnologias de redes sem fio, dispositivos móveis com menor poder computacional, qualidade de serviço (QoS) e *cache* de Web. A implicação dessas mudanças na simulação é que previsões futuras baseadas em simulações podem ficar totalmente inválidas com uma mudança brusca de comportamento da Internet.

2.4.2. Requisitos de simulação para pesquisa

Através de simulação é possível avaliar protocolos de rede sob uma grande variedade de condições. O estudo de protocolos sob condições variadas é um elemento crítico para compreender o seu comportamento e suas características. Pesquisadores necessitam de simuladores que lhes permitam avaliar os diversos aspectos que compõem uma rede complexa como a Internet. Alguns requisitos desses simuladores são descritos a seguir.

2.4.2.1. Abstração

A granularidade da simulação indica o nível de detalhamento que algum elemento a ser simulado é representado. Permitir diversos níveis de granularidade em um único simulador possibilita a realização de simulações de alto nível ou detalhadas. Uma dúvida comum de

iniciantes na simulação é: a que nível um determinado componente deve ser representado na simulação? Freqüentemente não é fácil a decisão entre implementar detalhadamente as especificações dos protocolos, isto é, usar um nível baixo de abstração, ou omitir certas partes elevando o nível de abstração. Na prática, o que está em jogo é a precisão dos resultados contra o tempo de desenvolvimento e de simulação. Cada abstração (ou simplificação) sacrifica algum nível de detalhe para economizar memória e/ou tempo de processamento, de modo que os pesquisadores deveriam utilizar esta técnica somente quando for adequado. Aumentar o nível de abstração permite executar simulações maiores ou em um tempo menor, enquanto que diminuí-lo torna as simulações mais realistas. O quanto é possível usar abstração em uma simulação depende do cenário simulado. Alguns exemplos são:

- **Redes locais**: Quando se está simulando tecnologias de acesso ao meio físico, um grande nível de detalhes é necessário. No caso de IEEE 802.3, por exemplo, é imprescindível que cada estação execute algoritmo CSMA/CD. No entanto, quando o que ocorre na rede local não é importante, é possível simplesmente conectar diretamente as estações ao roteador, eliminando o processamento pesado. Em alguns casos, é possível substituir centenas de estações por poucas, que geram tráfego agregado.
- **Protocolos de aplicação**: Quando o interesse está em protocolos de enlace, rede ou transporte, o funcionamento exato dos protocolos de aplicação não precisa ser representado em detalhes. Nesse caso, uma entidade que gera tráfego semelhante ao protocolo de aplicação é suficiente. O conteúdo dos pacotes (a mensagem) também na grande maioria das vezes não é importante.
- **Protocolos de roteamento**: O roteamento dinâmico nem sempre é necessário. Em cenários onde as rotas não mudam durante a simulação, pode-se calculá-las somente uma vez no início. Quando o interesse da simulação é em roteamento ou as rotas mudam dinamicamente, então a troca de pacotes e o processamento dos protocolos de roteamento (construção de grafos) certamente serão necessários.

2.4.2.2. Emulação

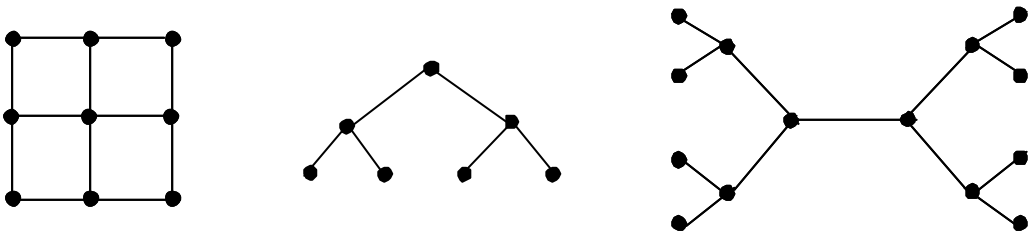
A maioria das simulações é limitada ao mundo simulado que utiliza somente protocolos e algoritmos do simulador. Em contrapartida, emulação permite que um simulador interaja com o mundo real, trocando pacotes com nós reais da rede (sistemas finais e roteadores) [16]. Os pesquisadores usam a emulação para auxiliar a experimentação e a simulação. Na experimentação, a emulação pode introduzir a dinâmica de pacotes (por exemplo, descartes, reordenação e atrasos) que é difícil de reproduzir com confiança em sistemas reais. Além disso, os pesquisadores podem capturar *traces* de tráfego real e utilizar ferramentas de visualização para a avaliar o comportamento dos sistemas. No caso da simulação, é possível verificar o comportamento de elementos de rede (por exemplo, mecanismos de filas) com tráfego gerado por uma aplicação real. Ou seja, pode-se fazer experimentação em um sistema real e fazer o tráfego passar por uma máquina que recebe os pacotes, emula o comportamento de uma rede inteira e depois devolve-os novamente para a rede. Outra opção é posicionar o emulador nos sistemas finais, fazendo com que uma aplicação no simulador interaja diretamente com uma aplicação real.

2.4.2.3. Geração de cenários

Testar protocolos e mecanismos sob condições apropriadas é fundamental para a obtenção de resultados confiáveis. Na maioria das simulações realizadas sobre a Internet, os cenários são configurados manualmente. No entanto, a criação automática de padrões de tráfego, topologias e eventos dinâmicos (falhas em enlaces) complexos podem auxiliar a geração de tais cenários. Particularmente, a avaliação da robustez de protocolos é mais confiável com cenários gerados automaticamente. Isso permite que pesquisadores abordem áreas maiores do espaço operacional dos protocolos (situações enfrentadas na realidade) do que é possível através de configuração manual.

Topologias

Simulações que pretendem abordar temas importantes, como o comportamento de protocolos interagindo com outros na Internet, não podem se basear em topologias simples e não representativas. Topologias típicas de redes locais, como anel, estrela e barramento, devem ser evitadas para simular protocolos da Internet. A Figura 2.4 mostra algumas topologias freqüentemente utilizadas. Nessa figura, os nós representam roteadores ou sistemas finais. A topologia em malha (Figura 2.4a), também conhecida como Manhattan, é alguma vezes utilizada para simular alguns aspectos específicos em topologias bem comportadas. Essa topologia deve ser evitada, pois sua regularidade não representa a realidade das redes existentes. A topologia em árvore (Figura 2.4b) é mais representativa, pois a interligação de roteadores freqüentemente apresente tal estrutura. Uma variação da topologia em árvore, conhecida como halteres (Figura 2.4c), é muito utilizada em simulações de redes. Nessa topologia, os sistemas finais estão conectados a roteadores, que se conectam a roteadores de um nível maior que agrega o tráfego de vários outros. Isso ocorre sucessivamente até atingir o gargalo da rede, representado geralmente por um enlace de capacidade muito inferior aos demais. A topologia em halteres geralmente apresenta simetria, o que lhe confere uma certa regularidade (e a torna mais fácil de trabalhar). Por outro lado, a Internet não possui uma estrutura específica (muito menos ela é regular e simétrica).



(a) **(b)** **(c)**

Figura 2.4 – Topologias comuns; a) malha; b) árvore; c) halteres

A Internet é formada por um grande número de redes (também chamadas de domínios, ou Sistemas Autônomos), que se interligam através de padrões ainda não totalmente compreendidos [24]. Os provedores não gostam de revelar a sua estrutura de interconexão [30], portanto a informação disponível atualmente vem de inferências, principalmente de tabelas de roteamento BGP (*Border Gateway Protocol*). Mesmo que em algum momento esse padrão de interconexão possa ser capturado, em pouco tempo ele pode não ser mais representativo, pois não existe nenhum padrão para o crescimento das redes. Internamente,

cada rede possui uma topologia que lhe é peculiar, geralmente irregular. Um fator de semelhança entre as redes é, no entanto, a existência de uma espinha dorsal (*backbone*), para otimizar os recursos. Quando a Internet deixou de ser financiada pelo governo americano, foi projetada uma estrutura hierárquica para a interconexão de redes. Redes locais se ligariam com provedores de acesso, que se ligariam com provedores regionais que finalmente se ligariam com provedores nacionais. Essa estrutura foi montada, mas logo se verificou que muitas vezes o tráfego tinha que fazer um caminho muito grande para trafegar entre dois pontos muito próximos. Então, logo começaram a aparecer atalhos nessa estrutura, representados pelas ligações diretas entre provedores, conhecidas como relacionamento de parceria (*peering*). Provedores nacionais deveriam se interligar em Pontos de Troca de Tráfego (PTT, ou NAP, *Network Access Point*).

Portanto, resultados confiáveis sobre simulação da Internet dependem de topologias representativas, que não são fáceis de serem criadas manualmente. A geração automática de topologias surge como uma solução. Em geral, os modelos (e geradores) de topologias para a Internet podem ser classificados em duas categorias [25]. A primeira categoria inclui os modelos *ad-hoc*, baseados principalmente suposições relativamente bem embasadas, como os geradores GT-ITM [7] e Tiers [11]. Ambos modelam a Internet como uma estrutura hierárquica, organizando as redes como níveis (usam três níveis) de provedores de serviços. A segunda categoria inclui modelos baseados em medições, por exemplo, a distribuição do número de interconexões dos roteadores externos das redes. Eles se baseiam em algumas características como o crescimento incremental da rede e a preferência dos novos nós em se ligarem com nós de nível mais alto. Os geradores BRITE [25] e Inet [21] se baseiam em propriedades de conectividade das redes. Atualmente acredita-se que os geradores baseados em medições criam topologias mais semelhantes à Internet.

Modelos de tráfego

O tráfego em simulações não é gerado por aplicações reais na maioria das vezes (algumas vezes se usam *traces*). Quando se está projetando ou avaliando algum protocolo ou mecanismo, deve-se utilizar modelos que simulem corretamente o tráfego gerado por aplicações, para que os resultados não sejam invalidados por suposições incorretas. Em geral, a menos que se esteja modelando alguma aplicação, o importante em um modelo de tráfego são as características dos pacotes gerados: tamanho, protocolo de transporte, periodicidade, rajada etc.

As características dos pacotes gerados dependem do cenário que se deseja simular e dos protocolos utilizados. É importante que um simulador possa gerar tráfego de vários protocolos de aplicação que utilizam o protocolo de transporte TCP, como HTTP (tráfego web), SMTP (correio eletrônico), FTP (transferência de arquivos) e TELNET/SSH (conexão remota). O protocolo TCP atualmente é responsável por mais de 90% do tráfego da Internet. Além disso, mesmo para simulações de redes multi-serviços, onde o foco são as aplicações multimídia (que não usam TCP), protocolos de aplicação que usam TCP são geralmente utilizados para gerar tráfego de retaguarda (*background*), cujo objetivo é competir com o tráfego multimídia, submetendo-o a interferências reais. Simulações de aplicações multimídia requerem modelos de tráfego específicos, principalmente para voz e vídeo. Freqüentemente são utilizados modelos de taxa constante de bits, ou CBR (*Constant Bit Rate*) para modelar

voz e vídeo, porque algumas aplicações realmente geram esse tipo de tráfego. Um modelo mais elaborado para tráfego de voz é o *On-Off*, que captura o tempo em que os interlocutores falam (*On*) e o tempo em que ouvem (*Off*). Para vídeo, os modelos mais representativos geram tráfego compactado, com taxa variável de bits, ou VBR (*Variable Bit Rate*). Os modelos VBR podem variar de acordo com a aplicação, como videoconferência ou vídeo sob demanda (VoD). Aplicações de jogos de ação em redes possuem também padrões de tráfego distinto, com baixa taxa de dados [23].

De acordo com o nível de abstração desejado, o tráfego pode ser gerado individualmente para cada fonte ou então agregado para várias fontes em conjunto, através de modelos diferentes. Essa questão deve ser analisada com cuidado, uma vez que gerar tráfego agregado consome menos recursos de memória e processamento.

Dinâmica de rede

Em qualquer topologia, uma rede é dinâmica se um ou mais nós ou enlaces na topologia podem ficar fora do ar periodicamente e possivelmente retornar ao seu estado operante. Existem vários modelos para dinâmica de rede que podem ser usados para especificar quando um determinado nó ou enlace vai estar no ar ou fora do ar. Essa configuração pode ser feita manualmente, através de um *trace* de ocorrências passadas em redes reais, ou através de distribuições que modelam a dinâmica da rede. Cada vez que um nó ou enlace passa de um estado para outro (no ar ou fora do ar) ocorre uma mudança na topologia da rede e então um protocolo de roteamento deve ser usado para calcular as novas rotas.

Como na Internet as alterações de topologias são comuns (e conseqüentemente, alteração de rotas), a dinâmica da rede deve ser levada em consideração quando se estiver fazendo uma simulação muito grande. Também pode ser utilizada para simular o crescimento da rede durante uma simulação. Outra utilização é para testes do comportamento específico de certos protocolos com relação às alterações na topologia e no roteamento. Obviamente, a avaliação de protocolos de roteamento também necessita de dinâmica de rede.

2.4.2.4. Visualização

Visualizar a simulação é uma ferramenta útil para auxiliar a compreensão do cenário simulado. Inclui a visualização da topologia e a animação da simulação, que mostra o tráfego de pacotes. Em simuladores comerciais geralmente visualização é uma característica básica, pois são baseadas em interface gráfica para construção de cenários e visualização da animação. Em simuladores utilizados pela comunidade científica, por outro lado, freqüentemente os cenários são configurados através de arquivos de entrada e os resultados da simulação são arquivos de saída, que geram tabelas e gráficos.

Pesquisadores necessitam de ferramentas que os ajudem a entender o comportamento complexo de simulações de rede. Algumas vezes, somente mostrar tabelas e gráficos de desempenho não descreve adequadamente o comportamento da simulação. Arquivos de *traces* gerados pela simulação são ainda mais difíceis de serem interpretados. Nesses casos, uma ferramenta de visualização permite descobrir características ocultas por valores brutos ou estatísticos. Geralmente a visualização é útil para auxiliar a compreensão de comportamentos inesperados, respondendo a perguntas como: a) por que a vazão está diminuindo ão

drasticamente a partir do tempo de simulação 40 segundos? ou b) por qual caminho os pacotes estão sendo encaminhados, uma vez que não estão passando pelo roteador 5?

Por elucidar questões de difícil compreensão, a visualização é útil para usuários iniciantes, auxiliando a compreensão do funcionamento dos protocolos e a dinâmica da simulação em geral. Outra grande aplicação para a visualização é o ensino de redes.

2.4.2.5. Possibilidade de expansão

O simulador deve poder ser facilmente estendido para adicionar novas funcionalidades, explorar uma grande variedade de cenários e estudar novos protocolos. Essa característica é fundamental para pesquisadores, pois na maioria das vezes o simulador não está preparado para executar as simulações com as características desejadas. Isso fica mais óbvio no caso em que o pesquisador está usando simulação para avaliar um protocolo que ele está propondo ou alterações a protocolos já existentes. Simuladores rígidos, por mais completos, amigáveis e precisos que sejam, não atendem às necessidades dos usuários nesses casos.

2.4.2.6. Disponibilidade de protocolos e mecanismos

Muitas vezes a escolha de um simulador de rede decorre da disponibilidade de protocolos e mecanismos que venham embutidos com o produto. Essa característica, aliada com a possibilidade de expansão, possibilita validar idéias rapidamente e com segurança. Utilizar software que já foi testado e depurado por várias pessoas também proporciona maior confiança nos resultados, como é o caso no *ns* (vide seção 2.5).

2.4.3. Configurações para simular a Internet

Algumas configurações básicas de simulação, apesar de triviais para pessoas experientes, geram grandes dúvidas em iniciantes, que freqüentemente criam cenários irreais. Abaixo é apresentada uma lista (certamente incompleta) de dicas para configuração.

- **Topologias**: Embora topologias pequenas e regulares não sejam representativas, não é recomendável iniciar com uma topologia grande e complexa. Um dos grandes desafios é compreender o que está se passando na simulação. Por isso, a sugestão é sempre iniciar com a menor topologia possível. Quando se quer simular uma rede com gargalo, uma boa dica é a topologia em halteres, que pode ter tantos níveis quanto for desejado (Figura 2.4c), resguardadas as suas restrições. Para grandes topologias, a melhor opção é utilizar algum gerador de topologias.
- **Aplicações e protocolos**: Existem algumas condições básicas que devem ser observadas. Aplicações multimídia quase sempre utilizam o protocolo UDP. Aplicações adaptativas (web, e-mail, ftp), usam o protocolo TCP. É muito comum usar FTP para simular o comportamento do protocolo TCP e/ou encher a rede com tráfego de retaguarda. No entanto, aproximadamente 65% do tráfego da Internet é de web (protocolo HTTP) e isso deve ser refletido nas simulações.
- **Modelos de tráfego**: Para gerar o modelo correto de tráfego é importante conhecer as distribuições que compõem os vários elementos daquele modelo e os parâmetros típicos a serem utilizados. Em uma aplicação de telefonia, por exemplo, a taxa de chegada de chamadas é um processo de Poisson, geralmente com média de 2 minutos. Tráfego de voz é melhor modelado como On-Off, sendo períodos On e Off exponencialmente

distribuídos com durações médias de 1,004 e 1,587 segundos, respectivamente. Cada fonte gera tráfego CBR de 80 Kbps nos períodos On e 0 Kbps nos períodos "off", utilizando uma codificação PCM. Para vídeo VBR existem vários modelos, mas em geral são todos complexos [19]. Uma técnica simples para gerar tráfego VBR pode ser encontrada em [4]. Configurações para simular tráfego web podem ser encontradas em [5].

- **Fluxos e agregações**: A quantidade de fontes e destinos na simulação depende da decisão de configurar fluxos individuais ou agrupar vários fluxos em uma agregação. Outra decisão importante é se devem ser criados vários nós para representar os sistemas finais e configurar um fluxo de dados em cada um deles, ou assumir que um único roteador representa toda uma rede e configurar todas as fontes partindo dele. Em geral é melhor utilizar as configurações mais simples, excetuando-se os casos onde os detalhes são importantes ou têm grande influência na validação do modelo. Pode-se iniciar com poucas fontes individuais e algumas agregações e depois aumentar gradativamente caso os modelo não esteja sendo considerado representativo. Uma questão muito importante é que agregações são modeladas de maneira diferente que tráfego individual: elas não podem ser formadas simplesmente configurando fontes individuais com taxas maiores.
- **Tamanho de pacotes**: Um pacote IPv4 pode ter no máximo 64 KB, mas na prática o tamanho máximo encontrado não vai além dos 1500 bytes. Isso se deve ao tamanho máximo do quadro Ethernet, que compõe a grande maioria das redes onde estão os servidores. Em geral, quanto maior o tamanho do pacote, melhor a utilização da rede, porque a quantidade de tráfego útil transmitido é maior, com o mesmo tamanho de cabeçalho. Aplicações adaptativas (como e-mail e ftp) usam pacotes do maior tamanho possível, como 1500 ou 1000 bytes. Aplicações interativas geram pacotes pequenos, porque não podem ficar esperando muito tempo para "encher" os pacotes. Isso vale para aplicações de voz, jogos, TELNET/SSH entre outras. Um pacote de voz interativa pode ter algo em torno de 200 bytes, apesar de que o valor real deve ser calculado em função do codificador de voz utilizado. Aplicações de vídeo sob demanda (VoD), que geralmente tem altas taxas de transmissão, também utilizam pacotes grandes. Uma simulação real deve conter pacotes de vários tamanhos.
- **Tempo de simulação**: Uma regra empírica para escolher o tempo de simulação é iniciar com um valor baixo e ir dobrando o tempo para verificar a alteração dos resultados. No momento em que as alterações forem muito baixas (algo subjetivo), pode-se ficar com o valor anterior. Para *tempo de simulação*, dependendo da quantidade de eventos da simulação, algo entre alguns minutos a algumas horas é um valor razoável. O *tempo de relógio* é geralmente menor que o tempo de simulação, mas ele é influenciado pelo número de repetições, necessário para garantir confiabilidade estatística aos resultados. A simulação pode demorar minutos, horas ou dias. Comentários gerais sobre o tempo de simulação podem ser encontrados na seção 2.3.6.2.

2.4.4. Simuladores existentes

Vários simuladores têm surgido nos últimos anos para permitir a avaliação de cenários de rede antes da sua implantação. Entre os simuladores mais utilizados estão OPNET, Parsec, SSF e *ns*. Os quatro simuladores têm características semelhantes: utilizam simulação discreta

baseada em eventos no nível de pacotes e modelam uma grande quantidade de protocolos da família TCP/IP. Porém, cada simulador em um foco diferente.

O OPNET [29] é um simulador comercial largamente utilizado no âmbito corporativo, principalmente devido ao seu altíssimo custo. Os outros três estão mais direcionados ao projeto de protocolos e mecanismos de rede. O Parsec [3] destina-se principalmente a simulações de redes sem fio e permite simulações paralelas de alto desempenho. SSF [9] também permite simulações paralelas, mas está mais direcionado a simulação de protocolos de roteamento e simulações de larga escala. O *ns* [26] está sendo muito utilizado no meio acadêmico e por isso é abordado com mais detalhes na seção 2.5.

2.5. O simulador de redes *ns*

O *ns* (Network Simulator-2) [27] é um simulador baseado em eventos discretos e orientado a objetos para a simulação de redes. O objetivo do *ns* é proporcionar um ambiente para o desenvolvimento de pesquisas em torno dos protocolos que constituem a Internet, isto é, que utilizam a pilha TCP/IP, tanto no contexto das redes fixas quanto móveis, com fio e sem fio (redes locais e de satélite).

2.5.1. Características do simulador ns

O *ns* tem sua origem a partir do simulador REAL (*Realistic and Large*) [23], que por sua vez derivou do NEST (*Network Simulation Testbed*) da universidade de Columbia, NY. REAL foi desenvolvido por S. Keshav da Universidade de Cornell com o intuito de ser uma ferramenta para o estudo do comportamento dinâmico do controle de fluxo e congestionamento em redes comutadas por pacotes. Posteriormente, o *ns* surgiu como uma variante do REAL e em 1995 seu desenvolvimento foi suportado como parte do projeto VINT (*Virtual InterNetwork Testbed*) (netweb.usc.edu/vint). Atualmente seu desenvolvimento e distribuição são mantidos pelo ISI (*Information Sciences Institute* - www.isi.edu), financiado pela DARPA e NSF. A distribuição do *ns* é gratuita, inclusive o código fonte, que pode ser alterado para refletir a pesquisa que está sendo desenvolvida [13].

O *ns* foi desenvolvido (portado) para várias plataformas computacionais, sendo possível instalar o pacote em vários sabores Unix como, FreeBSD, Linux, SunOS e Solaris, além da plataforma Windows.

A biblioteca de protocolos e mecanismos implementados no *ns* é bastante vasta, abrangendo implementação dos protocolos TCP, UDP, IP além de disciplinas de serviços, como, WFQ (*Weight Fair Queueing*), protocolos para redes móveis, como o IP móvel, tecnologias de redes sem fio locais e de longa distância, como, 802.11, Bluetooth e GPRS (*General Packet Radio Service*). Alguns destes não são distribuídos diretamente no pacote do *ns*, sendo contribuições disponibilizadas como *patchs* na página que hospeda as informações sobre o *ns* e que podem ser baixadas e adicionadas ao módulo básico através de recompilação do núcleo do simulador.

O *ns* fornece também bibliotecas de funções para a geração de alguns tipos de tráfego como: CBR (*Constant Bit Rate*) utilizado para simular tráfego constante e voz, ON-OFF para tráfego em rajada e voz comprimida, FTP para gerar tráfego correspondente a aplicações de transferência de arquivos e VBR (*Variable Bit Rate*) para tráfego com taxa de dados variável. Além das bibliotecas com os módulos específicos de protocolos, tecnologias e

geração de tráfego, o *ns* possui funções específicas de simulação e geração de números aleatórios.

2.5.2. O modelo de programação do *ns*

Ao invés de adotar uma única linguagem de programação, o projeto do simulador levou em consideração o fato de que diferentes simulações requerem diferentes modelos de programação. Portanto, o ns adota duas linguagens de programação: C++ para o núcleo do simulador (*back-end*) e Otcl para construção de *scripts* e modelagem da simulação (*fron-end*). O objetivo é prover flexibilidade sem prejudicar o desempenho.

O núcleo do simulador implementa tarefas que exigem processamento de alto desempenho como o processamento de eventos de baixo nível e tratamento de encaminhamento de pacotes através de um roteador simulado. Possui bibliotecas com uma quantidade razoável de protocolos implementados e funções de controle da simulação para escalonamento de eventos. Estes aspectos são mais eficazes se implementados em um a linguagem compilada.

Os *scripts* que contém os comandos e o modelo a ser simulado são construídos em Otcl (Object Tool Control Language), uma versão orientada a objetos da linguagem de script tcl (Tool Control Language). Otcl é interpretada e um dos motivos porque foi escolhida é que os scripts são tarefas interativas e freqüentemente refinadas (alteradas) no programa de simulação, sem necessidade de recompilação.

Em função dos várias contribuições realizadas por diversos grupos de pesquisa ao redor do mundo, em boa parte dos casos, o usuário precisará conhecer apenas os blocos básicos (comandos, classes dos objetos que implementam os protocolos na linguagem de scripts OTcl) para a construção de uma simulação. Contudo, quando o usuário desejar incluir novas características a protocolos implementados na distribuição do *ns* ou, ainda, implementar novos protocolos, é aconselhável e, mais eficaz, a implementação direta utilizando C++.

A implementação de novos mecanismos e protocolos em C++ é possível porque praticamente todo objeto C++ tem um correspondente Otcl e vice-versa. Pode-se implementar um novo protocolo através de herança de classes existentes em C++ e depois ligar o novo objeto C++ a um objeto Otcl correspondente. O novo objeto Otcl, resultado da ligação citada anteriormente, poderá então ser invocado a partir de scripts Otcl. Assim, será possível executar as novas funções implementadas em C++ através do desenvolvimento de scripts de simulação em Otcl. Estes scripts conterão uma chamada a um objeto Otcl ligado a um objeto C++ incluído pelo usuário no back-end do *ns*. Além da possibilidade de chamar funções em C++ a partir do Otcl pode-se também chamar funções definidas em Otcl no C++.

2.5.3. Fases no desenvolvimento da pesquisa utilizando o *ns*

Para o desenvolvimento das pesquisas com o *ns*, o usuário deve estar a par dos componentes usados em cada fase do processo de simulação, desde a concepção do modelo até a análise dos resultados.

Inicialmente, o usuário deve implementar seu modelo de simulação utilizando Otcl a partir dos objetos (ex: protocolos, aplicações) existentes na biblioteca *ns*. Quando o usuário desejar implementar novos protocolos ou estender os já existentes na biblioteca do *ns*, deverá utilizar C++ e, posteriormente, utilizar as novas funções chamadas através de scripts OTcl.

O programa de simulação desenvolvido alimentará um interpretador OTcl e serão gerados arquivos de traces formatos que conterão os resultados da simulação. Comandos e métricas a serem coletadas para a geração da saída da simulação deverão ser explicitamente incluídos no script OTcl. Outras saídas da simulação também são possíveis. Pode-se enviar os dados coletados em arquivos de trace diretamente para o ferramentas que geram gráficos como o xgraph. Além disso, pode-se gerar um tipo de arquivo de trace para o animador de redes *nam* (Network Animator) que é uma ferramenta importante para a visualização da dinâmica da simulação e para a depuração. O *nam* será visto com mais detalhes na seção 2.5.7.

2.5.4. Vantagens do *ns*

Algumas vantagens que levam à adoção do *ns* são destacadas a seguir.

- Disponibilidade de um simulador padrão para a comunidade acadêmica e científica;
- Simulador de código aberto e gratuito;
- Boa infra-estrutura para desenvolver novos protocolos;
- Grande quantidade de protocolos e tecnologias existentes (*multicast*, redes *ad-hoc*, MPLS (*Multiprotocol Label Switching*), redes de satélites etc);
- Oportunidade para estudar interações de protocolos em um ambiente controlado.

2.5.5. Limitações do *ns*

O *ns* tem algumas limitações que são ser desvantagens em alguns casos. O fato de ter sido originalmente desenvolvido para o estudo da Internet, especificamente, de redes IP, limita sua aplicação para o estudo de protocolos/tecnologias sub-IP.

Outra limitação é a curva de aprendizado difícil e a falta de documentação adequada para iniciantes. O desenvolvimento de modelos simples requer conhecimento em Otcl que não é complexa, mas pouco conhecida. À medida que se necessita desenvolver novos protocolos ou mecanismos não presentes no *ns*, deve-se ter não apenas intimidade com o desenvolvimento em C++, mas desvendar a estrutura de dados do *ns* e seu esquema de hierarquias de classes para saber de onde novas classes podem ser derivadas e encaixadas, o que não é uma tarefa simples para um iniciante no *ns*.

Apesar da distribuição do *ns* possuir uma versão para a plataforma Windows, preferencialmente, adota-se a plataforma de desenvolvimento FreeBSD/Linux, na qual o simulador apresenta um funcionamento mais robusto, pelo fato de ser a plataforma nativa de desenvolvimento do *ns*.

2.5.6. Componentes básicos

Os elementos básicos para a criação de uma simulação utilizando o *ns* são:

Nó (*node*): Nós são elementos que implementam a lógica associada a *hosts* e roteadores em uma topologia. Estes elementos implementam o protocolo IP, possuindo desta forma as funcionalidades da camada de rede no contexto arquitetura de TCP/IP para tratar endereçamento e roteamento. É importante citar que o *ns* foi desenvolvido para o estudo de redes IP e, com tal objetivo, a implementação do elemento nó abstrai as características específicas das tecnologias de rede (Ethernet, ATM, FDDI, etc). Dessa forma, os nós não implementam as camadas inferiores da pilha de protocolos.

Enlace (*link*): É o elemento que interconecta dois nós. Estes elementos servem como abstração da interface física para enlaces ponto a ponto, meios de difusão e enlaces sem fio. Na maior parte dos modelos de simulação utilizando *ns*, as tecnologias sub-IP não interessam e, por isso, as filas que, normalmente, esperava-se encontrar nos nós (roteadores), estão presentes no elemento enlace. Dessa forma, quando deseja-se obter algum resultado de desempenho com relação a métricas de perda e atraso, por exemplo, o elemento monitorado deve ser o enlace e não o nó.

Agente (*agent*): Como os nós não implementam diretamente nenhum protocolo da camada de transporte, este papel é desempenhado pelos agentes. Os dois agentes definidos no *ns* correspondem aos protocolos TCP e UDP. O simulador suporta vários outros agentes. Além disso, novos protocolos presentes em várias camadas da arquitetura de rede podem ser implementados a partir da definição da classe *agent* do *ns*.

Aplicação (*application*): Este elemento é responsável por gerar o tráfego de dados para o simulador. Na verdade, o *ns* define alguns modelos de tráfego que são utilizados para simular aplicações, daí o nome dado a este elemento. Alguns dos modelos implementados no *ns* são: exponencial, On-Off e CBR.

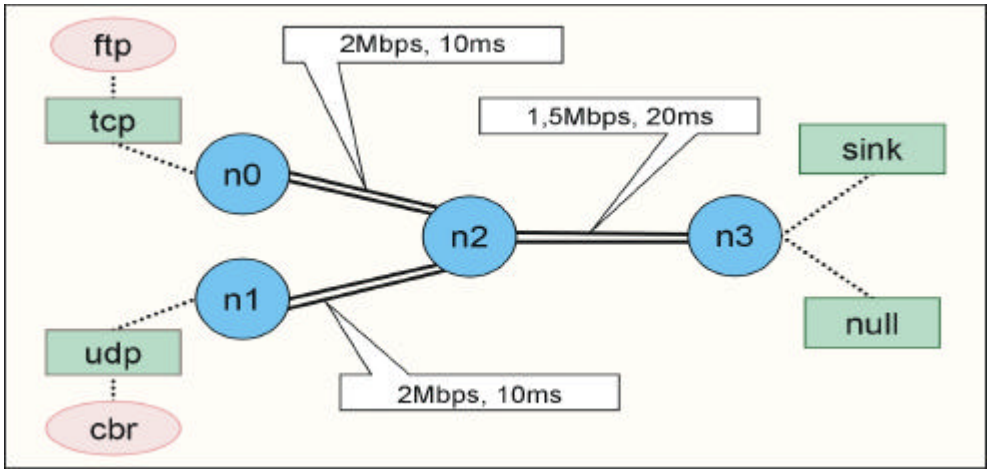


Figura 2.5. Componentes do modelo de simulação.

A Figura 2.5 ilustra como estes elementos podem ser utilizados para criar um modelo de simulação para uma rede IP com quatro nós. Alguns aspectos para configurar uma topologia são importantes e serão ressaltados a seguir. Inicialmente, devem-se criar os elementos do tipo nó (**n0**, **n1**, **n2** e **n3**) que implementam a camada IP. Estes nós devem ser ligados através de elementos do tipo enlace. Neste exemplo, existem três objetos do tipo enlace representados pelas linhas duplas (*full duplex*) que unem **n0** a **n2**, **n1** a **n2** e **n2** a **n3**. Na criação do enlace devem ser especificas as características de capacidade e atraso. No exemplo, os dois nós a esquerda têm enlaces de 2Mbps e atraso de 10ms, já o enlace que liga **n2** a **n3** tem uma capacidade de 1,5Mbps e atraso de 20ms.

Em seguida, os agentes devem ser criados e conectados aos seus respectivos nós: **tcp** conectado a **n0**, **udp** conectado a **n1**, **sink** e **null** conectados a **n3**. tcp e **udp** implementam os protocolos da camada de transporte. Da mesma forma que há a necessidade de ligar os nós através de enlaces, os agentes também precisam ser conectados para que o nó destino da comunicação saiba a que agente entregar os pacotes que chegam. Isto é representado pelas

linhas pontilhadas. O agente tcp presente em **n0** é conectado ao agente **sink** em **n3**. Da mesma forma, o agente **udp** em **n1** é conectado ao agente **null** presente em **n3**.

Como, neste exemplo, o que importa é o tráfego em uma direção (de **n0** e **n1** para **n3**), os dois agentes (**sink** e **null**) ligados ao nó **n3** implementam nenhum protocolo e servem apenas como sorvedouros do tráfego. Caso o modelo exigisse tráfego bidirecional entre **tcp** e **sink**, poderíamos substituir **sink** por outro agente **tcp**.

O último elemento a ser inserido no modelo é a fonte de tráfego que representa a aplicação. O *ns* possui alguns modelos de tráfego que simulam aplicações. No exemplo mostrado, as duas aplicações **ftp** e **cbr** modelam tráfego gerado por uma aplicação de transferência de arquivos e tráfego gerado por uma fonte com taxa constante, respectivamente. As fontes de tráfego devem ser conectadas aos agentes presentes no nó em questão.

2.5.7. O animador de redes *nam*

O *nam* (Network Animator) [12] é um animador de redes que acompanha o *ns* e utilizado para compreender o que ocorre durante a simulação. Através do *nam* pode-se visualizar a topologia da rede, bem como acompanhar o fluxo dos pacotes e seus conteúdos. Quando o *nam* é ativado, apresenta uma console que pode gerenciar várias atividades paralelamente, como animações e criação de simulações. Quando essa console é finalizada, todas as outras janelas são fechadas também.

Durante a simulação, o *ns* gera um ou mais arquivos de *trace* que contêm dados detalhados da simulação, para visualização posterior. A criação destes arquivos é opcional e depende da adição de alguns comandos no script de simulação Otcl. No final da simulação, o *nam* pode ser acionado explicitamente no *script* para interpretar o arquivo de *trace* e mostrar a animação da simulação.

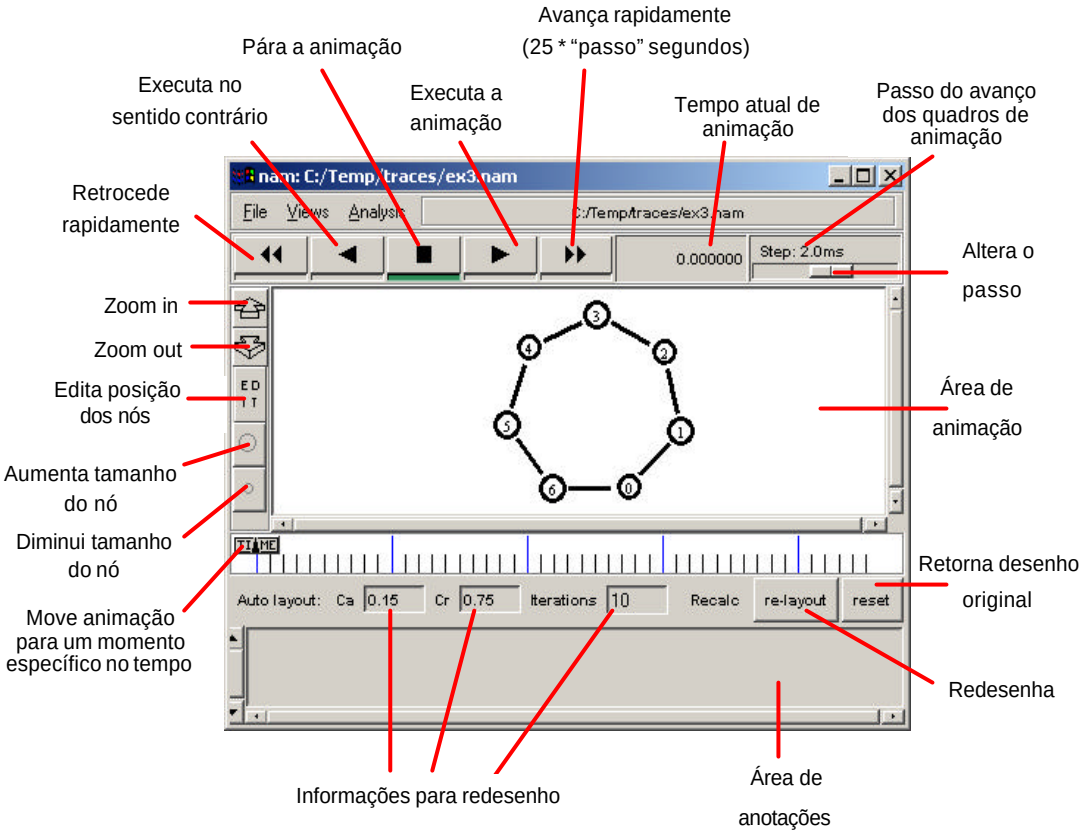


Figura 2.6. *nam – Network Animator.*

O arquivo de *trace* do *nam* contém todas as informações necessárias para animação, tanto do leiaute estático da rede quanto eventos dinâmicos, como saída e chegada de pacotes e quedas de enlaces. A Figura 2.6 mostra uma sessão típica do *nam* com textos explicativos nas suas funções. Para um melhor controle por parte do usuário, o *nam* proporciona uma interface com botões no estilo VCR (*play*, *fast forward*, *rewind*, *stop*). Pode-se controlar o momento particular da simulação que se deseja visualizar, bem como, redefinir a apresentação do desenho da topologia.

O *nam* também proporciona um editor gráfico simples que auxilia na criação de modelos de simulação sem a necessidade de desenvolvimento em código Otcl. Usando facilidades de clicar e arrastar pode-se inserir nós, agentes TCP, enlaces entre nós e incluir algumas fontes de tráfego. Esta é uma facilidade que vem sendo aprimorada a cada nova versão do *ns* e deve auxiliar sobremaneira a reduzir o tempo de desenvolvimento.

Além de servir para a animação de modelos de simulação, o *nam* aceita arquivos formatados de traces de dados gerados por redes reais. Esta é uma característica importante dada a necessidade de obter resultados de desempenho baseados em tráfegos que modelem de forma mais fiel as aplicações existentes.

2.5.8. Aplicações do ns no ensino

Recentemente, o *ns* tem sido reconhecido por educadores do mundo inteiro como uma ferramenta importante no ensino de redes de computadores e da Internet. Existe uma página

[28] devotada a exercícios aos estudantes interessados em aspectos específicos do funcionamento da Internet e dos seus protocolos, além de tutoriais disponíveis on-line.

Exemplos da utilização do *ns* no aprendizado de redes de computadores incluem a visualização da dinâmica dos protocolos. Um exemplo clássico e a verificação do funcionamento do mecanismo de controle de congestionamento do protocolo TCP. Um primeiro passo é seria a criação de scripts que permitam variar parâmetros do protocolo como, por exemplo, tamanhos de janelas de transmissão e temporizadores que controlam a retransmissão de segmentos TCP. Posteriormente, visualizando o resultado da simulação com a variação destes parâmetros através do *nam* pode-se obter uma descrição mais aprimorada dos mecanismos e funcionamento do protocolo TCP. Será possível entender o mecanismo de reconhecimentos positivos (ACK) enviados pelo receptor quando da recepção de um segmento correto; o efeito da mudança do tamanho de janelas de transmissão e o impacto na vazão da aplicação; verificar o efeito de congestionamento nos nós intermediários através da retransmissão de pacotes quando da ocorrência de um estouro de temporizador (*timeout*) etc.

2.6. Exemplos de simulação usando o *ns*

Esta seção apresenta exemplos de simulações com o *ns* visando discutir aspectos que podem ser usados no ensino de redes em geral e da Internet em particular. Dos exemplos apresentados, alguns foram elaborados especificamente para este curso e outros foram modificados a partir de exemplos que acompanham o próprio *ns*.

A estratégia é apresentar um *script* completo que simula algum cenário específico de uma rede e, em seguida, discutir a construção desse *script* é discutido quanto aos seus aspectos da linguagem de programação (como fazer o quê) e quanto à intenção dos comandos usados (por que fazer). Sempre que necessário (e possível) algum conceito teórico é discutido para motivar e facilitar o entendimento dos objetivos da simulação. Depois, mostramos algumas imagens extraídas do visualizador *nam* e discutimos o que aconteceu na simulação, apontando aspectos importantes e relacionando as imagens aos eventos gerados pela interação dos componentes.

2.6.1. Anatomia de um script de simulação no ns

Antes de mostrar exemplos de código, vamos mostrar como é o formato geral de um script *ns*. A seqüência abaixo tenta expor qual o ***roteiro mais comum*** para criar simulações no *ns*, muito embora seja possível a omissão de algumas das partes (exemplo: itens 7, 8) bem como a inversão de alguns outros.

1. Criar instância do simulador
2. Fazer configurações necessárias
   2.1. Ajustar opções gerais do simulador e do *nam*
   2.2. Configurar arquivo de trace do *nam*
   2.3. Criar função de finalização
3. Criar a topologia (nós e enlaces entre os nós)
4. Criar agentes e aplicações
   4.1. Criar agentes de transporte de dados (TCP, UDP)

4.2. Criar aplicações geradoras de tráfego
4.3. Anexar os agentes aos nós e os aplicativos aos agentes
4.4. Conectar agentes para transmissão de dados

5. Escalonar os eventos (geração de tráfego)
   5.1. Agente inicia transmissão de dados
   5.2. Agente finaliza transmissão de dados
6. Executar a simulação
7. Visualizar animação
8. Analisar arquivos de trace

2.6.2. Exemplo 1 – Fluxo CBR competindo com FTP

Abaixo mostramos o código fonte de uma simulação cuja intenção é ilustrar a competição de duas aplicações que geram tráfego em direção a um destino comum. Uma das aplicações envia dados em fluxo constante (CBR) usando o UDP como protocolo de transporte e a outra é uma aplicação FTP, que usa o protocolo de transporte TCP.

```
1  Exemplo-1 – Adaptado de ns/tcl/ex/simple.tcl (exemplo do ns)
2  set ns [new Simulator]
3
4  # arquivos de trace
5  set f [open out.tr w]
6  set nf [open out.nam w]
7
8  $ns trace-all $f
9  $ns namtrace-all $nf
10
11 # criacao de alguns nós
12 set n0 [$ns node]
13 set n1 [$ns node]
14 set n2 [$ns node]
15 set n3 [$ns node]
16
17 # Criacao de alguns agentes
18 set udp0  [new Agent/UDP]
19 set null0 [new Agent/Null]
20 set tcp   [new Agent/TCP]
21 set sink  [new Agent/TCPSink]
22
23 # Criacao de algumas aplicacoes
24 set cbr0  [new Application/Traffic/CBR]
25 set ftp   [new Application/FTP]
26
27 # Criacao de enlaces
28 $ns duplex-link $n0 $n2 5Mb 2ms DropTail
29 $ns duplex-link $n1 $n2 5Mb 2ms DropTail
30 $ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
31
32 # anexando agentes
33 $cbr0 attach-agent $udp0
34 $ns attach-agent $n0 $udp0
35 $ns attach-agent $n3 $null0
36 $ns connect $udp0 $null0
37
38 $ftp attach-agent $tcp
```

```
39  $ns attach-agent $n1 $tcp
40  $ns attach-agent $n3 $sink
41  $ns connect $tcp $sink
42
43  # Imprimindo algumas variáveis na saída
44  puts [$cbr0 set packetSize_]
45  puts [$cbr0 set interval_]
46
47  # escalonando algumas tarefas
48  $ns at 0.1 "$cbr0 start"
49  $ns at 0.5 "$ftp start"
50  $ns at 1.35 "$ns detach-agent $n1 $tcp ; $ns detach-agent $n3 $sink"
51  $ns at 3.0 "finaliza"
52
53  proc finaliza {} {
54          global ns f nf
55          $ns flush-trace
56          close $f
57          close $nf
58          puts "Executando o nam..."
59          exec nam out.nam &
60          exit 0
61  }
62  # Finalmente, inicia a simulação
63  $ns run
```

Na linha 2, o objeto principal da simulação é criado na variável ***ns***. Observe que as variáveis são referenciadas com um dólar antes do seu nome (exemplo: ***$ns***). Nas linhas 5 e 6 dois descritores de arquivo são criados e armazenados nas variáveis ***f*** e ***nf***. Esses arquivos serão usados para armazenar os dados de "trace" do *ns* e do *nam*, respectivamente. Veja que a criação de arquivos no *ns* é feita com a sintaxe "**set** descritor [**open** nome_arquivo modo]". No caso da linha 5, por exemplo, o descritor é ***f***, o nome do arquivo é ***out.tr*** e o modo de abertura é ***w*** (write).

A linha 8 aciona o procedimento ***trace-all*** do objeto ***ns*** e passa como parâmetro a referência para o objeto ***f***. O procedimento ***trace-all*** conecta a saída do trace do *ns* ao arquivo representado por ***$f***, ou seja, ***out.tr***. A linha 9, de maneira semelhante, faz a associação dos dados de trace do *nam* ao arquivo representado pela variável ***$nf***, ou seja, o arquivo ***nam.tr***.

As linhas de 12 a 15 criam 4 instâncias do tipo "node", armazenando-as nas variáveis ***n0***, ***n1***, ***n2*** e ***n3***. Os objetos do tipo "node" são usados para compor a topologia da rede e serão visíveis nas animações do *nam*.

As linhas de 18 a 21 criam "agentes" armazenando-os nas variáveis ***udp0***, ***null0***, ***tcp*** e ***sink***. Os agentes representam pontos da rede onde os pacotes são produzidos e consumidos. Os agentes são usados na implementação dos protocolos em várias camadas da rede. Por exemplo, o agente ***udp0*** implementa o comportamento do protocolo UDP, ou seja, quando ele é usado para transporte de pacotes, ele emula o UDP. O agente ***tcp***, emula a versão "Tahoe" do protocolo TCP (default do *ns*). O agente ***null0*** é um agente especial que apenas recebe e descarta os pacotes. Ele é geralmente usado quando o interesse da simulação não envolve o recebimento dos dados (apenas o envio deles). O agente ***sink*** é um agente TCP

especial que recebe dados de outro TCP, mas não envia dados de volta (novamente, o interesse é voltado apenas para o envio). Entretanto, ao contrário do agente null, o TCPSink envia pacotes de *acknowledgement* de volta para o emissor. Existem outros agentes possíveis e detalhes podem ser vistos em [13].

As linhas 24 e 25 criam dois objetos do tipo "Application". O primeiro deles, ***cbr0***, simula um tráfego CBR, ou seja, um tráfego com taxa constante de envio de dados. Já o objeto ***ftp*** simula uma aplicação FTP. Ambas as aplicações podem ser ajustadas através de algumas variáveis, mas, em nosso caso, os valores *default* serão mantidos. Até o momento, nosso cenário de simulação pode ser representado Figura 2.7. Ou seja, alguns objetos foram criados mas não há conexão entre eles ainda.

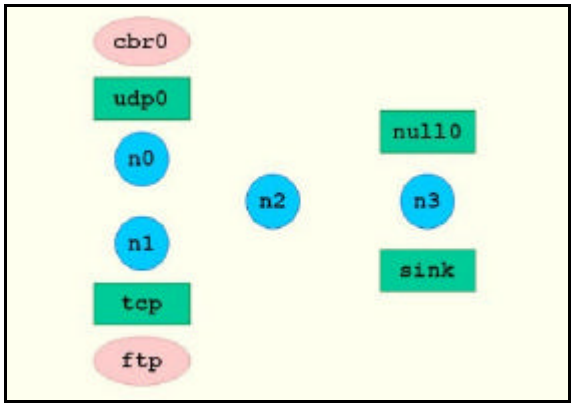


Figura 2.7 – Objetos criados, mas ainda desconectados

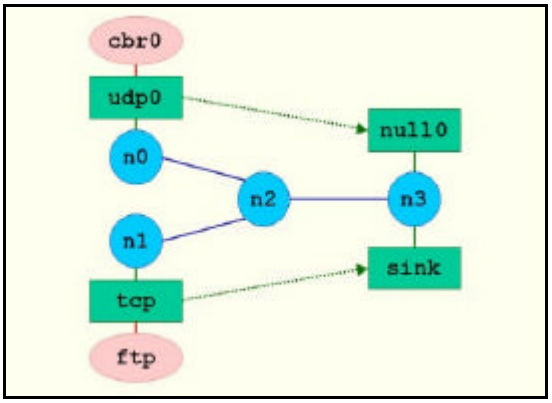


Figura 2.8 – Conexão de nós e agentes

As linhas 28-30 criam 3 enlaces conectando os nós existentes. A criação de um enlace requer, dois nós e a sintaxe é a seguinte: "**$**ns duplex-link **$nx $ny vazão atraso, tipo_de_fila**". No nosso exemplo, ambos os enlaces ***n0-n2*** e ***n1-n2*** têm uma vazão de 5Mb (megabits por segundo), um atraso de 2ms (milisegundos) e uma fila do tipo "droptail". O enlace ***n2-n3*** tem uma vazão de 1.5Mb, atraso de 10ms e fila do tipo "droptail".

Nos enlaces, as filas são lugares onde pacotes podem permanecer antes de serem processados e encaminhados ou simplesmente descartados[5]. As filas simulam o comportamento de um roteador real dentro da rede. "Escalonamento de pacotes" é o termo usado para referir-se ao processo de decisão usado para escolher quais pacotes devem ser processados ou descartados. No caso do método "droptail", os pacotes são enfileirados usando uma disciplina "primeiro a chegar, primeiro a sair" (FIFO) e se a fila enfrentar problemas de falta de espaço para armazenamento de pacotes, os pacotes do fim da fila serão descartados. O *ns* também implementa outros mecanismos para "gerenciamento de buffer", os quais podem ser vistos no manual do *ns* [13].

As linhas 33-36 estão se encarregando de conectar os agentes entre si e aos nós da rede. No *ns* é necessário anexar as aplicações aos agentes e os agentes aos nós. No primeiro caso, temos a sintaxe "**$aplicação** attach-agent **$agente**", que na verdade é a execução do

[5] Em uma rede real, as filas ficam nas interfaces dos roteadores. Entretanto, no *ns* as filas foram implementadas nos enlaces, porque enlaces são abstrações de interfaces.

método ***attach-agent*** contido no objeto ***aplicação***, passando o parâmetro ***$agente***. Traduzindo, o método ***attach-agent*** conecta o objeto ao qual está vinculado (objeto ***aplicação***) ao agente indicado (agente ***agente***). Isso significa que os dados gerados pela aplicação ***cbr0*** serão repassados para o objeto ***udp0*** para serem encaminhados (roteados pela rede) e os dados recebidos da rede pelo agente ***udp0*** serão encaminhados para a aplicação para serem consumidos (na verdade este não é exatamente o caso aqui, pois a aplicação ***cbr0*** apenas produz dados e seu objetivo é gerar tráfego).

A linha 34 está anexando o agente ***udp0*** ao nó ***n0***. Observe que, desta vez, o procedimento ***attach-agent*** pertence ao objeto *ns* e está recebendo dois parâmetros (***$n0*** e ***$udp0***). A linha 35 faz a conexão entre o agente ***null0*** e o nó ***n3***.

A linha 36 está conectando dois agentes: ***udp0*** e ***null0***. O efeito dessa conexão é habilitar o tráfego de informação entre eles, da mesma forma quando dois protocolos estabelecem uma conexão. Uma vez que a conexão está estabelecida entre dois agentes, os dados podem ser encaminhados de um lado para o outro pela rede a partir do momento que estes sejam gerados pela aplicação que está anexada a cada um dos agentes. A conexão estabelecida entre os agentes é uma espécie de conexão "lógica", representada pela seta pontilhada da Figura 2.8. Portanto, nenhum agente é capaz de gerar dados: ele espera que a aplicação gere os dados e seu papel é repassá-los para o seu agente-par através do nó da rede ao qual está conectado.

As linhas 38-41 executam basicamente o mesmo que as linhas 33-36, já discutidas anteriormente, com a diferença de atuam sobre os agentes ***tcp*** e ***sink***, a aplicação ***ftp*** e os nós ***n1*** e ***n3***. Apenas duas observações relevantes: a primeira é que no caso do agente ***tcp*** a aplicação que produz dados é o objeto ***ftp***. A outra observação é que é possível ver que o nó ***n3*** possui dois agentes conectados: ***null0*** e ***sink***. No *ns* não há limite teórico quanto ao número de agentes anexados a um nó. E isso não se constituirá em um problema, desde que as conexões entre os agentes sejam feitas com cuidado. No nosso caso, o agente ***udp0*** está conectado ao agente ***null0*** e o agente ***tcp*** está conectado ao agente ***sink***. Portanto, o *ns* vai saber exatamente para qual agente um pacote deve ser despachado quando este chegar ao nó ***n3.*** O cenário de conexão entre os objetos agora está completo (vide Figura 2.8).

As linhas 44-45 imprimem na saída o conteúdo das variáveis ***packetSize_*** e ***interval_***, ambas do objeto ***cbr0***. Essas variáveis contêm um valor *default* e poderiam ser alteradas a qualquer momento pelo usuário para mudar as características do fluxo de dados CBR a ser gerado pelo objeto. A variável ***packetSize_*** controla o tamanho do pacote gerado (em bytes) e a variável ***interval_*** controla o intervalo de tempo entre a geração de pacotes (em segundos). O ajuste da variável ***interval_*** define a taxa de bytes por segundos a serem enviados, mas é possível também ajustar diretamente uma outra variável chamada ***rate_*** se você preferir.

As linhas 48-51 fazem a programação de eventos da simulação. A sintaxe padrão é "**$obj_**ns at **tempo evento**", onde ***$obj_***ns deve ser um objeto do tipo "simulador", o tempo é dado em segundos e o evento pode ser qualquer comando do *ns*, contido entre aspas. Novamente, o que estamos fazendo é executando o procedimento ***at*** associado ao objeto *ns*, passando para ele dois parâmetros: o tempo (em segundos, relativo ao início da simulação) e

o evento a ser escalonado. O procedimento ***at*** se encarrega de agendar o evento, sem verificar sua consistência, o que será feito apenas no momento da sua execução.

O primeiro evento é escalonado para o tempo 0.1s após o início da simulação e trata-se da ativação do procedimento ***start*** do objeto ***cbr0***. Isso significa que o objeto ***cbr0*** irá começar a produzir dados. O segundo evento é agendado para 0.5 segundos do início da simulação e irá ativar o procedimento ***start*** para o objeto ***ftp***, que iniciará a aplicação de transferência de arquivos. Ao tempo 1.35s, um evento "composto" por 2 comandos é agendado (observe o "ponto-e-vírgula" entre dois comandos). O primeiro deles desconecta o agente ***tcp*** do nó ***n0*** e o segundo desconecta o agente ***sink*** do nó ***n3***. No tempo 3.0s, o procedimento ***finaliza*** será executado (disctido adiante). Observe que os eventos estão programados, mas somente serão executados no tempo marcado, em estrita ordem cronológica, após o início da simulação.

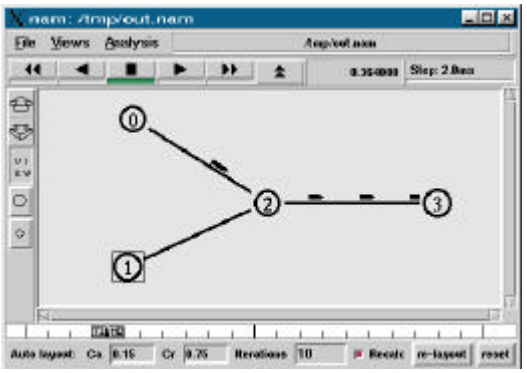


Figura 2.9 – tráfego CBR inicia no tempo 0.1s do nó 0 para o nó 3

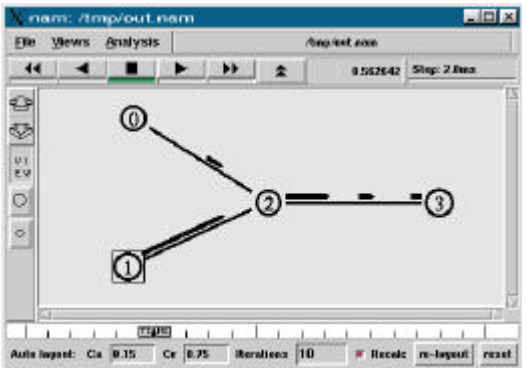


Figura 2.10 – tráfego FTP inicia no tempo 0.5s do nó 1 para o nó 3.

As linhas 53-61 definem o corpo do procedimento ***finaliza***, que contém comandos relacionados com a finalização da simulação. A palavra ***proc*** é reservada para definição de procedimentos e o nome ***finaliza*** é o nome atribuído ao procedimento. O par de chaves "{}" após o nome ***finaliza*** indica que o procedimento não recebe parâmetros. A chave aberta "{" no final da linha 53 inicia o corpo de comandos do procedimento que é fechado pela chave "}" da linha 61. O comando ***global ns f nf*** declara as variáveis ***ns***, ***f*** e ***nf*** como tendo sido criadas fora do escopo do procedimento, permitindo que sejam manipuladas dentro do procedimento (sem isso, elas não poderiam ser referenciadas corretamente). Na linha 55 o procedimento ***flush-trace*** é acionado para gravar qualquer trace pendente nos buffers do sistema. Depois os arquivos associados aos descritores ***f*** e ***nf*** são fechados. Na linha 58 o texto "Executando o *nam*..." será impresso e, em seguida, na linha 59, o comando do *ns* ***exec*** forçará a execução do comando ***nam out.nam &*** pelo sistema operacional (não pelo *ns*), cujo efeito é a ativação do *nam* para animação da simulação. Observe que o *nam* é executado em "background" (veja o símbolo ***&*** no final da linha) e observe também a importância de fechar os arquivos de trace antes de executar comandos que os manipulem fora do ambiente do *ns*. Finalmente, o comando ***exit 0*** finaliza o *ns*.

Na última linha, o comando ***$***ns ***run*** inicia a simulação, causando a disparada dos eventos programados. Alguns trechos da animação do *nam* para a simulação descrita podem ser vistos nas figuras a seguir. Na Figura 2.9 podemos ver o instante t=0.36s, no qual o

tráfego CBR está em pleno fluxo. A Figura 2.10 mostra o tráfego simultâneo dos dados produzidos pelas aplicações ***cbr0*** e ***ftp***, no instante t=0.56s.

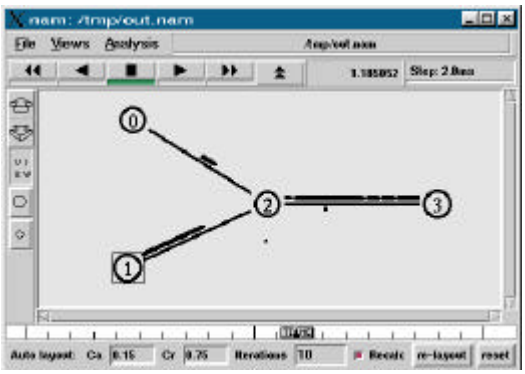


Figura 2.11 – A concorrência no enlace 2-3 causa perda de pacotes

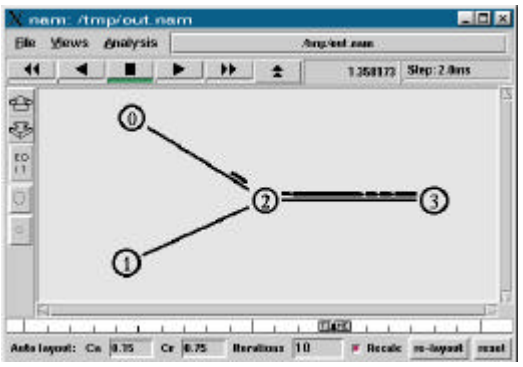


Figura 2.12 – Em t=1.35s o tráfego FTP é perdido e o CBR continua

A Figura 2.11, que mostra o instante t=1.18s, nos permite ver que a concorrência pelo enlace ***n2-n3*** forçou o descarte de alguns pacotes (mostrado como o pequeno ponto embaixo do nó ***n2***), que é um indicativo de que houve congestionamento nesse ponto da rede. A Figura 2.12 mostra o instante t=1.35s, já depois que os agentes ***tcp*** e ***sink*** foram bruscamente desanexados dos seus respectivos nós ***n1*** e ***n3***. Dizemos que foram "bruscamente" desanexados porque, normalmente, aciona-se o procedimento ***stop*** na aplicação para parar o fluxo de dados (exemplo: ***$***ns ***at 1.35 "$ftp stop"***). A Figura 2.12 mostra, portanto o fluxo CBR que é mantido até o final da simulação.

2.6.3. Exemplo 2 – Fluxos CBR competindo em fila SFQ

Este exemplo é parecido com o anterior, mas ambas a aplicações são CBR usando UDP e a fila do roteador que controla o enlace onde a competição ocorre usa uma política diferente de descarte de pacotes: SFQ - S*tochastic Fair Queue*, como veremos adiante.

```
1   Exemplo-2 – Adaptado de example2.tcl (Tutorial de Marc Greis [15])
2   set ns [new Simulator]
3
4   # Define cores para os fluxos de dados
5   $ns color 1 Blue ; $ns color 2 Red
6
7   # Abre arquivos de trace para o nam
8   set nf [open out.nam w]
9   $ns namtrace-all $nf
10
11  proc finish {} {
12          global ns nf
13          $ns flush-trace
14        close $nf
15          exec nam out.nam &
16          exit 0
17  }
18
19  # Cria quatro nós
20  set n0 [$ns node] ; set n1 [$ns node] ; set n2 [$ns node] ; set n3 [$ns node]
21
22  # Cria enlaces
```

```
23  $ns duplex-link $n0 $n2 1Mb 10ms DropTail
24  $ns duplex-link $n1 $n2 1Mb 10ms DropTail
25  $ns duplex-link $n3 $n2 1Mb 10ms SFQ
26
27  $ns duplex-link-op $n0 $n2 orient right-down
28  $ns duplex-link-op $n1 $n2 orient right-up
29  $ns duplex-link-op $n2 $n3 orient right
30
31  # Monitora a fila no enlace n2-n3
32  $ns duplex-link-op $n2 $n3 queuePos 0.45
33
34  # Cria agentes CBR e anexa aos nós
35  set cbr0 [new Application/Traffic/CBR]
36  set cbr1 [new Application/Traffic/CBR]
37  set udp0 [new Agent/UDP]
38  set udp1 [new Agent/UDP]
39  set null0 [new Agent/Null]
40  set null1 [new Agent/Null]
41  $cbr0 attach-agent $udp0
42  $cbr1 attach-agent $udp1
43  $ns attach-agent $n0 $udp0
44  $ns attach-agent $n1 $udp1
45  $ns attach-agent $n3 $null0
46  $ns attach-agent $n3 $null1
47  $ns connect $udp0 $null0
48  $ns connect $udp1 $null1
49
50  $cbr0 set packetSize_ 512
51  $cbr0 set interval_ 0.005
52  $cbr1 set packetSize_ 512
53  $cbr1 set interval_ 0.005
54
55  $udp0 set fid_ 1
56  $udp1 set fid_ 2
57
58  # Escalona eventos para os agents CBR
59  $ns at 0.1 "$cbr0 start" ; $ns at 0.5 "$cbr1 start"
60  $ns at 1.0 "$cbr1 stop"  ; $ns at 1.5 "$cbr0 stop"
61  $ns at 2.0 "finish"
62
63  # Inicia a simulação
64  $ns run
```

Neste exemplo, algumas novidades aparecem em relação aos exemplos anteriores. Na linha 5 temos dois comandos separados por ponto-e-vírgula, o que é permitido em Otcl. Os comandos associam números a cores, as quais poderão ser usadas para colorir os fluxos de dados. Na linha 25, com a atribuição de um tipo de fila SFQ ao enlace ***n2-n3***, ao contrário das filas do tipo DropTail dos enlaces ***n0-n2*** e ***n1-n2***. A sigla SFQ significa *Stochastic Fair Queue*, um tipo de fila que implementa um tratamento justo no descarte de pacotes dos vários fluxos de dados que passam pelo enlace. DropTail descarta pacotes indiscriminadamente.

As linhas 27-29 são instruções de orientação gráfica para o *nam* e não interferem na simulação em si. Elas indicam a orientação dos enlaces no momento que o *nam* for desenhar a topologia. A instrução ***right-down*** para o enlace ***n0-n2*** indica que a linha que representa o enlace deverá ser desenhada no sentido para a direita e para baixo. Outras instruções de

orientação para os enlaces podem combinar as palavras ***right***, ***left***, ***up*** e ***down*** (exemplo: ***left-down***, ***left***, ***left-up***). A linha 32 contém um comando que adiciona uma opção de monitoração da fila do enlace ***n2-n3***, indicando que o tamanho da fila e o descarte de pacotes sejam mostrados pelo *nam* durante a animação.

As linhas 34-48 criam agentes e os conectam aos nós, conforme já vimos em exemplos anteriores. As linhas 50-53 manipulam o tamanho do pacote em bytes e a freqüência com que cada pacote deve ser enviado por segundo pelas aplicações ***cbr0*** e ***cbr1***. O resultado será o envio de 200 pacotes de 512 bytes a cada segundo, partindo de cada uma das aplicações CBR. Isso significa que cada um produzirá 100Kbytes a cada segundo, ou 800Kbps, e que, portanto, o nó ***n2*** deverá receber uma demanda de 1.6Mbps (a soma do tráfego vindo de ***n0*** e ***n1***) para encaminhar para o nó ***n3*** através de um enlace com capacidade de apenas 1Mbps. Esta simples conta nos mostra que um congestionamento acontecerá no enlace ***n2-n3*** com a conseqüente perda (descarte) de pacotes.

As linhas 55-56 associam um número para cada fluxo de dados UDP (que por sua vez já estão associados aos fluxos CBR), de maneira que eles possam ser identificados pelas cores já definidas na linha 5 (azul para o fluxo 1 e vermelho para o fluxo 2). Os eventos são programados nas linhas 59-61 e a simulação inicia com o comando ***$***ns ***run*** da linha 64.

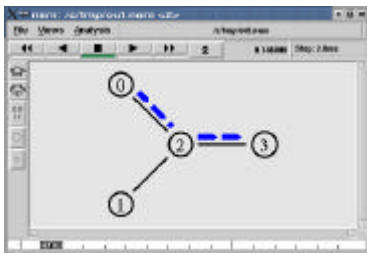


Figura 2.13 – Fluxo cbr0 iniciado

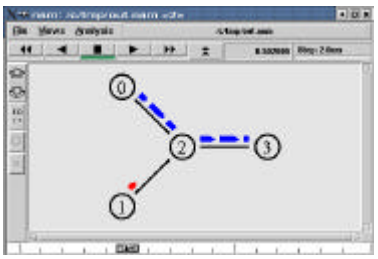


Figura 2.14 – Fluxo cbr1 iniciado

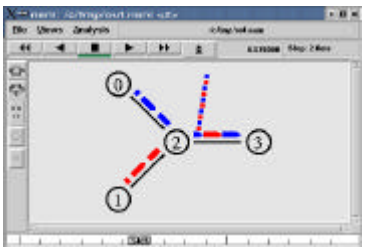


Figura 2.15 – Monitor na fila no link 2-3

Durante a animação podemos ver na Figura 2.13 o instante em que o fluxo ***cbr0*** está em andamento e o enlace ***n2-n3*** encaminha o tráfego sem congestionamento, uma vez que sua capacidade é suficiente para um dos fluxos. A Figura 2.14 mostra o instante em que o segundo fluxo se inicia e a Figura 2.15 mostra a monitoração da fila no instante em que os dois fluxos de dados começam a competir pelo enlace ***n2-n3***.

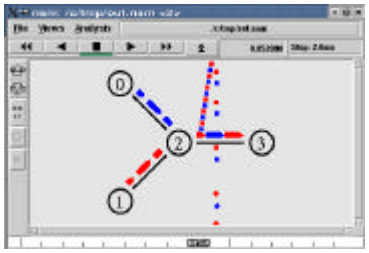


Figura 2.16 –Descarte "justo" na fila SFQ

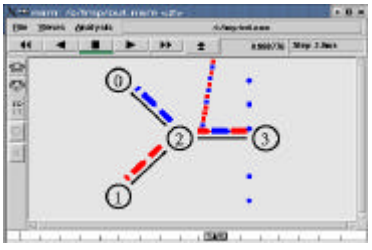


Figura 2.17 – Descarte "injusto" na fila DropTail

Figura 2.18 – Fim cbr1, com fluxo ainda na fila.

Quando a fila não suporta mais armazenar toda a demanda de dados na entrada do enlace, o nó ***n2*** começa a descartar os pacotes da fila. Entretanto, ao contrário de uma fila do tipo "DropTail", que faz um descarte aleatório, a fila SFQ tenta descartar os pacotes de

maneira proporcional à quantidade de pacotes de cada fluxo, já armazenados na fila. No nosso caso, como a quantidade de pacotes é a mesma para cada fluxo, o descarte é feito em quantidade igual e, portanto, justa.

A Figura 2.16 mostra o instante em que os pacotes são descartados (a figura original é colorida e os pacotes de cores diferentes podem ser vistos). A Figura 2.17 mostra como seria o descarte de pacotes em uma fila "DropTail" (apenas os pacotes azuis são descartados). A Figura 2.18 mostra o instante em que o fluxo ***cbr1*** é encerrado e a fila ainda continua a descartar pacotes por um período de tempo necessário à estabilização do congestionamento (relação da demanda com a capacidade). Em seguida, a fila irá diminuir de tamanho até zero, estabilizando o fluxo ***cbr0*** até que ele seja encerrado e a simulação termine.

2.6.4. Exemplo 3 – Controle de congestionamento no TCP

A idéia deste exemplo é ilustrar o funcionamento do controle de congestionamento do TCP, mostrando na prática o funcionamento da "janela de congestionamento".

```
1   # Exemplo-3 – Controle de congestionamento no TCP
2   set ns [new Simulator] ;# Cria o objeto "simulador"
3
4   set nf [open out.nam w]  ;# Abre o arquivo de trace do nam
5   $ns namtrace-all $nf
6
7   proc finish {} { ;# Define procedimento a ser executado no final
8           global ns nf
9           $ns flush-trace
10          close $nf
11          exec nam out.nam &
12          exit 0
13  }
14  set n0 [$ns node] ; set n1 [$ns node]  ; #Cria 4 nós
15  set n2 [$ns node] ; set n3 [$ns node]
16
17  $ns duplex-link $n0 $n2 1Mb 10ms DropTail;# Cria links entre os nós
18  $ns duplex-link $n1 $n2 1Mb 10ms DropTail
19  $ns duplex-link $n2 $n3 1Mb 10ms DropTail
20
21  $ns queue-limit $n2 $n3 10 ;# Limita fila no enlace 2-3 (10 pacotes)
22  $ns duplex-link-op $n2 $n3 queuePos 0.5 ;# monitora a fila do enlace 2-3
23
24  # Orientaçoes para o nam
25  $ns duplex-link-op $n1 $n2 orient up
26  $ns duplex-link-op $n0 $n2 orient right
27  $ns duplex-link-op $n2 $n3 orient right
28  Agent/TCP set nam_tracevar_ true ;# Habilita "trace" de var. TCP (nam)
29
30  set tcp0 [new Agent/TCP] ;# Cria agente TCP
31  $ns attach-agent $n0 $tcp0 ;# Anexa agente tcp0 ao nó 0
32  $tcp0 set packet_size_ 1500 ;# Tamanho máximo de pacote em bytes
33  set sink0 [new Agent/TCPSink] ;# Cria agente TCP consumidor
34  $ns attach-agent $n3 $sink0 ;# Anexa agente sink0 ao nó 3
35  $ns connect $tcp0 $sink0  ;# Connect TCP source with TCP sink
36  set ftp0 [$tcp0 attach-source FTP] ;# Cria aplic. FTP e anexa ao tcp0
37
38  set tcp1 [new Agent/TCP] ;# Cria agente TCP
39  $ns attach-agent $n1 $tcp1 ;# Anexa agente tcp1 ao nó 1
```

```
40  $tcp1 set packet_size_ 1500 ;# Tamanho máximo de pacote em bytes
41  set sink1 [new Agent/TCPSink] ;# Cria agente TCP consumidor
42  $ns attach-agent $n3 $sink1 ;# Anexa agente sink1 ao nó 3
43  $ns connect $tcp1 $sink1   ;# Connect TCP source with TCP sink
44  set ftp1 [$tcp1 attach-source FTP] ;# Cria aplic FTP e anexa ao tcp1
45
46  # Define cores para os fluxos de dados tcp0=azul, tcp1=vermelho
47  $ns color 1 Blue ; $ns color 2 Red
48  $tcp0 set fid_ 1 ; $tcp1 set fid_ 2
49
50  # Adiciona o rastreamento de variáveis
51  $ns add-agent-trace $tcp0 tcp0 ;# label "tcp0"
52  $ns add-agent-trace $tcp1 tcp1 ;# label "tcp1"
53  $ns monitor-agent-trace $tcp0 ;# (nam) monitorar variáveis do tcp0
54  $ns monitor-agent-trace $tcp1 ;# (nam) monitorar variáveis do tcp1
55  $tcp0 tracevar cwnd_ ;# rastreia a variável cwnd_ do tcp0
56  $tcp1 tracevar cwnd_ ;# rastreia a variável cwnd_ do tcp1
57
58  # Programa os eventos da simulação
59  $ns at 0.1 "$ftp0 start"
60  $ns at 1.0 "$ftp1 start"
61  $ns at 14.0 "$ftp1 stop"
62  $ns at 14.5 "$ftp0 stop"
63  $ns at 25.0 "finish"
64
65  # Executa a simulação
66  $ns run
```

Vamos antes discutir quais as novidades surgidas no código fonte deste exemplo em relação aos exemplos anteriores. Observando o código fonte, primeiro foram criados o objeto simulador principal, a rotina de finalização, os nós (***n0***,...,***n4***) e os enlaces entre eles (tudo isso entre as linhas 2 e 19). Na linha 21 temos um comando para limitar o tamanho da fila no enlace ***n2-n3*** em 10 pacotes, o que será suficiente para forçar um congestionamento no "roteador" ***n2*** quando houver fluxo contínuo vindo dos nós ***no*** e ***n1***. As linhas 24-27 fornecem instruções de posicionamento de arrumação dos nós e enlaces para o *nam*. Ok, isso não é mais novidade.

A linha 28 traz um comando novo que permite ao *nam* o monitoramento de variáveis internas dos agentes, neste caso o TCP. O que esse comando faz, na verdade, é avisar ao TCP para gerar informações dentro do arquivo de trace do *nam* sobre os valores de suas algumas variáveis internas, sempre que elas sofrerem mudança, ou seja, um monitoramento de variáveis. Dessa forma, o *nam* vai poder mostrar a dinâmica dos valores das variáveis sincronizada no tempo com os demais eventos animados. Quais variáveis monitorar serão indicadas adiante.

E para quê serve o monitoramento de variáveis internas? Bem, como sabemos, o *ns* implementa seus objetos (exemplo: agentes TCP) de maneira que eles se comportem como uma implementação em uma rede real. E o TCP, por exemplo, para funcionar de acordo com sua especificação mantém internamente algumas variáveis de controle, entre elas: RTT (tempo aproximado de ida e volta de um pacote no caminho), tempo de espera por um reconhecimento de pacote enviado (*acknowledgement*), tamanho da janela do receptor, tamanho da janela de congestionamento etc, que permitem ao TCP lidar com o envio de dados, mantendo controle de fluxo e controle de congestionamento. Algumas dessas variáveis

são dinâmicas e o acompanhamento da variação de seus valores nos permite entender como de fato o TCP lida com os eventos gerados pela rede.

As linhas 30-44 criam agentes, anexam-nos aos nós, conectam-nos uns aos outros, criam 2 aplicações FTP que vão ficar ligadas aos nós ***n0*** e ***n1***, enviando dados para o nó ***n3***. As linhas 46-48 associam cores aos fluxos, sendo azul para o ***tcp0/ftp0*** e vermelho para o ***tcp1/ftp1***.

As linhas 50-56 trazem novos comandos que fornecem informações sobre como fazer o monitoramento de variáveis. As linhas 51-52 fazem o *nam* adicionar um rótulo aos agentes indicados (***tcp0*** e ***tcp1***), que podem ser vistos na Figura 2.19 ao lado dos nós ***n0*** e ***n1***. Cada rótulo é desenhado acompanhado de uma uma linha vertical, abaixo da qual é desenhado um número (0 para ***tcp0*** e 1 para ***tcp1***). Os números indicam a caixa de monitoração de variáveis associada a cada agente. As caixas de monitoração ficam na janela do *nam* na parte de baixo do desenho da topologia (veja o espaço indicando "Monitors") e nela serão mostrados os valores de cada variável monitorada. As linhas 53-54 criam as caixas de monitoração para os agentes ***tcp0*** e ***tcp1***. Finalmente, as linhas 55-56 indicam qual a variável que desejamos monitorar para cada agente (a variável – "***cwnd_***" que representa a janela de congestionamento). Para monitorar outras variáveis é necessário consultar o manual do *ns* ou olhar o código fonte do agente (muitas vezes é suficiente consultar o arquivo ***<DIR_NS>ns/tcl/lib/ns_defaults.tcl***. As linhas 58-63 escalonam o início e fim dos fluxos de dados e a linha 66 inicia a simulação.

Antes de iniciar a discussão sobre o andamento da simulação, vamos explicar resumidamente como funciona o controle de congestionamento do TCP e esclarecer alguns conceitos ligados ao controle do fluxo de dados entre o emissor e o receptor.

O emissor pode transmitir até o valor resultante do mínimo entre a capacidade do receptor (conhecido como "janela do receptor") e a capacidade da rede (conhecido como "janela de congestionamento", que é mantida numa variável interna chamada **cwnd_**). Logo, a janela de congestionamento é o controle de fluxo imposto pela rede e a janela do receptor é o controle de fluxo imposto pela capacidade do receptor. Para simplificação, podemos aceitar que o receptor tem maior capacidade que a rede, o que é verdade pela limitação de 10 pacotes que fizemos na fila do roteador ***n2*** (veja linha 21). Isso nos permitirá pensar no controle de congestionamento baseado apenas nas indicações fornecidas pela rede.

O emissor inicia transmitindo 1 segmento[6] (***cwnd_***=1) e espera por um reconhecimento do receptor indicando que este foi recebido (ACK). A cada ACK recebido, a janela de congestionamento do emissor é incrementada em 1. Quando o primeiro pacote for reconhecido, teremos ***cwnd_***=2, indicando que é permitido enviar até 2 segmentos. Quando esses 2 segmentos forem reconhecidos (ACK), a janela é aumentada em 2 e vai para ***cwnd_***=4, permitindo ao receptor enviar até 4 segmentos. Depois de reconhecidos os últimos 4 segmentos enviados, teremos ***cwnd_***=8 e assim por diante. Isso dá um crescimento

[6] A unidade de dados tratada pelo protocolo TCP é chamada de "segmento". Entretanto, é perfeitamente aceitável pensar em um segmento TCP como um pacote IP a trafegar na rede.

exponencial[7] para a janela de congestionamento e, quanto mais ela crescer, mais tráfego será injetado na rede pelo emissor.

Em algum momento quando a capacidade da rede for alcançada, algum roteador intermediário começará a descartar pacotes, em resposta a sua incapacidade de lidar com o tráfego oferecido. Isso servirá de aviso para o emissor reduzir sua janela de congestionamento, diminuindo assim a sua taxa de envio. Na prática, a reação do TCP a uma perda de pacote é a redução drástica da janela de congestionamento para 1 segmento[8], iniciando novamente o processo de crescimento da janela a cada ACK recebido.

E como o TCP sabe que um pacote foi descartado? Ele usa um mecanismo simples. Ele aciona um temporizador[9] (timer) para cada pacote enviado e, se o pacote não for reconhecido (ACK) antes que o tempo expire, ele assume que o pacote foi perdido e deve ser retransmitido. Detalhes sobre os mecanismos de controle de congestionamento do protocolo TCP podem ser encontrados em [2].

Inicialmente a janela de congestionamento do ***tcp0*** tem o valor 1 (***cwnd_=1***). Ele segue o procedimento de enviar pacotes e aguardar o ACK correspondente, aumentando cada vez mais o volume de transmissão e incrementando ***cwnd_*** a cada ACK recebido. A Figura 2.19 mostra os pacotes sendo transmitidos (faixa sobre as linhas que representam os enlaces) e os ACKs sendo retornados (pequenos pontos abaixo das linhas dos enlaces). Observe ainda que a caixa de monitoração "**[0] Agent tcp0**" mostra que o valor da variável ***cwnd_*** neste momento é 17 (tempo de simulação=0.38s).

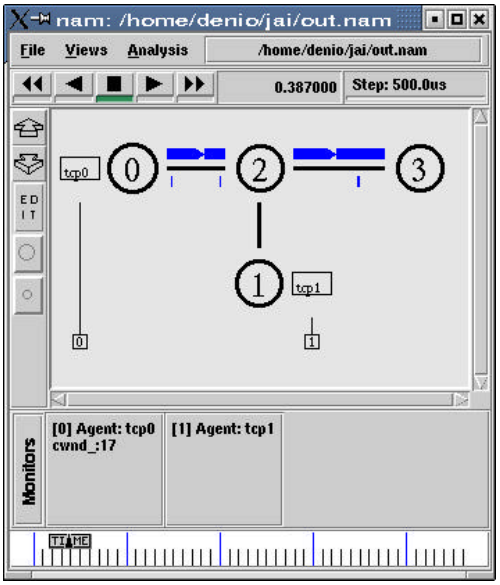


Figura 2.19 – tcp0 transmite e recebe

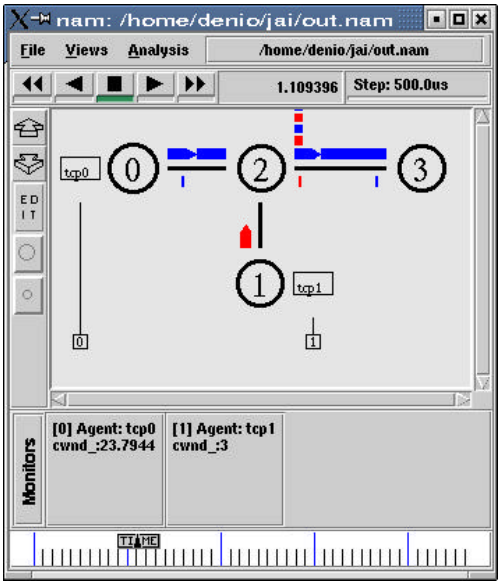


Figura 2.20 – tcp1 inicia transmissão

[7] Na verdade, a janela de congestionamento não cresce até o infinito. Existe um limitador, chamado *slow start threshold* (ssthresh) a partir do qual o crescimento é linear.

[8] O comportamento do TCP frente a uma perda de pacote varia dependendo da sua versão, mas a idéia básica é reduzir a janela de congestionamento para diminuir a taxa de envio e evitar o congestionamento.

[9] O tempo limite de retransmissão (RTO – *Retransmission Timeout*) é calculado dinamicamente em função do tempo aproximado de ida e volta de um pacote (RTT – *Round Trip Time*).

o ACK dos pacotes enviados e concorre pelo enlace 2-3

A Figura 2.20 mostra a situação no tempo=1.10s, no qual o ***tcp1*** já iniciou sua transmissão. Nesse momento ambos os emissores competem pelo enlace ***n2-n3*** o que força um enfileiramento de pacotes no roteador ***n2***. Observe que o valor de ***cwnd_*** do ***tcp0*** continua crescendo e agora tem o valor 23, enquanto que o valor de ***cwnd_*** do ***tcp1*** ainda está em 3. Isso significa que ***tcp0*** está mandando muito mais pacotes do que ***tcp1***.

A excessiva demanda de tráfego ofertada para o enlace ***n2-n3*** causa o descarte de pacotes na fila do roteador ***n2***, cujo tamanho está limitado a 10 pacotes. Isso pode ser observado na Figura 2.21 que mostra a situação no tempo=1.29s. Ainda na Figura 2.21, observamos que os valores de ***cwnd_*** são 24 e 9 para ***tcp0*** e ***tcp1***, respectivamente. Ou seja, ambos continuam crescendo, uma vez que ainda não perceberam o congestionamento que já se iniciou.

A Figura 2.22 (tempo=1.44s) mostra que o ***tcp0*** já percebeu a perda de pacotes e reduziu drasticamente sua janela de congestionamento para 1, fazendo com que seu fluxo de dados praticamente pare. Ele agora vai novamente tentar aumentar a taxa de transmissão e experimentar a capacidade da rede até haver outro sinal indicando para ele reduzir. Enquanto isso, o fluxo do ***tcp1*** está dominante e a todo vapor e, como ainda há pacotes do fluxo ***tcp0*** esperando na fila (tomando espaço), alguns pacotes do fluxo ***tcp1*** já começam também a serem descartados.

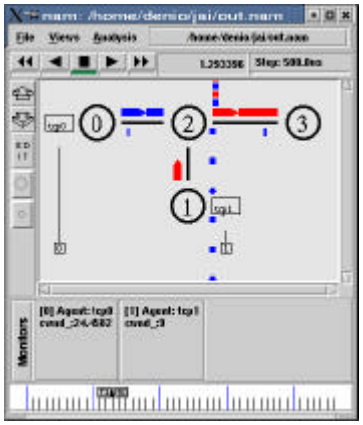


Figura 2.21 – Pacotes descartados

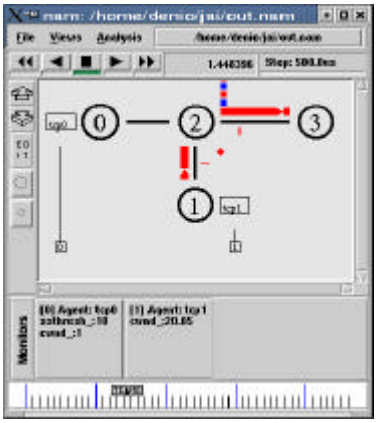


Figura 2.22 – Redução de cwnd_ para 1 (tcp0)

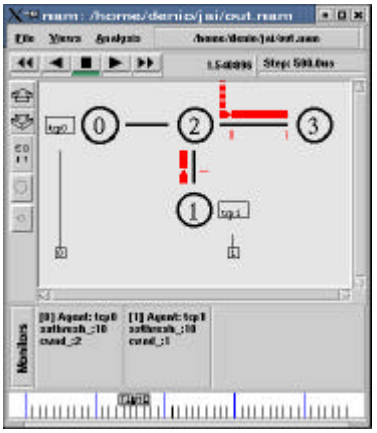


Figura 2.23 – Redução de cwnd_ para 1 (tcp1)

A Figura 2.23 mostra que algum tempo depois (tempo=1.54s) o ***tcp1*** percebeu o descarte de pacotes e também reduziu sua taxa de transmissão, atualizando ***cwnd_*** para 1. Nesse momento, o ***tcp0*** ainda está se recuperando e ainda tem ***cwnd_=2***.

Esse comportamento se dará até o final da simulação, com os agentes ***tcp0*** e ***tcp1*** competindo pelo envio de dados, aumentando sua taxa de transmissão e depois tendo que reduzi-la em face de um congestionamento na rede. Esse comportamento dinâmico é necessário porque a rede é dinâmica, ou seja, novas aplicações podem entrar ou sair da competição e as demais aplicações devem se ajustar à situação atual da rede.

2.7. Estudo de caso: QoS na Internet

Esta seção apresenta um estudo de caso, visando auxiliar o aluno a iniciar pesquisas sobre a Internet através de simulação. Esse estudo é uma versão muito simples e preliminar do que poderia ser considerada uma avaliação que apresentasse contribuições científicas. As conclusões apresentadas também já são do conhecimento da comunidade científica.

2.7.1. Objetivo e contexto de simulação

O enfoque do estudo desse caso é na aplicação de diferentes tecnologias para Qualidade de Serviço (QoS) na Internet. O objetivo é executar uma comparação do desempenho de uma aplicação multimídia utilizando o serviço de melhor esforço tradicional da Internet e as tecnologias IntServ e DiffServ. Abaixo são apresentados os conceitos básicos de QoS com um nível mínimo de detalhes.

2.7.1.1. QoS na Internet

Na literatura podem ser encontradas várias definições de Qualidade de Serviço. Pela definição adotada pelos autores, QoS tem dois aspectos [22]:

- O desempenho de uma rede relativo às necessidades das aplicações.
- O conjunto de tecnologias que possibilita à rede oferecer garantias de desempenho.

2.7.1.2. O serviço de melhor esforço

A Internet atual utiliza um modelo de serviço de melhor esforço, que significa que todos os usuários e aplicações têm o mesmo tratamento nos roteadores no caminho entre origem e destino dos pacotes. Em situações de congestionamento, roteadores guardam pacotes em filas na ordem estrita de chegada (FIFO). Quando a capacidade da fila se esgota, os pacotes são simplesmente descartados. Esse modelo apresenta uma grande simplicidade e robustez, um dos motivos do sucesso da Internet, embora não permita o desenvolvimento de aplicações avançadas e a diferenciação de serviços entre usuários.

2.7.1.3. Métricas de QoS

A garantia de transmissão para obtenção de QoS na Internet pode ser expressa como a combinação de alguns dos seguintes parâmetros:

- **Atraso**: É o tempo necessário para um pacote percorrer a rede, do momento em que é transmitido pelo emissor até ser recebido pelo receptor (fim-a-fim).
- **Variação do atraso**: É a variação no atraso fim-a-fim, também chamada de *jitter*.
- **Vazão**: A taxa máxima que alguma aplicação ou protocolo consegue manter entre dois pontos em uma rede.
- **Confiabilidade**: está relacionada à taxa de perda (descarte) de pacotes pela rede.

2.7.1.4. Propostas para QoS na Internet

A IETF [17] tem ultimamente dedicado uma grande atenção ao tema "QoS na Internet" e vem apresentando algumas propostas [35], principalmente:

- **IntServ**: o modelo de *Serviços Integrados*, que utiliza o protocolo RSVP (*Resource Reservation Protocol*) para reservar recursos específicos para cada fluxo. Acredita-se que IntServ tem problemas de escalabilidade e não vá funcionar bem na Internet.

- **DiffServ**: o modelo de *Serviços Diferenciados* que utiliza o conceito de agregação de fluxos para obter escalabilidade. Pacotes são agrupados em PHBs, que recebem algum tipo de tratamento diferenciado nos roteadores.

2.7.2. Plano de simulação

Esta seção descreve as decisões tomadas com relação ao planejamento da simulação. Como topologia de simulação (Figura 2.24), foi utilizada uma simplificação da espinha dorsal da RNP2 [32], representada pelos PoPs (pontos de presença) de Recife (PE), Florianópolis (SC), Rio de Janeiro (RJ), São Paulo (SP), Belo Horizonte (MG) e Brasília (DF). A simulação compara o desempenho de tráfego de voz entre SC e PE, com ou sem a utilização de tecnologias de QoS. Embora o tráfego de interesse não seja encaminhado por MG e DF, esses PoPs são importantes para aumentar a realidade da simulação, contribuindo para aumentar as rajadas nos PoPs.

As métricas utilizadas para comparar as tecnologias de QoS foram **vazão** e **atraso**. O objetivo é mostrar que muitas vezes, embora a vazão seja garantida para uma determinada aplicação, como voz, o atraso excessivo pode inviabilizar a sua utilização.

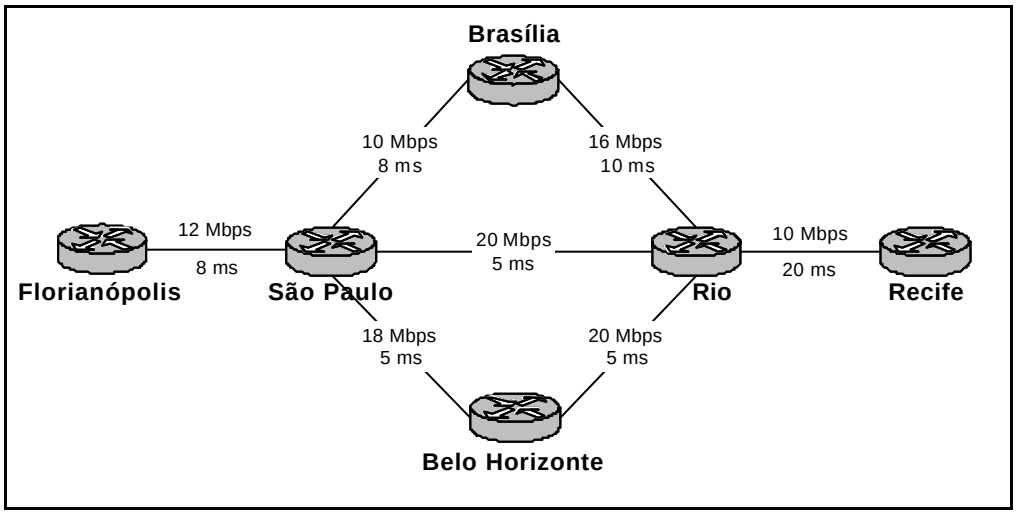


Figura 2.24 – Topologia de simulação

Os parâmetros de simulação estão listados abaixo:

- **Roteadores**: Cada PoP foi representado através de um roteador. Os enlaces entre RJ, SP, MG e DF estão de acordo com a topologia da RNP2. PE está ligado apenas a RJ enquanto que SC apenas a SP (na realidade, tanto PE quanto SC estão conectados a RJ e SP). Isso foi feito para que o tráfego entre SC e PE tivesse que ser encaminhado obrigatoriamente entre o RJ e SP, criando um gargalo.
- **Enlaces**: A capacidade de cada enlace representa a situação real da RNP2 em abril de 2002. O atraso foi configurado levando em consideração a distância física, mais algum atraso adicional representado por outros equipamentos nos PoPs e na Embratel (operadora que fornece os enlaces).
- **Modelo de tráfego**: Foi simulado tráfego de voz (tráfego principal) e de dados (tráfego de retaguarda).
    - Para representar o tráfego de voz foram utilizadas fontes CBR, porque facilita a compreensão dos resultados em forma de gráfico, principalmente com relação à

vazão (que é constante). As quantidades de fontes CBR foram: 20 fontes entre SC e PE; 10 entre SC e DF; 10 entre SC e MG; 10 entre DF e PE; 10 entre MG e PE; 10 entre SP e RJ. Cada fonte CBR transmitiu pacotes de 100 bytes a uma taxa de 64 Kbps.

- Para representar o tráfego de dados foram utilizadas fontes FTP,simulando transferência de arquivos. Uma simulação mais complexa deveria certamente incluir fontes HTTP. A quantidade de fontes foi variada na simulação (fator). Cada fonte FTP transmitiu pacotes de 1500 bytes. Não se configura taxa de transmissão para fontes FTP. Como FTP usa o protocolo TCP, ele adapta a sua taxa de transmissão às condições da rede (ou seja, o quanto puder).

- **Técnica de QoS**: Foram comparadas as técnicas de melhor esforço (padrão na Internet), IntServ (serviço de carga controlada) e DiffServ (PHB EF).
- **Tempo de simulação**: Cada execução da simulação teve a duração de 10 segundos. Esse tempo foi suficiente para obter resultados expressivos, uma vez que foram realizadas simulações com tempos maiores e os resultados não apresentaram variações significativas. Além disso, dada a modelagem das fontes, pode-se considerar que 10 segundos é um período suficientemente generoso para observar as métricas de interesse. As fontes CBR e FTP iniciam a transmissão em algum momento entre os tempos 0 e 1 segundo, escolhido aleatoriamente de acordo com uma distribuição uniforme.
- **Replicações**: Cada execução da simulação foi replicada 100 vezes para garantir confiabilidade estatística aos resultados e permitir trabalhar corretamente com intervalos de confiança. Esse número foi escolhido porque testes com um número maior de replicações não mostraram nenhuma variação significativa nos resultados.
- **Fatores e níveis**: Dois parâmetros foram variados entre simulações diferentes, sendo assim considerados fatores. A carga da rede, representada pela quantidade de fontes FTP de retaguarda assumiu os níveis 0, 5 e 50, para ilustrar situações com a rede super-provisionada, com leve carga e fortemente carregada, respectivamente. As tecnologias de QoS mencionadas acima também foram consideradas como fatores.

2.7.3. Execução da simulação e coleta de dados

Para esse estudo de caso foi utilizado o simulador *ns*, na sua versão *ns*-2.1*b*8*a*[10]. Foram utilizadas as funcionalidades oferecidas na distribuição padrão do *ns*, com algumas alterações locais. O serviço de melhor esforço não necessitou de nenhuma configuração adicional. O PHB EF do DiffServ foi simulado com filas WRR (*Weighted Round-Robin*, ou fila circular com pesos) configuradas através do módulo de CBQ (*Class-Based Queuing*), disponível no *ns*.

Também foi realizada a substituição do gerador de números aleatórios padrão do *ns*, o Park-Miller [26], pelo gerador de Mersenne-Twister [25], que tem um período de $2^{19937}-1$, o que implica em uma menor probabilidade de geração de seqüências de números repetidos (o gerador Park-Miller tem um período de $2^{31}-2$). Além disso ele é mais rápido e passou nos testes mais rígidos, com resultados superiores aos outros geradores conhecidos.

[10] A versão *ns*-2.1*b*9 foi lançada enquanto este estudo estava sendo realizado.

A coleta de informações de vazão foi realizada através do componente LossMonitor, que guarda o número de bytes recebidos por um destino de dados[11]. Foi coletada uma amostra da vazão a cada período de 0,5 segundos e calculada a média de cada replicação a cada 10 segundos. Para a coleta de informações relativas ao atraso foi utilizado um componente especialmente desenvolvido pelos autores para esse fim, chamado PktStats[12]. Esse componente é conectado entre o enlace e o nó de destino. Ele recebe cada pacote, calcula o atraso e variação do atraso e grava em um arquivo. Foi coletada uma amostra de atraso para cada pacote e calculada a média de cada replicação a cada 10 segundos. Para calcular o atraso é necessário incluir um campo no cabeçalho dos pacotes, para registrar o momento em que eles foram transmitidos. Uma alternativa para calcular o atraso é fazer um pós-processamento do arquivo de *trace* do *nam*.

As simulações foram executadas em uma máquina com CPU AMD Athlon de 1,3 GHz, com 512 MB de memória, executando sistema operacional Linux. Foram realizados nove conjuntos (três tecnologias de QoS vezes três níveis de carga de retaguarda) de 100 replicações de simulações de 10 segundos. Cada conjunto foi executado em poucos minutos de tempo de relógio.

2.7.4. Apresentação e análise de resultados

Esta seção apresenta os resultados da comparação das tecnologias de QoS utilizando simulação. Todos os resultados apresentados se referem apenas a uma fonte CBR de 64 Kbps (chamada de CBR1) entre SC e PE. O desempenho das demais fontes não foi avaliado. Um estudo mais detalhado poderia avaliar todas as fontes CBR e analisar a média de todas elas. No entanto, para os objetivos deste estudo, a análise de apenas uma fonte já produz conclusões significativas, com baixa complexidade.

Os resultados se referem à média (de vazão e atraso) das médias calculadas para cada uma das 100 replicações. Foram utilizados intervalos de confiança assintóticos ao nível de 99,9%. Para os fins desse projeto será considerado que o tempo entre as coletas (10 segundos) é suficiente para prover independência entre elas.

2.7.4.1. Vazão

A Figura 2.25 mostra um gráfico comparativo da vazão para serviço de melhor esforço, IntServ e DiffServ, com carga de retaguarda variando de zero, 5 e 50 fontes FTP. Pode-se ver que quando a rede está totalmente super-provisionada (nenhuma fonte de retaguarda competindo pelos enlaces), as três técnicas são capazes de encaminhar todos os pacotes gerando em todas os momentos uma taxa de 64 Kbps. IntServ e DiffServ conseguem manter a mesma vazão com a rede levemente carregada (5 fontes FTP) e congestionada (50 fontes FTP).

[11] Ver exemplo VIII do tutorial de Marc Greis [15].

[12] O componente PktStats e os *scripts* de simulação podem ser obtidos diretamente com os autores através dos seus endereços eletrônicos.

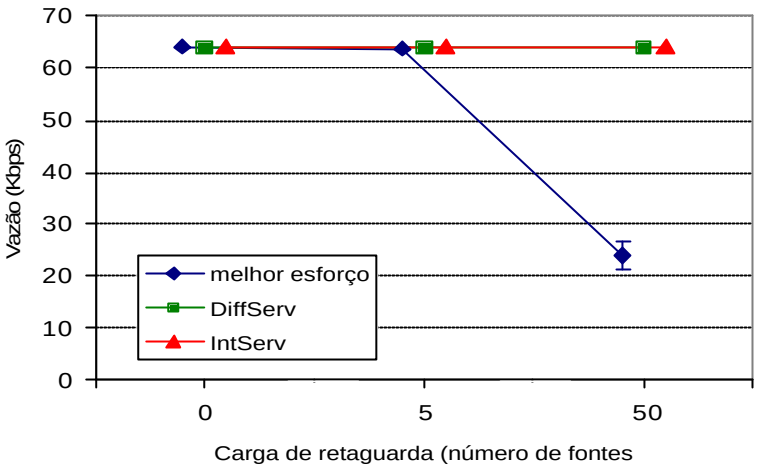


Figura 2.25 – Comparação de tecnologias de QoS através da vazão

Já o serviço de melhor esforço, a partir de 5 fontes FTP começa a dar sinais de que os pacotes CBR estão enfrentando problemas na rede, embora com 5 fontes FTP ele ainda consiga manter a vazão média de 64 Kbps para a fonte CBR. Com 50 fontes FTP, o congestionamento na rede torna-se muito grande e como as fontes CBR (que usam protocolo UDP) não diminuem a taxa de transmissão (como o protocolo TCP), a taxa média sustentada fica em aproximadamente 24 Kbps, ou seja, menos da metade. Em geral, as médias das repetições apresentam poucas variações, por isso os intervalos de confiança são muito pequenos, como pode ser observado na Figura 2.25.

Pode-se concluir que como as curvas e os intervalos de confiança se sobrepõem para um número de zero e 5 fontes FTP, não existe diferença entre o desempenho das três técnicas avaliadas. Com 50 fontes FTP, não existe diferença entre IntServ e IntServ, mas ambos tem um desempenho muito melhor que o serviço de melhor esforço. Como foram feitas 100 replicações e calculados os intervalos de confiança, esse tipo de conclusão tem um peso maior, pois apresenta maior segurança estatística.

Resultados de vazão para uma simulação de 60 segundos, com amostras coletadas em períodos de 0,5 segundos são mostrados na Figura 2.26. O comportamento da vazão para zero, 5 e 50 fontes FTP fica mais claro através desse gráfico. Embora com 5 fontes haja variações na vazão, a média de 10 segundos permanece constante em 64 Kbps. No caso de 50 fontes FTP, as variações são muito grandes, de 0 a aproximadamente 96 Kbps, a média permanece baixa e se mantém constante do decorrer da simulação (ou seja, as amostras com alta e baixa vazão se alternam constantemente no tempo, caracterizando um processo estacionário).

Um detalhe interessante mostrado na Figura 2.26 é que embora a fonte CBR1 esteja transmitindo a 64 Kbps, o receptor chega a observar até 96 Kbps (ou seja, a taxa recebida é maior que a enviada). Isso ocorre devido ao acúmulo de pacotes dessa fonte em filas dos roteadores, que posteriormente são liberados em seguida, a uma taxa maior do que entraram na fila. As séries temporais para IntServ e DiffServ não são mostradas, pois permanecem constantes em 64 Kbps.

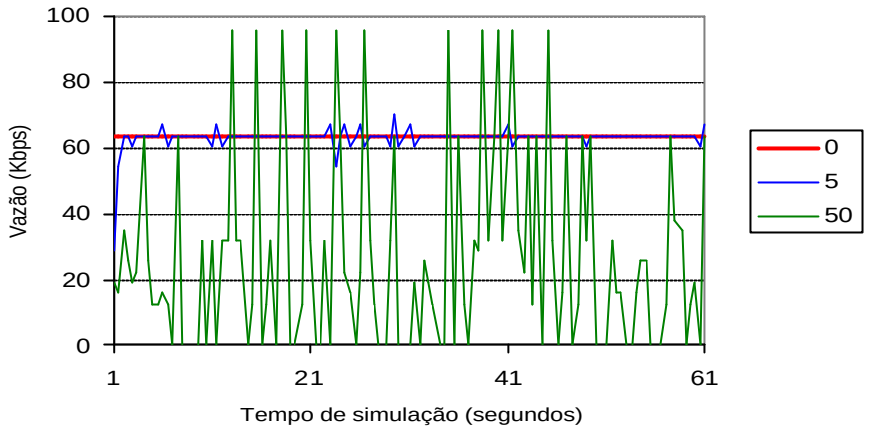


Figura 2.26 - Vazão com melhor esforço (série temporal)

2.7.4.2. Atraso

A Figura 2.27 apresenta uma comparação do atraso. Com a rede super-provisionada (nenhuma fonte FTP), as três técnicas conseguem encaminhar os pacotes com o mínimo atraso possível, de aproximadamente 34 segundos (ou seja, o atraso dos enlaces mais o tempo que um pacote inteiro leva para transitar por cada enlace). IntServ apresentou o melhor resultado, pois conseguiu manter o atraso praticamente constante com o aumento da carga (a variação do atraso para 5 e 50 fontes FTP foi muito pequena). Com DiffServ, para 5 e 50 fontes FTP, houve um aumento médio do atraso para aproximadamente 43ms e 44ms respectivamente. Isso significa que DiffServ não consegue manter as garantias individuais de atraso para todos os fluxos com o mecanismo de escalonamento de filas utilizado (WRR). No entanto, a diferença não é muito significativa e DiffServ mostrou-se escalável, ou seja, com o aumento do número de fontes o atraso não sofreu um impacto significativo.

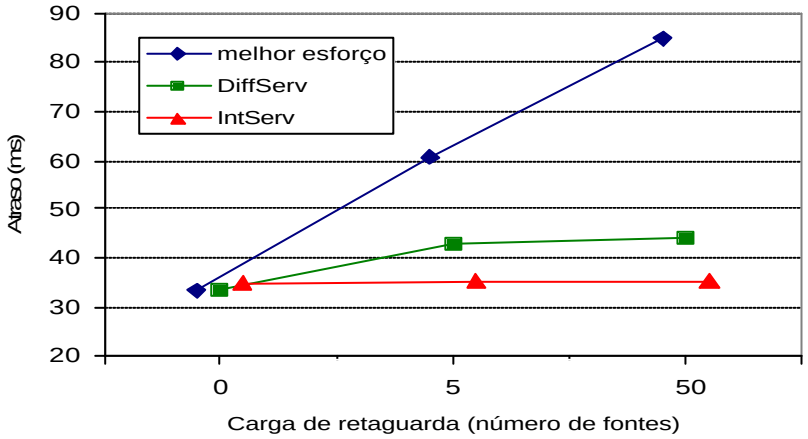


Figura 2.27 – Comparação de tecnologias de QoS através do atraso

Já para o serviço de melhor esforço, a rede levemente carregada foi suficiente para gerar um grande aumento no atraso (cerca de 100%). Embora com 5 fontes FTP a vazão tenha se mantido constante em 64 Kbps (Figura 2.25), o atraso pode ter um impacto negativo na aplicação de voz. Isso significa que com esse nível de carga, a rede não está descartando

pacotes, mas as filas dos roteadores estão com um tamanho médio alto e os pacotes estão permanecendo muito tempo nessas filas. Uma continuação desse estudo poderia avaliar a taxa de perda de pacotes para cada situação, além do tamanho médio das filas. Outro tópico interessante seria avaliar o desempenho da vazão e do atraso para as fontes de background quando as fontes CBR recebem um tratamento prioritário (com IntServ e DiffServ).

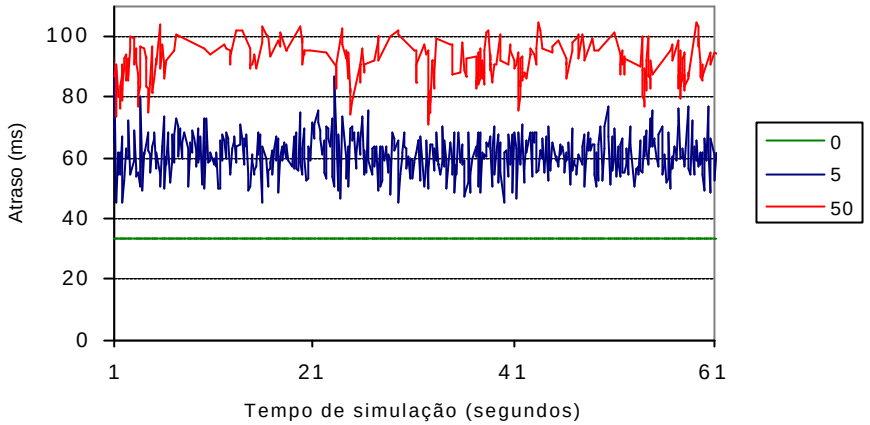


Figura 2.28 - Atraso com melhor esforço (série temporal)

A Figura 2.27 mostra que realmente existe uma diferença entre o comportamento do serviço de melhor esforço, IntServ e DiffServ com relação ao atraso.Os intervalos de confiança ficaram muitos pequenos (no máximo 1,35 segundos para melhor esforço com 50 fontes FTP) e não podem ser visualizados no gráfico. Como as curvas não se sobrepõem, pode-se concluir com segurança estatística que existe uma diferença significativa entre elas.

Na Figura 2.28 podem ser vistas amostras do atraso para uma simulação de 60 segundos para o serviço de melhor esforço. O gráfico mostra que existe uma diferença visual entre o atraso para as cargas da rede e que a variação é contínua no tempo, explicando o fato das médias das replicações de 10 segundos serem muito semelhantes umas as outras e gerando desvios padrões pequenos (isso também explica os intervalos de confiança pequenos).

2.8. Conclusão

A Internet é formada por uma grande diversidade de redes interconectadas pelo protocolo IP. Compreender o seu funcionamento, prever o seu crescimento futuro e propor novos protocolos e mecanismos são atividades complexas, mas que podem ser auxiliadas em grande parte com a realização de simulações adequadamente conduzidas. Simulação é uma técnica que pode ser utilizada tanto para obter resultados de pesquisa sobre a Internet quanto para auxiliar a compreensão de aspectos relacionados aos seus protocolos e mecanismos internos.

Esperamos que, através desse trabalho, o aluno tenha conseguido adquirir uma visão razoavelmente clara dos problemas, dificuldades e desafios envolvidos com simulação da Internet, bem como de técnicas, procedimentos e dicas que podem auxiliá-lo nessa tarefa. No entanto, o mais importante para melhorar a compreensão dos tópicos abordados é iniciar a prática de simulação. O estudo das referências bibliográficas citadas também é importante para o aprendizado dos iniciantes, como tem sido para os autores.

2.9. Referências Bibliográficas

[1] Advanced, "Surveyor", http://www.advanced.org/surveyor.

[2] Allman, M, Paxson, V. & Stevens W., TCP Congestion Control, RFC 2581, Abril 1999.

[3] Bagrodia, R., et al., "PARSEC: A Parallel Simulation Environment for Complex Systems", IEEE Computer, Outubro 1998.

[4] Bajaj, S., Breslau, L. & Shenker, S., "Uniform versus Priority Dropping for Layered Video", ACM SIGCOMM'98, Setembro 1998.

[5] Barford, P. & Crovella, M., "Generating Representative Web Workloads for Network and Server Performance Evaluation", ACM SIGMETRICS '98, Novembro 1998.

[6] CAIDA, http://www.caida.org.

[7] Calvert, K. L. Doar, M. B., Zegura, E. W., "Modeling Internet Topology", IEEE Communications Magazine, Junho 1997.

[8] Campos, M. A. Silva, J., L., C. & Cunha, P. R. F., "Modelagem Estocástica de Tráfego em Redes de Alta Velocidade", Jornada de Atualização em Informática (JAI 2001), agosto 2001.

[9] Cowie, J. H., Nicol, D. M., Ogielski, A. T., "Modeling the Global Internet", Computing in Science & Engineering, Janeiro 1999.

[10] Daze Networks, "Internet Health Report", http://www.ihr.daze.net.

[11] Doar, M. B., "A Better Model for Generating Test Networks", IEEE GLOBECOM, Novembro 1996.

[12] Estrin, D., "Network Visualization with Nam, the VINT Network Animator", IEEE Computer, Novembro 2000.

[13] Fall, K. & Varadhan, K., "The *ns* manual", The VINT Project , Fevereiro 2002.

[14] Floyd, S. & Paxson, V., "Difficulties in Simulating the Internet", IEEE/ACM Transactions on Networking, Fevereiro 2001.

[15] Greis, M. & VINT Project, "Tutorial for the Network Simulator 'ns'", http://www.isi.edu/nsnam/ns/tutorial/index.html.

[16] Herrscher, D., Leonhardi, A., & Rothermel, K., "Modeling Computer Networks for Emulation", PDPTA'02, Las Vegas, Junho 2002.

[17] IETF, "Internet Engineering Task Force", http://www.ietf.org.

[18] ITR, "Internet Traffic Report", http://www.internettrafficreport.com.

[19] Izquierdo, M. R. & Reeves, D. R., "A Survey of Statistical Source Models for Variable-bit-rate Compressed Video", Multimedia Systems, Agosto 1999.

[20] Jain, R., "The Art of Computer Systems Performance Analysis", Wiley, 1991.

[21] Jin, C., Chen Q. & Jamin, S., "Inet: Internet Topology Generator", Technical ReportSE-TR-433-00, Universidade de Michigan, 2000.

[22] Kamienski, C.A. & Sadok, D., "Qualidade de Serviço na Internet", minicurso, 18o SBRC, Belo Horizonte/MG, Maio 2000.

[23] Keshav, S., “REAL 5.0 Overview”, http://www.cs.cornell.edu/skeshav/real/overview.html Agosto 1997.

[24] Magoni, D. & Pansiot, J. J., “Analysis of the Autonomous System Network Topology”, ACM Computer Communication Review, Outubro 2001.

[25] Marsaglia, G., “A Current View of Random Number Generators”, 16th Symposium on the Interface Computer Science and Statistics, 1984, Elsevier Press.

[26] Miller, K. W., "Random Number Generators: Good Ones are Hard to Find," Communications of the ACM, Outubro 1988.

[27] Network Simulator (versão 2.1b8a), http://www.isi.edu/nsnam/ns/, Agosto 2001.

[28] *Ns*, “Using *ns* and *nam* in Education “, http://www.isi.edu/nsnam/ns/edu/index.html, 2002.

[29] OPNET Technologies, “OPNET Modeler”, www.opnet.com, 2002.

[30] Paxson, V. & Floyd, S., “Why We Don’t Know How To Simulate The Internet”, Winter Simulation Conference, Atlanta, GA, 1997.

[31] RIPE, “RIPE NCC Test Traffic Measurements”, http://www.ripe.net/ripencc/mem-services/ttm/index.html.

[32] RNP, “RNP2: Infra-estrutura Internet2 para o Brasil”, http://www.rnp.br/rnp2.

[33] Telcordia Netsizer, “Internet Growth Forecasting Tool”, 2002, http://www.netsizer.com.

[34] Thomson, K., Miller, G. J. & Wilder, R., “Wide-Area Internet Traffic Patterns and Characteristics”, IEEE Network, Novembro 1997.

[35] Xiao, X. & Ni, L. M., “Internet QoS: A Big Picture”, IEEE Network, Março/Abril 1999.