UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIA E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIAS DA COMPUTAÇÃO
MESTRADO ACADÊMICO EM CIÊNCIAS DA COMPUTAÇÃO

ACÉLIO SOUSA CARVALHO

AVALIANDO O DESEMPENHO DO NETWORK SIMULATOR 3 NA SIMULAÇÃO DE GRANDES REDES PEER-TO-PEER UTILIZANDO PROTOCOLO BITTORRENT

FORTALEZA
2010

ACÉLIO SOUSA CARVALHO

AVALIANDO O DESEMPENHO DO NETWORK SIMULATOR 3 NA SIMULAÇÃO DE GRANDES REDES PEER-TO-PEER UTILIZANDO PROTOCOLO BITTORRENT

Dissertação submetida à Coordenação do Curso de Mestrado em Ciências da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciências da Computação.

Orientação: Prof. Dr. Marcial Porto Fernandez

FORTALEZA
2010

ACÉLIO SOUSA CARVALHO

AVALIANDO O DESEMPENHO DO NETWORK SIMULATOR 3 NA SIMULAÇÃO DE GRANDES REDES PEER-TO-PEER UTILIZANDO PROTOCOLO BITTORRENT

Dissertação submetida à Coordenação do Curso de Mestrado em Ciências da Computação em __/__/____ da Universidade Estadual do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciências da Computação.

Aprovada em __/__/_____

BANCA EXAMINADORA

_____

Prof. Dr. Marcial Porto Fernandez (Orientador)
Universidade Estadual do Ceará

_____

Profa. Dra. Cláudia Maria Fernandes Araújo Ribeiro
Universidade do Estado do Rio Grande do Norte

_____

Prof. Pós-Dr. Miguel Franklin de Castro
Universidade Federal do Ceará

_____

Prof. Pós-Dr. Joaquim Celestino Júnior
Universidade Estadual do Ceará

AGRADECIMENTOS

Ao meu Deus, que diante de diversas dificuldades surgidas, me deu forças para continuar essa caminhada. À minha família, pelo apoio. Ao meu orientador Marcial, que com paciência, imensa colaboração, idéias e amizade, foi fundamental para que eu enxergasse meios de direcionar a pesquisa e concluir este trabalho. Ao professor André Cardoso, que foi grande responsável pelo incentivo inicial das pesquisas. Aos demais professores participantes da banca de qualificação, Joaquim Celestino e Jorge Luiz, que através de suas sugestões e conversas nos corredores, permitiram a evolução deste trabalho, e aos demais professores durante o curso, que me proporcionaram, com esforço, ampliar meus conhecimentos na área.

”...pra se perder no abismo que é pensar e sentir.”

”Ando só, como se voasse em bando.”

RESUMO

Atualmente temos a nossa disposição simuladores de redes em diferentes níveis de modelagem e precisão. A simulação em nível de pacotes, por apresentar maior detalhamento de informações e consequentemente maior precisão nos resultados, pode provocar um oneroso tempo para a obtenção dos mesmos. A simulação de redes *peer-to-peer* é uma das áreas que mais sofrem com esse problema. Isso ocorre porque normalmente essas redes apresentam uma grande quantidade de nós e um intenso tráfego de dados, exigindo bastante da escalabilidade do simulador. Várias pesquisas têm se dedicado a encontrar alternativas que vão desde a proposta de novas modelagens de simulação ao desenvolvimento de novos simuladores que oferecem a escalabilidade como um dos principais atrativos. O recém chegado Network Simulator 3 é um deles. O objetivo desse trabalho é avaliar o comportamento do NS-3 diante da simulação dessas grandes redes. Para isso, implementamos no simulador, o protocolo *peer-to-peer* Bittorrent em dois níveis de abstração: pacotes e fluxos.

Palavras-chave: Simulação de redes, P2P, NS-3

ABSTRACT

Currently we have network simulators available at different levels of modeling and accuracy. The packet level simulation, by presenting more detailed information and thus greater precision in the overall result may result in a costly time to obtain them. The simulation of peer-to-peer networks is one of the most affected areas by this problem. This is because these networks usually have a lot of nodes and heavy data traffic, requiring a lot of scalability from simulator. Several studies have been devoted to finding alternatives ranging from proposed new simulation modeling to develop new simulators that offer scalability as a major attraction. The newcomer Network Simulator 3 is one of them. The aim of this study is to evaluate the behavior of NS-3 on the simulation of these large networks. For this, we implemented in the simulator the peer-to-peer protocol Bittorrent at two levels of abstraction: packets and flows.

Keywords: Network Simulation, P2P, NS-3

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Motivação

A avaliação de desempenho de sistemas é o processo de análise e compreensão do sistema, permitindo a tomada de decisões sobre o mesmo. É possível, então, a realização de ajustes, identificação de gargalos, planejamento de capacidade e previsão de desempenho em determinadas situações críticas [5].

A simulação de redes é uma das principais técnicas de avaliação utilizadas pelos pesquisadores da área. O uso de simuladores permite a experimentação de novos protocolos, ajudando a entender o comportamento dos mesmos na rede. É ainda capaz de fornecer informações relacionadas aos fluxos de tráfego, filas, congestionamentos, descarte de pacotes, atrasos e verificar se os acordos de qualidade de serviço estão sendo respeitados.

O nível de abstração em que a simulação é executada geralmente determina a precisão dos resultados obtidos. Quanto maior a fidelidade do comportamento do simulador com o mundo real e quanto maior a quantidade de dados que ele tem acesso, mais confiáveis serão as informações extraídas. Em contrapartida, simuladores mais precisos normalmente exigem mais de sua capacidade de escalabilidade. O aumento do número de nós na rede ou o aumento da quantidade de tráfego circulante, por exemplo, poderá resultar em considerável aumento do tempo de processamento

da simulação. Em alguns casos, as proporções dessa relação aumentam de forma a tornar a experimentação inviável.

Aplicações de redes *peer-to-peer* para distribuição de recursos normalmente apresentam a característica de envolver uma grande quantidade de nós e haver um intenso tráfego de dados entre eles. Esses aspectos têm se mostrado um desafio quando o assunto é a simulação desses protocolos de rede, pois o elevado tráfego de arquivos sendo compartilhados entre os inúmeros membros da rede exige bastante da escalabilidade do simulador. Algumas propostas de simuladores que tentam encontrar alternativas recaem em problemas de precisão, confiabilidade, escassez de documentação etc. Questões como essas levam pesquisadores a reconhecer a carência de um simulador P2P que satisfaça suas necessidades. Isso colabora para que os mesmos tenham necessidade de criar ou adaptar seus próprios simuladores para satisfazer suas necessidades imediatas, prejudicando a validação de pesquisas e reprodução de resultados.

O Network Simulator 3 [6], por sua vez, é o recente sucessor do já difundido simulador de redes NS-2 [7]. Consiste de um *framework* ainda mais complexo atuando com maior fidelidade ao comportamento da rede, ao mesmo tempo em que devido a sua reestruturação de código e modificações na linguagem em que foi implementado, promete ganhos de desempenho.

Visando observar o desempenho do NS-3 na simulação de grandes redes com intenso tráfego e em diferentes níveis de abstração de simulação, realizamos um estudo que confronta os resultados desses experimentos em diferentes situações e diferentes simuladores. Como nativamente o NS-3 não tem suporte a nenhum protocolo *peer-to-peer*, foram desenvolvidos módulos de uma aplicação Bittorrent a fim de representar o cenário proposto. Os resultados são comparados com o simulador NS-2.

1.2 Trabalhos relacionados

A simulação de redes requer pesquisas e evoluções contínuas, uma vez que as necessidades exigidas dessas ferramentas eventualmente sofrem mudanças devido ao surgimento de novas tecnologias. Essas novas tecnologias podem ser na rede que precisa ser simulada, se referindo à evolução da velocidade e complexidade das mesmas, ou podem ser relativas às novas técnicas e ferramentas de simulação que permitem ganho de velocidade de processamento dos experimentos.

As atuais pesquisas na área têm se concentrado na experimentação de diferentes níveis de abstração com que os dados da rede são interpretados no simulador ou têm se concentrado na implementação de simuladores para fins específicos.

O grau de abstração da simulação define em que nível de detalhes o simulador trabalha. Determina, portanto, o quanto a simulação se aproxima do mundo real. O tráfego da rede pode ser visto como um conjunto de pacotes, onde todas as unidades de mensagens são processadas e interpretadas individualmente pelo simulador; um fluido que atravessa o meio, onde as mensagens da rede são abstraídas para um fluxo contínuo que se reajusta às características da rede a cada fila por onde passa; ou simplesmente o tráfego pode ser ignorado em algumas camadas da pilha de protocolos, como na simulação em nível de fluxos, onde são simuladas apenas saídas e chegadas dos pacotes.

Em [8], um estudo a respeito das técnicas de simulação de redes é apresentado, comparando-se o desempenho de tradicionais simuladores em nível de pacotes em relação a métodos híbridos que adotam a simulação em nível de fluidos. Verificou-se que isolando cenários desfavoráveis, a simulação em nível de fluidos é capaz de obter, com precisão, resultados semelhantes aos obtidos na simulação em nível de pacotes em um tempo de processamento bastante inferior, e portanto, sendo bastante útil para a simulação de redes com intenso tráfego.

Outros trabalhos discutem ainda a respeito dos níveis de abstração da simulação de redes, realizando comparações de desempenho e propondo modelagens para alguns protocolos. [4, 9, 10]

Como as redes P2P normalmente apresentam grande complexidade com relação ao número de nós e ao intenso tráfego que circula por ela, vários trabalhos foram realizados se concentrando no desenvolvimento de simuladores para protocolos e objetivos específicos.

GnutellaSim [11, 12] e GnuSim [13] são exemplos de simuladores do protocolo Gnutella. O primeiro desenvolvido sobre o NS-2 e o outro implementado com bibliotecas de simulação CSIM [14] e C++.

GPS [15] é outro simulador projetado inicialmente para qualquer protocolo P2P, mas que foca na modelagem do Bittorrent. É um simulador que trabalha em nível de fluxos de mensagens, ganhando em velocidade, mas podendo penalizar sua precisão.

OverSim é um *framework* de simulação de redes *overlay* baseado no OMNeT++ [16] e que promete ser um completo ambiente para a experimentação de novos protocolos P2P.

PlanetSim [17] e PeerSim [18, 19], são, por sua vez, simuladores implementados em linguagem Java. O primeiro é um *framework* para redes *overlay* focando na simulação de protocolos como o Chord. O segundo têm suporte a redes *overlay* estruturadas e não-estruturadas. Foi desenvolvido focando a sua escalabilidade, podendo ser executado em dois modos: baseado em ciclos, no qual sua documentação se concentra, ou no modo baseado em eventos. O modo baseado em ciclos é menos preciso, focando apenas na rede *overlay*, entre outras simplificações. É capaz, no entanto, de gerenciar um grande número de nós. No modo baseado em eventos, um conjunto de eventos são gerados e escalonados, sendo invocados de acordo com o tempo associado a cada um deles. É um modelo mais detalhado, porém, permite um menor número de *peers* simulados.

Algumas pesquisas [1, 20, 21, 22] realizaram estudos sobre esses diversos simuladores P2P. Essas mesmas pesquisas concluíram, entretanto, que nenhum deles obedece satisfatoriamente todos os requisitos esperados, permanecendo a busca e necessidade por desenvolvimento de ainda outros simuladores para tais fins.

O Network Simulator 3, por sua vez, foi projetado para ser capaz de simular uma grande quantidade de protocolos e com grande fidelidade ao comportamento da rede real. Com base em algumas leituras de sua documentação e alguns artigos publicados a seu respeito [23, 24, 6], o NS-3 promete maior precisão de informações e sobretudo escalabilidade com relação à capacidade de simular uma grande quantidade de nós sem comprometer seu desempenho. Em [23], o desempenho do NS-3 é comparado com o de outros simuladores como OMNeT++, NS-2, Jist e SimPy, onde um grupo de pequenos pacotes é propagado para um grande número de nós, sendo avaliados critérios como tempo de processamento e consumo de memória. O NS-3 foi o simulador que, de forma geral, apresentou melhores resultados.

Como visto anteriormente, pretende-se neste trabalho, implementar um protocolo *peer-to-peer* no NS-3 a fim de observar o seu desempenho diante da simulação dessa categoria de rede. Com base em estudos anteriores [25], que implementaram Bittorrent para o simulador NS-2 tanto em nível de pacotes como em nível de fluxos, optamos por implementar o mesmo protocolo Bittorrent no NS-3 a fim de permitir a comparação dos resultados obtidos. Nesse trabalho relacionado, o objetivo era mostrar que no caso específico das redes P2P, é possível obter uma considerável precisão da simulação em nível de fluxos frente à simulação em nível de pacotes porque o gargalo das redes *peer-to-peer* estaria localizado nas bordas, ou seja, no computador dos usuários, e não, por exemplo, nos canais dos provedores de serviços de Internet. Assim sendo, o fator limitante dessas redes seria a capacidade de transmissão de cada *peer*, não havendo, dessa forma, uma grande influência do protocolo TCP caso o simulador de fluxos, que negligencia as camadas de rede e transporte, adotasse essa taxa de transmissão dos usuários como sendo a taxa de transmissão geral na rede.

1.3 Estrutura do trabalho

O trabalho está organizado da seguinte forma: nesta seção de introdução damos uma visão geral sobre o que o trabalho abordará. Falamos sobre o que nos motivou a explorar o tema e então, apresentamos uma série de trabalhos relacionados que fundamentam e fortalecem nossas metas. O capítulo seguinte descreve com maiores detalhes alguns conceitos essenciais para a compreensão do trabalho, apresentando explanações acerca de redes *peer-to-peer*, avaliação de desempenho de sistemas, simulação de redes e protocolo TCP. Em seguida, expomos a metodologia aplicada. Todo o trajeto, ferramentas adotadas e implementações realizadas são descritos, para então, na penúltima seção apresentarmos e discutirmos os resultados dos testes executados. Por fim, o último capítulo aborda nossas conclusões, expectativas para melhorias e planos de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Redes *peer-to-peer*

As redes *peer-to-peer* (P2P) são redes projetadas com um nível de abstração mais elevado. Todos os nós participantes se conectam logicamente formando vizinhanças independentemente de suas localizações físicas. São portanto, por definição, redes *overlay*, ou seja, redes virtuais criadas sobre uma rede já existente. Tem-se uma topologia física e uma topologia virtual que liga *peers* de uma mesma aplicação, com objetivos comuns e sem restrições de quantidade de participantes ou de tempo de conexão entre eles [1, 26].

A popularização das redes *peer-to-peer* modificou os padrões de uso até então da Internet. Apesar de sua infraestrutura também abranger aplicações de mensagem instantânea e computação distribuída, foram as vantagens oferecidas ao compartilhamento de arquivos que a tornou atrativa e popular. A principal vantagem das redes P2P é permitir maior escalabilidade e distribuição de recursos de forma colaborativa entre seus participantes. A distribuição dos arquivos entre várias máquinas, além de aumentar a disponibilidade dos mesmos, reduz a sobrecarga dos recursos de rede, evitando a concentração das requisições para um único servidor.

Na tecnologia P2P não existe entidade central nem hierarquia. Nela, todos os integrantes podem executar papel de cliente e servidor, ou seja, têm as mesmas

capacidades funcionais e responsabilidades. O acesso aos recursos do parceiro é realizado de forma direta. Há capacidade de lidar com diferentes estruturas, conectividade temporária e variação de endereços. Uma grande e robusta rede é construída usando os modestos dispositivos de hardware e rede dos próprios usuários.

As redes P2P precisam garantir a escalabilidade dos nós, distribuição de conteúdo, localização eficiente de dados, autenticação, armazenamento redundante, além de oferecer ferramentas de seleção de *peers* etc. É o nível de abstração mais elevado que facilita a resolução desses problemas, que seriam de difícil solução se tratados de forma convencional.

Os protocolos de redes P2P normalmente são diferenciados pela arquitetura e método de busca utilizados [27]. Arquiteturas centralizadas apresentam uma entidade central de grande porte que armazena e mantém atualizadas as informações da rede. Normalmente a busca também é centralizada, mas as informações podem ser difundidas pelos *peers* entre si. O Napster, que foi a primeira aplicação P2P de compartilhamento de arquivos a se destacar, usava essa organização topológica. O usuário realizava uma requisição ao servidor central que por sua vez respondia onde estava localizado o arquivo desejado, para então ser estabelecida a conexão direta entre os dois nós. Apesar de utilizar uma entidade centralizadora, e portanto apresentar um gargalo em potencial, o desempenho da rede não era comprometido, uma vez que a troca dos arquivos compartilhados ocorria diretamente entre os *peers*[28].

Na arquitetura descentralizada/estruturada não há a presença do servidor central, mas a topologia dos nós é controlada e o conteúdo da rede é alocado estrategicamente facilitando a posterior busca de arquivos. Essas informações são gerenciadas em estruturas chamadas DHT (*Distributed Hash Table*) onde cada objeto recebe um identificador único para que seja devidamente mapeado em todo o sistema. Cada *peer* será responsável por mapear um conjunto de registros de objetos e deverá manter também uma pequena tabela de roteamento para que possa encaminhar requisições a seus vizinhos. Cada registro deve poder ser acessado, removido ou inserido usando sempre o mesmo espaço de identificadores. Diferentes sistemas

P2P adotarão diferentes esquemas de organização e estratégias de roteamento para a localização desses objetos, mas em geral bons esquemas permitem localizar qualquer arquivo na ordem de O(logN), onde N é o número de *peers* na rede. Alguns desses sistemas são o Chord, Pastry, Tapestry e CAN [1].

Finalmente, uma arquitetura descentralizada/não-estruturada, não apresenta um servidor central nem tenta organizar os nós estrategicamente. Utiliza busca por *broadcast*, mas limitada à uma determinada vizinhança mais próxima, não abrangendo todo o espaço de nós disponíveis quando a rede tem grande escala. É o modelo adotado pela rede Gnutella, onde cada requisição é encaminhada aos nós diretamente conectados, os quais enviam para os nós conectados a eles e assim sucessivamente até que o arquivo seja encontrado ou ocorra o número máximo de encaminhamentos. Uma vez que apresenta arquitetura totalmente descentralizada, espera-se maior escalabilidade e robustez, porém, como não apresenta uma estratégia de busca mais eficiente, o grande número de nós compromete seu desempenho quando o objetivo é que a requisição alcance todos os participantes. Algumas tentativas para melhorar o desempenho desse modelo são o uso de *caching* de pesquisas recentes e a criação de supernós, que são máquinas de maior potência que armazenam informações dos *peers* mais próximos. A busca é inicialmente executada no supernó mais próximo, para então ser encaminhada aos demais supernós [29].

Redes P2P estruturadas são mais eficientes na localização de arquivos raros, no entanto são mais complexas e geram maior *overhead* do que redes P2P não-estruturadas [1].

As redes não-estruturadas apresentam escalabilidade limitada. O modelo de requisições por enxurrada, pode ser eficiente para objetos populares, entretanto, pode consumir bastante largura de banda ou não ser capaz de encontrar arquivos raros, devido ao limite de saltos imposto pelo TTL (*Time to Live*). Apesar de permitir a eficiente localização de qualquer objeto compartilhado na rede, desde que ele exista, a complexidade dos modelos estruturados normalmente não é necessária em redes utilizadas em massa, daí as aplicações não-estruturadas ainda serem predominante-

mente utilizadas [1].

2.1.1 Chord (Arquitetura descentralizada/estruturada)

No Chord, tanto *peers* como arquivos compartilhados recebem um identificador *hash* de m bits baseado no algoritmo SHA-1. Os *peers* são ordenados circularmente e por sua vez, a chave k de cada arquivo é associada ao primeiro *peer* cujo identificador é maior ou igual a essa chave. Esse *peer* é denotado como o sucessor da chave k. Quando um novo *peer* entra na rede, as chaves associadas ao seu sucessor devem ser reavaliadas para saber se devem estar associadas ao novo *peer* que entrou. Quando um *peer* sai da rede, todas as chaves passam a estar associadas ao sucessor dele [30].

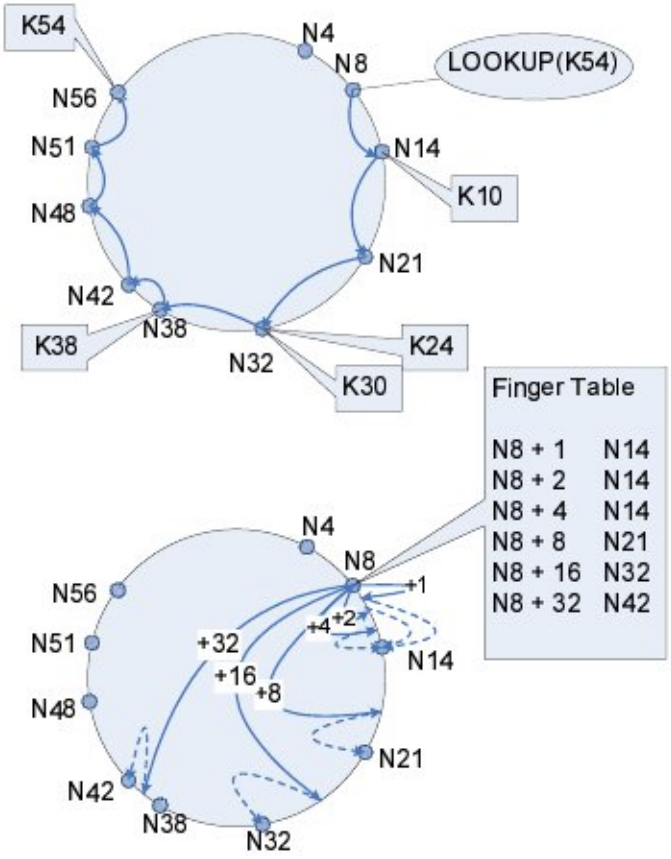


Figura 2.1: Exemplo de consulta da chave 54 dentro de um anel Chord [1]

A Figura 2.1 mostra um anel Chord com espaço de endereçamento $m = 6$ bits. O anel apresenta 10 *peers* e 5 chaves armazenadas entre eles. No exemplo,

o sucessor da chave 10 é o *peer* 14, assim, a chave será armazenada nesse *peer*. Da mesma forma, se um *peer* com identificador 26 entrar na rede, a chave 24, que estava em 32, passará a estar associada ao novo nó. Para tentar agilizar a busca, cada *peer* mantém uma tabela de roteamento com m entradas. A iézima entrada aponta para o *peer* definido pela seguinte fórmula $sucessor(2^i - 1) mod 2^m$. A figura mostra também a tabela de roteamento do nó 8. Devido a entrada e saída de *peers*, periodicamente o Chord executa um protocolo de ”estabilização”a fim de atualizar as tabelas de roteamento e os sucessores de cada nó. Ainda assim, devido a dinâmica e entrada/saída de nós ou eventuais falhas nos algoritmos de atualização, cada nó mantém uma lista de alguns sucessores sequenciais no anel.

2.1.2 Gnutella (Arquitetura descentralizada/não-estruturada)

Gnutella é um protocolo descentralizado onde o *peer* localiza o arquivo desejado enviando uma enxurrada de mensagens para sua vizinhança até um determinado limite de saltos. Seu projeto é tolerante a falhas e bastante adaptável a entrada/saída de nós da rede, porém, o mecanismo de busca não é escalável. É uma rede com nós independentes e que se auto-organizam. O *peer* inicia sua participação na rede se conectando a um *host* inicial e previamente conhecido, para então interagir trocando mensagens de *broadcast* com os demais *peers* da rede. Cada mensagem recebe um identificador, devendo o nó guardar em memória as mensagens roteadas mais recentes para evitar reencaminhar mensagens desnecessárias. As mensagens possuem um limite de saltos (TTL - *Time To Live*) para evitar congestionamentos e sua circulação indefinida. Sua implementação é bem simples e suas mensagens se resumem em mensagens de inserção na rede (mensagens de *broadcast* com sua vizinhança para a troca de informações iniciais); mensagens para consulta de objetos; mensagens de resposta às consultas; e mensagens de transferência de arquivos. Periodicamente os *peers* reenviam mensagens de *ping* na sua vizinhança para verificar a entrada ou saída de nós [29].

2.1.3 Bittorrent (Arquitetura centralizada)

O Bittorrent [31, 32] é um protocolo P2P que usa uma entidade centralizadora para gerenciar os usuários interessados em obter um mesmo arquivo. As requisições e trocas de dados permanecem sendo realizadas diretamente entre os *peers*. É um protoloco para distribuição de grandes recursos que interessam simultaneamente a uma grande quantidade de pessoas. Se esse arquivo compartilhado estivesse armazenado em uma rede cliente/servidor centralizada, o aumento da demanda do mesmo exigiria de forma proporcional o aumento de largura de banda e processamento da rede.

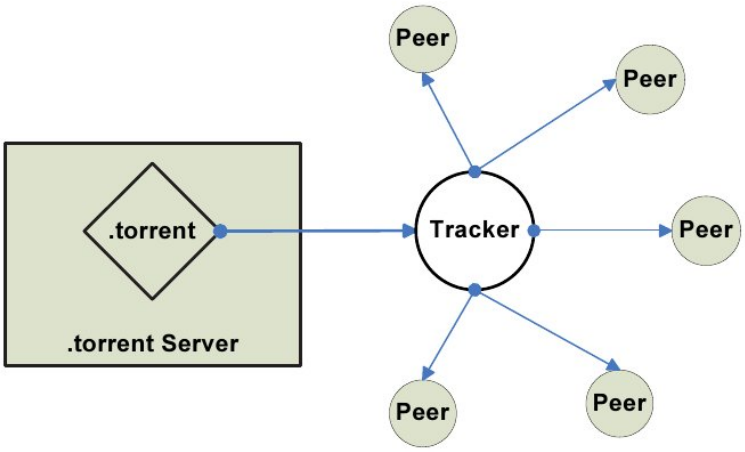


Figura 2.2: Arquitetura do Bittorrent

O Bittorrent oferece essa distribuição de recursos de forma escalável, baseada na cooperação entre os nós que naquele momento participam do processo. Enquanto um *peer* obtém um dos trechos do arquivo desejado de uma determinada fonte, simultaneamente ele disponibiliza para *upload* o conteúdo do arquivo que ele já tenha adquirido. Quando ele conclui o *download*, suas conexões de *upload* podem permanecer ativas. Apesar de, na maioria das vezes, os usuários finalizarem a aplicação ao concluir um *download*, há sempre a entrada de novos parceiros requisitando e replicando novos trechos do arquivo compartilhado.

O processo é iniciado publicando-se um arquivo comum com extensão *.torrent* em um servidor web qualquer. O *.torrent* apresenta as informações do arquivo buscado, tamanho, *hashing* e a url de um *tracker*. *Tracker* é a entidade responsável por localizar os *hosts* que naquele momento estão baixando o mesmo arquivo. A

comunicação com o *tracker* é feita em nivel HTTP, requer pouca largura de banda e resulta na entrega de uma lista com informações de contato para os parceiros. O *tracker* não toma decisões, nem aloca *peers* baseando-se em taxas de *upload* ou *download*. Sua única função é mapear a localização deles, retornando aleatoriamente uma lista de contato para uma determinada quantidade de nós. Para essa lista ser inicializada, é necessário haver ao menos um *peer* que já tenha o arquivo completo e esteja disposto a distribuir ao menos uma cópia dele entre os demais usuários. Os *peers* que possuem cópias completas de arquivos são chamados *seeds*.

Para ser capaz de identificar quais trechos de arquivo cada *peer* já adquiriu, o Bittorrent divide o arquivo principal em pedaços de tamanho fixo, normalmente $\frac{1}{4}$MB, chamados *chunks*. Para verificar a integridade de cada trecho recebido, o *peer* deve calcular o *hash* SHA-1 e comparar com os apresentados no *.torrent*. Cada nó é capaz, então, de relatar, após a verificação, quais os *chunks* que ele possui. As conexões entre os *peers* são simétricas, podendo ocorrer troca de mensagens em ambos os sentidos. O envio de cada mensagem de trecho de arquivo ao outro *host* é limitado a um tamanho máximo (*piece*) permitido por requisição.

Durante toda sua execução, cada nó executa algoritmos de seleção do *chunk* e algoritmos de seleção do *peer*. A aplicação deve decidir estrategicamente de qual *chunk* realizar cada requisição. Essas políticas evitam que haja uma má distribuição dos *chunks* replicados na rede, além de proporcionar que cada um deles seja completado o mais rápido possível para que esteja disponível aos demais parceiros. Por sua vez, as políticas de seleção do *peer* permitem definir, dentre aqueles concorrentes, quais deles, em uma fatia de tempo, terão permissão de realizar requisições ao nó. Essas ações de bloqueio/desbloqueio (*choking/unchoking*) são executadas periodicamente e priorizam os *peers* mais colaborativos na rede.

2.2 Avaliação de Desempenho de Sistemas

A avaliação de desempenho de um sistema consiste na observação das características e interpretação dos resultados obtidos com a submissão desse sistema a um conjunto de experimentos, permitindo assim, a tomada de decisões. A observação do seu comportamento proporciona a compreensão do seu funcionamento e a previsão de situações problemáticas, podendo haver assim, o planejamento de ações de acordo com as finalidades previstas.

É possível, então, verificar se todos os requerimentos do sistema estão sendo respeitados, determinar valores ótimos para seus parâmetros, identificar gargalos no sistema, planejar capacidade e prever seu desempenho em determinadas situações críticas.

Pode-se também identificar quais são os fatores responsáveis por um possível desempenho indesejado do sistema, e assim, permitir a aplicação das medidas corretas, uma vez que o aumento de determinado recurso pode não significar necessariamente a ocorrência de melhorias ou, caso ocorram, os ganhos de desempenho podem não ocorrer na proporção imaginada.

A avaliação de desempenho em sistemas de comunicação é capaz de fornecer informações relacionadas aos fluxos de tráfego ou políticas aplicadas nas filas de roteamento. Pode-se identificar congestionamentos ou gargalos no cenário analisado, determinar os aspectos que levam ao início do descarte de pacotes, entre outras aplicações.

A comparação do desempenho do sistema avaliado com o desempenho de outros sistemas semelhantes ou com padrões de qualidade, proporciona uma avaliação qualitativa do mesmo, e não somente quantitativa. Ou seja, a interpretação dos resultados com base em um referencial gera a idéia do quão relativamente bom é o sistema.

A forma como a execução das avaliações são realizadas, bem como a escolha das métricas de desempenho (tempo de resposta, consumo de recursos, *throughput*, etc.) devem ser criteriosas. Por um lado, aspectos como tempo de

duração dos testes, número de repetições e geração aleatória dos dados, proporcionam resultados imparciais e consistentes. Já a escolha de métricas adequadas, assim como as inferências estatísticas aplicadas corretamente sobre elas, permitem fornecer informações úteis, precisas e corretas.

2.2.1 Técnicas de Avaliação de Sistemas

A avaliação de desempenho do sistema pode ser realizada através de monitoração, simulação ou modelagem analítica. Como existe uma grande variedade de tipos de aplicações, não se pode generalizar e apontar qual dessas técnicas é a melhor. Cada uma tem suas vantagens e cada situação requer uma técnica específica.[5]

- Monitoração ou medição: é a observação do comportamento do sistema real e coletagem dos dados observados para o levantamento estatístico. Embora seja uma técnica confiável, pode ser muito custosa e demorada, uma vez que o sistema precisa ser implementado por completo para a sua realização

- Simulação: nesta técnica implementa-se uma ferramenta que simula o funcionamento do sistema fazendo uso de abstrações que dependendo do grau em que são aplicadas, podem representar o sistema com maior ou menor fidelidade [33]. Promovem ganho de tempo e flexibilidade nos parâmetros de testes em relação a monitoração.

- Modelagem analítica: consiste na elaboração de equações matemáticas que representem a dinâmica do sistema. Na maioria das vezes, a aplicação não é simples o suficiente para ser possível formular todas variáveis envolvidas, e então obtermos resultados exatos. Dessa forma, é comum o uso de simplificações ou omissão de parâmetros que tenham pouca influência sobre o comportamento do sistema, podendo, no entanto, resultar em conclusões diferentes das reais.

O estágio do ciclo de vida do sistema em que uma técnica de avaliação pode ser aplicada, varia de acordo com seu tipo. Diferentemente da medição, que

Tabela 2.1: Critérios para escolha da técnica de avaliação de desempenho [5]

Critério	Modelagem Analítica	Simulação	Medição
Estágio	Todos	Todos	Pós-implementação
Tempo	Pouco	Médio	Variado
Habilidades	Análise matemática	Linguagens de programação	Operação do sistema
Precisão	Variada	Média/Variada	Alta/Variada
Parâmetros	Fácil	Moderado	Difícil
Custo	Baixo	Médio	Alto

exige que o projeto já esteja implementado para poder ser realizada, as técnicas de simulação e a modelagem analítica são independentes da fase do implementação do sistema.

Com relação ao tempo necessário para a conclusão dos testes relativos à análise, naturalmente a modelagem analítica corresponde a técnica mais rápida, enquanto na medição, o tempo necessário para a avaliação torna-se intimamente relativo às características do sistema. A simulação tem seu tempo inversamente proporcional ao grau de abstração em que o sistema será considerado. Quanto maior a abstração, menor o tempo necessário e quanto menor a abstração, maior o tempo de simulação.

No que diz respeito às habilidades necessárias ao usuário que implementará uma das técnicas de avaliação, a modelagem analítica normalmente exige sólidos conhecimentos matemáticos. Já na simulação, se o usuário precisar criar seu próprio simulador, conhecimentos de linguagens de programação são exigidos, enquanto na medição é necessário saber operar adequadamente o sistema.

A precisão é outro fator crítico, afinal, se a capacidade de apresentar resultados dentro de um determinado intervalo de confiança não é obedecido, a análise de desempenho pode ser invalidada. Como discutido anteriormente, dada a dificuldade de tratarmos todas as variáveis que influenciam no comportamento do sistema, os modelos analíticos podem ser simplificados havendo redução na precisão dos resultados. Dessa forma, a simulação tende a obter maior precisão que a modelagem analítica, porém, mais uma vez inversamente proporcional ao nível de abstração adotado. A

técnica de medição normalmente apresenta precisão superior às demais técnicas. No entanto, o próprio sistema implementado pode apresentar limitações se executado em um momento ou cenário que não condiz com sua normalidade. Por essa razão sua precisão é considerada variável.

Como a avaliação de desempenho também busca encontrar o conjunto de parâmetros do sistema que otimizem o seu comportamento, a facilidade com que os testes podem ser adaptados variando esses parâmetros, também é um aspecto que deve ser considerado. A modelagem analítica é a técnica que oferece maior flexibilidade para os ajustes e testes das variáveis. A simulação também permite essa flexibilidade embora algumas vezes exija a reprogramação do simulador. A obtenção de parâmetros ótimos na medição não é uma tarefa simples uma vez que é necessário criar novo protótipo e reimplementar o projeto.

Outro fator importante é o custo envolvido no processo da avaliação do desempenho do sistema. A modelagem analítica e a simulação são, sem dúvida, as técnicas mais baratas nesse aspecto, embora a simulação possa ser um pouco mais encarecida se for necessária a produção de um software simulador específico. O custo envolvido na técnica de medição corresponde ao custo do projeto, dessa forma, mudanças necessárias observadas após os testes de desempenho significam o custo da reimplementação do sistema.

2.2.2 Fundamentos de Simulação de Redes

Normalmente a simulação é a técnica mais adequada para o estudo das redes de computadores. Os simuladores de redes são aplicados principalmente no desenvolvimento e experimentos de novos protocolos. São ferramentas capazes de simular o comportamento do sistema em um determinado nível de abstração através do gerenciamento de um conjunto de variáveis cujos valores definem o estado da rede naquele instante.

As mudanças nas variáveis de estado da rede, por sua vez, são provocadas pela ocorrência de eventos que representam a dinamicidade e interação dos elemen-

tos da rede. O simulador de redes que trabalha com esse conceito é chamado de simulador orientado a eventos. Para funcionar de forma controlada, esses simuladores possuem um escalonador de eventos central, que gera um conjunto de eventos organizados em fila, possuindo cada um deles um tempo de execução. À medida que cada evento é processado, o tempo global do simulador é atualizado para o tempo desse evento. Esse processo se repete até que não haja mais eventos agendados ou até que o limite de tempo da simulação seja alcançado [34]. Esse esquema é o mais adequado para a simulação de redes. A Figura 2.3 ilustra a estrutura do gerenciador de eventos de um simulador que usa essa técnica.

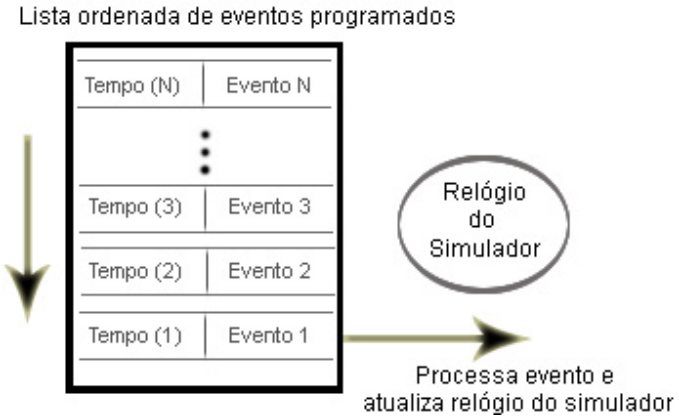


Figura 2.3: Arquitetura de um simulador orientado a eventos

Uma outra modalidade de simulador, são aqueles baseados em ciclos (simulação orientada a tempo), periodicamente cada nó da rede é consultado verificando se há operações a serem executadas. Quando todos os nós da rede são consultados, completa-se um ciclo e recomeça o processo. Esse método pode resultar em alto custo computacional, uma vez que todos os nós da rede são necessariamente consultados a cada intervalo de tempo.

O relógio do simulador é responsável por relacionar e coordenar a variável tempo interna do simulador com o tempo real do sistema. Se o tempo não é considerado uma variável, ou seja, o estado do sistema não muda com o tempo, temos um modelo estático, caso contrário, temos um modelo de simulação dinâmico [35].

Todo simulador de rede deve ser capaz de representar ao menos o comportamento básico das principais entidades da rede: o nó, o enlace e o tráfego.

Os nós representam os roteadores ou os *hosts*. Apresentam objetos associados a eles, como por exemplo filas em cada interface de nó com enlace. Essas filas possuem uma capacidade máxima de processamento e servem para armazenar pacotes que ainda não foram encaminhados ou processados. Quando seu limite de armazenamento é alcançado ocorrerá o descarte dos pacotes de acordo com a política adotada por elas.

Os enlaces apresentarão atributos como capacidade e tempo de propagação do dados de um nó para outro.

Por fim, o tráfego da rede (carga) caracterizará o tipo de dado que atuará na rede simulada, a taxa de transmissão de cada nó e a modelagem de tráfego que, se adotada, insere distribuições probabilísticas sobre ele permitindo aumentar o realismo da aplicação. Dependendo do grau de abstração da simulação, os dados podem ser vistos como um conjunto de pacotes ou um fluido que atravessa o meio.

Para facilitar a mudança de parâmetros e a visualização dos resultados gerados pelo simulador, normalmente os mesmos utilizam a leitura de *scripts* como dados de entrada e geram saídas em arquivos de texto (*trace*) ou animações que descrevem os eventos ocorridos durante a simulação. Os *scripts* de entrada podem apresentar as informações do cenário, parâmetros da rede, além de eventos definidos pelo usuário.

Quando as saídas do simulador são totalmente determinadas pelos valores passados como parâmetros de entrada, ou seja, seus resultados podem ser previstos sempre que os mesmos valores de entrada forem utilizados, o simulador é considerado determinístico. Caso contrário, teremos modelos estocásticos ou probabilísticos, pois participarão da simulação alguns aspectos de aleatorieadade, assim, os testes, quando repetidos diversas vezes, mesmo utilizando os mesmos dados de entrada, apresentarão resultados diversos.

2.2.3 Níveis de simulação de redes

A simulação de grandes redes com intenso tráfego de dados implica um oneroso tempo para se obter os resultados desejados. Com a necessidade de simular redes cada vez maiores e mais complexas, tem-se procurado formas alternativa de resolver o problema.

Pode-se conquistar ganho de velocidade na simulação abordando a dimensão física, que se aplica aumentando a velocidade dos processadores ou o processamento paralelo; a dimensão lógica, relacionada com a melhoria nos algoritmos implementados no simulador; e por fim abordando a dimensão da modelagem de simulação, onde os ganhos podem ser obtidos por meio de mudanças na forma de tratar o objeto que está sendo submetido ao processo de simulação.

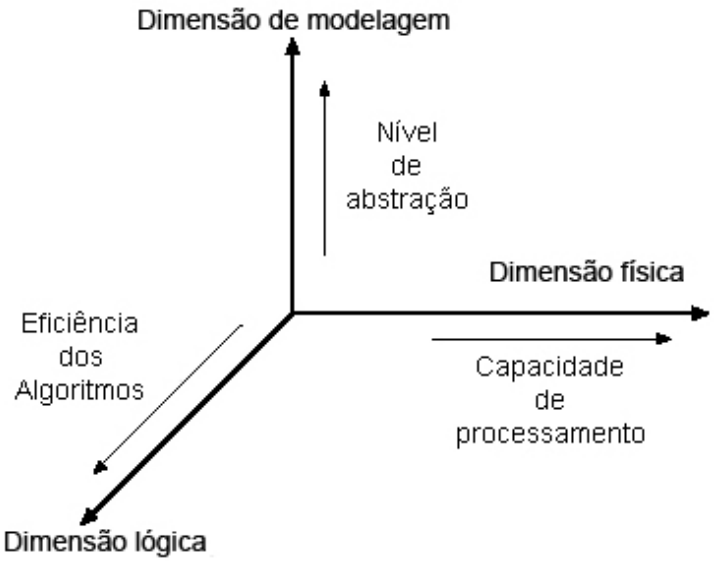


Figura 2.4: Aspectos que influenciam no desempenho na simulação [2]

A essa terceira dimensão, a modelagem, aplica-se o grau de abstração em que o tráfego da rede será simulado. O mesmo pode ser tratado como um conjunto de pacotes, um fluido que atravessa o meio ou simplesmente ser ignorado em algumas camadas da pilha de protocolos.

A precisão da simulação está intimamente relacionada ao grau de abstração, definindo em que nível de detalhes o simulador trabalha. Determina, portanto, o quanto a simulação se aproxima do mundo real. Em contrapartida, a precisão pode

estar relacionada à escalabilidade do simulador. Em simuladores de nível de pacotes, por exemplo, com o aumento do número de nós na rede ou o aumento da quantidade de tráfego circulante, aumenta-se consideravelmente o tempo de processamento da simulação.

A simulação em nível de pacotes é a que mais se aproxima do comportamento real de uma rede. Cada entrada/saída de pacotes das filas, descartes e recebimentos de mensagens gera um novo evento a ser processado pelo simulador. São portanto simuladores bastante precisos e detalhados em todos os eventos ocorridos na rede. A simulação torna-se ainda mais complexa de acordo com o grau de detalhes com que a pilha de protocolos do simulador é implementada. São entretanto, mais demorados para concluir tarefas de simulação. Normalmente não são escaláveis. O aumento da quantidade de nós ou capacidade dos canais de comunicação e consequentemente a quantidade de pacotes trafegando na rede resulta em um maior número de eventos gerados e portanto um tempo de simulação mais oneroso.

Na simulação em nível de fluxos, a influência da camada de rede e de transporte na simulação é ignorada, sendo o tráfego entregue diretamente a camada de aplicação do nó destino. A política de filas, a busca da rota, e por exemplo, a atuação do protocolo TCP na camada de transporte não são levados em consideração. Calculam-se informações de tráfego como retardo, *throughput*, pacotes perdidos, entre outros, apenas fim a fim e com base nas informações já existentes da rede ou com base nos dados obtidos na camada de aplicação. Dependendo do que precisarão simular e da forma como foram implementados, poderão ser mais limitados com relação a precisão, porém, por controlarem menos informação circulante na rede, são mais rápidos em suas simulações.

A simulação em nível de fluidos é uma alternativa recente aos tradicionais simuladores de redes de pacotes e de fluxos. Como nos simuladores de pacotes, eles abrangem todas as camadas da pilha de rede, porém, tratam os dados que trafegam pelos canais de comunicação como se fosse um fluxo de fluido contínuo, em vez de pacotes. Apresentam maior nível de realidade que a simulação baseada em fluxos,

uma vez que os dados precisam passar por filas em roteadores e sofrem a ação das políticas de congestionamento da camada de transporte. Espera-se com isso, uma pequena redução do grau de precisão em relação aos simuladores de pacotes, mas um grande ganho no tempo de processamento [8, 2, 4, 9].

2.2.4 Simulação de redes em nível de fluidos

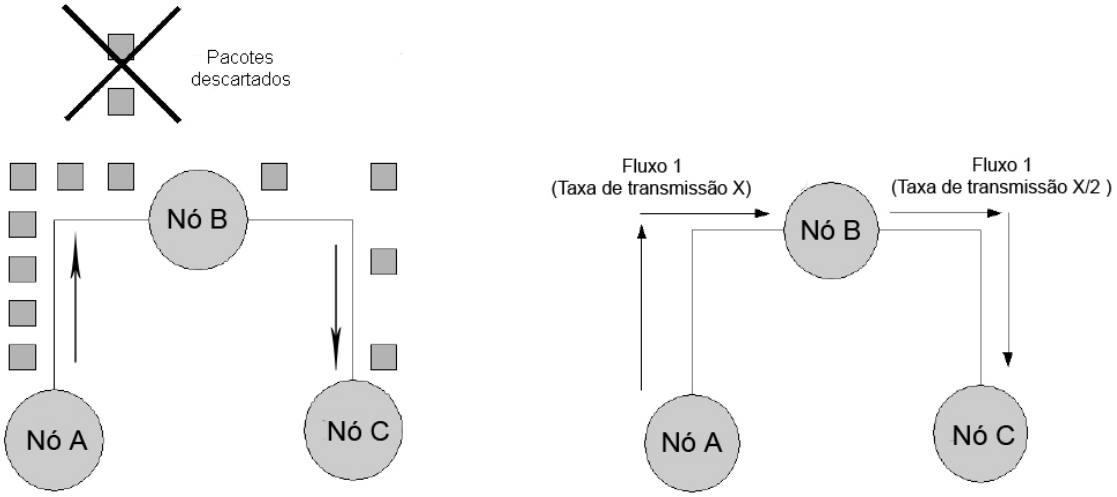
A simulação em nível de fluidos trabalha resolvendo equações diferenciais que modelam as políticas de fila e o tráfego da rede como fluxos de fluido [36, 3, 37, 2]. As soluções dessas equações geram os eventos, que por sua vez serão escalonados e deverão ser tratados pelo simulador de rede. Os eventos do modelo de fluidos são tratados de forma diferenciada em relação a simulação baseada em pacotes. Enquanto esta monitora e dedica atenção a todos os pacotes que circulam na rede, no nível de fluidos, a monitoração é sensível apenas à variação das taxas de fluxo (vazão) das fontes e das filas da rede [2].

- São considerados eventos nos simuladores de pacotes:
  - Geração de pacotes pelas fontes;
  - Entrada e saída de cada pacote das filas;
  - Recebimento de pacotes pelos nós;
  - Entre outros eventos como descartes, colisões, etc.;

- São considerados eventos nos simuladores de fluidos:
  - Mudanças nas taxas de saída de cada fluxo nas fontes;
  - Mudanças nas taxas de entrada ou saída de cada fluxo nas filas.

A Figura 2.5 exemplifica a diferença no tratamento da perda de dados nos simuladores de pacotes em relação aos simuladores de fluidos. Nos simuladores tradicionais novos eventos são gerados sempre que cada pacote chega, sai ou é descar-

tado pelo nó B. Nos simuladores de fluidos apenas um evento é gerado, informando a mudança na taxa de fluxo que passou pelo nó.



(a) Simulação em pacotes (b) Simulação em fluidos

Figura 2.5: Tratamento dos eventos de descarte em ambos os métodos e simulação [3]

Se a taxa de transmissão do nó A fosse N vezes maior, o simulador de pacotes necessitaria tratar na ordem de N vezes mais pacotes gerando assim N vezes mais eventos. Porém, quando utilizado o simulador de fluidos, a quantidade de eventos processados permanece a mesma, evidenciando a escalabilidade deste modelo de simulação.

Por sua vez, as Figuras 2.6 e 2.7 ilustram os diferentes eventos gerados por simuladores no nível de pacotes e simuladores em nível de fluidos sob influência de uma fila FIFO (*First In First Out*). Em um simulador de pacotes, para esse tipo de fila, o tráfego chega de diversas fontes e sai discretamente na ordem em que chega, de acordo com a capacidade da fila ou do enlace. Embora sua política continue a mesma, a dinâmica de uma fila FIFO em simuladores de fluidos é bem mais complexa. Nessas filas, podem chegar ou sair simultaneamente fluxos de diferentes fontes, mas estes não podem se misturar dentro dela.

No simulador em fluidos, quando dois ou mais fluxos concorrem e juntos extrapolam a capacidade de processamento da fila ou do canal de destino, a taxa de saída relativa ao fluxo de cada fonte é proporcional à sua taxa de entrada na fila em questão.

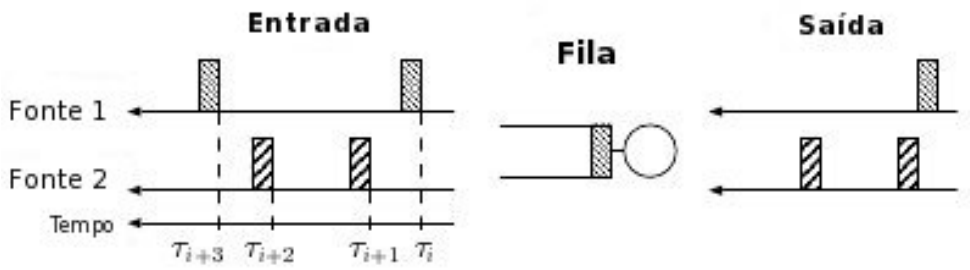


Figura 2.6: Filas FIFO em simuladores baseados em pacotes [2].

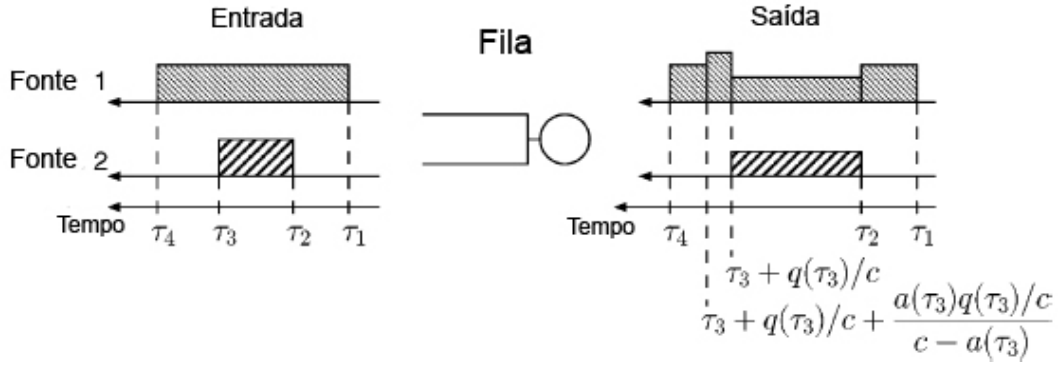


Figura 2.7: Filas FIFO em simuladores baseados em fluidos [2].

No exemplo acima, os fluxos de ambas as fontes chegam a uma taxa f numa fila de capacidade c, tal que $f < c < 2f$. Na mesma figura 2.7, o tempo da ocorrência de cada evento pode ser calculado em função da taxa de entrada $a(T)$ e do tamanho da fila $q(T)$ naquele instante.

Em algumas situações, a complexidade da dinâmica de fluidos pode exigir muito esforço computacional negando seu principal benefício. Podemos notar, no exemplo anterior, que embora sejam apenas dois fluxos concorrendo em apenas uma fila, ocorreram várias mudanças em suas taxas de saída. Essas variações foram decorrentes do acúmulo de dados na fila durante o período em que a exigência de processamento era superior a sua capacidade. Quando a fila esvaziou, o fluxo da fonte 1 voltou ao normal.

Os eventos no simulador de pacotes corresponderam às chegadas e saídas de cada pacote na fila. Já no simulador de fluidos, foram gerados eventos sempre que a taxa de saída da fila sofreu mudanças.

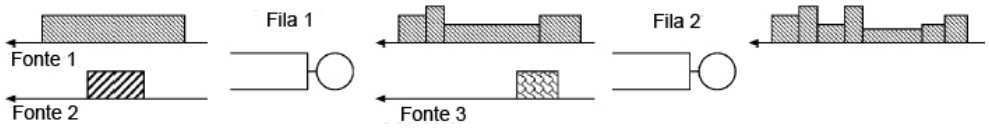


Figura 2.8: Efeito *Ripple* [2]

Essas variações de taxas ficam ainda mais frequentes quando os fluxos passam por diversas filas e sofrem influência de outros fluxos, resultando em perda de eficiência na simulação de fluidos. Esse efeito é conhecido como *ripple effect* [2], como mostra a Figura 2.8.

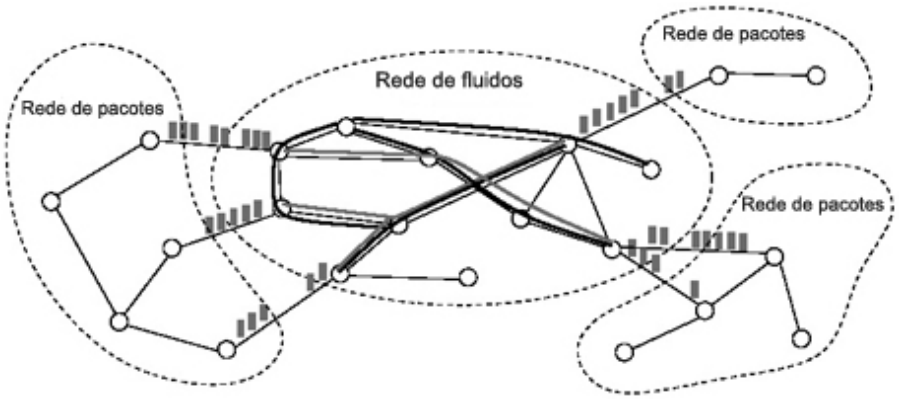


Figura 2.9: Simulação de uma rede híbrida [4]

Alguns trabalhos [8] comprovam que os simuladores de redes baseados em fluidos são capazes de fornecer a mesma precisão de simuladores em nível de pacotes em um tempo de processamento muito reduzido, garantindo a escalabilidade esperada. Em contrapartida, como os pacotes estão abstraídos num mesmo fluido, não é mais possível acompanhá-los individualmente, como era possível na simulação de pacotes. Pode-se, então, tornar a implementação do simulador ainda mais complexa, aplicando a simulação híbrida [4, 38], como na Figura 2.9, em que um trecho da rede é simulado com fluidos e um outro trecho, em que se deseja obter informações mais detalhadas, é simulado usando nível de pacotes. Os pacotes podem ser gerados individualmente pelas fontes e a partir de determinado nó da rede os fluxos de pacotes podem ser abstraídos e transformados em fluxos de fluidos retornando a serem tratados como pacotes ao chegarem ao destino. Uma grande quantidade de canais e roteadores podem ser atravessados pelos fluidos poupando processamento

da simulação de pacotes.

2.2.5 Simuladores de redes

Atualmente temos a nossa disposição uma grande variedade de simuladores de redes [39]. Alguns deles foram criados visando a simulação de protocolos ou arquiteturas específicas, enquanto outros constituem um *framework* capaz de simular diversos tipos de redes e aplicações. Cada um deles apresenta características peculiares que o tornam mais apropriados a determinados tipos de redes. Alguns dos aspectos que diferem esses simuladores são: a arquitetura, a precisão das informações, a velocidade, escalabilidade, usabilidade, documentação, entre outros.

- Arquitetura do simulador

  A arquitetura do simulador avalia a forma como ele foi projetado. Costuma-se classificá-lo de acordo com o método de escalonamento ou verificação da ocorrência de eventos. Como visto anteriormente, simuladores orientados a eventos trabalham com o conceito de eventos escalonados por fatias de tempo. À medida que eles são executados, podem provocar novos eventos, que são mais uma vez direcionados para a fila de processamento. A outra arquitetura referenciada é a orientada ao tempo, ou baseada em ciclos. Nela, em vez de cada evento ter seu próprio rótulo de tempo, é o simulador que verifica a cada intervalo se há rotinas a serem executadas em cada nó.

- Precisão das informações

  A precisão das informações avalia em que nível de detalhes o simulador trabalha, que por sua vez está intimamente relacionado ao nível de abstração que ele adota.

  Como vimos anteriormente, os simuladores de redes podem atuar em nível de pacotes, fluxos ou fluidos. Na simulação em nível de pacotes, todo o trajeto, comportamento e passagens dos pacotes resultam em eventos que deverão se processados pelo simulador. Apesar de proporcionar elevado nível de precisão, isso

resulta em elevado custo computacional, e dependendo no tamanho da rede, a conclusão da simulação pode se tornar inviável.

A simulação em nível de fluxos constitui um menor nível de precisão, uma vez que ignora algumas camadas da pilha de protocolos (geralmente as camadas de rede e transporte) visando gerar um menor número de eventos a serem processados pelo simulador. Isso resulta, entretanto, em baixo nível de realidade das informações apresentadas.

A simulação em nível de fluidos constitui um meio termo entre os dois tipos de simulação citados acima. Há a simulação de todas as camadas da pilha de protocolos, porém, o tráfego é enxergado como um fluido que atravessa os canais da rede. Resulta em ganho de desempenho em relação à simulação de pacotes e baixa perda de precisão, porém, trata-se de um simulador de difícil implementação, sobretudo quando se deseja simular modalidades de redes ou protocolos que exigem uma modelagem específica de seus comportamentos na rede.

Com relação as redes P2P, é comum haver simuladores que tratam apenas da rede em nível *overlay*. Simuladores que implementam apenas essa rede virtual de vizinhanças dos *peers*, tendem a ser menos precisos que aqueles que implementam todas as camadas da pilha de protocolos, desde o enlace até a aplicação *peer-to-peer*.

A precisão da simulação avalia, portanto, o quanto a simulação se aproxima do mundo real, verificando se as informações como atraso de pacotes, descarte, etc., são dinâmicas e geradas de acordo com o comportamento instantâneo da rede ou se são apresentadas por equações que simplesmente utilizam parâmetros da rede para calcular informações desse tipo [5].

- Tempo de simulação

  O tempo de simulação verifica em quanto tempo de execução o simulador é capaz de oferecer os resultados da simulação. Os principais fatores que influen-

ciam na velocidade do simulador são a linguagem em que ele foi programado e o nível de detalhes da simulação. Como é sabido, linguagens totalmente compiladas e que apresentam um eficiente gerenciamento de memória possuem melhor desempenho em sua execução. Por sua vez, o tempo de simulação é inversamente proporcional ao nível de realidade do simulador. Quanto maior a precisão do comportamento da rede, maior será o número de eventos gerados, e por sua vez maior a necessidade de processamento do simulador. Dessa forma, o desafio é tentar manter um bom desempenho do simulador em relação ao tempo, sem sacrificar a precisão das informações que ele é capaz de gerar.

- Escalabilidade

  Entende-se por um sistema de simulação escalável, aquele que permite um crescimento do tamanho da rede, seja no número de nós, seja no tráfego que ela gera, sem haver comprometimento de seu desempenho. Normalmente esse desempenho se refere ao tempo necessário para concluir a simulação proposta. Deseja-se que o aumento da complexidade da rede simulada não implique em tempo de simulação impraticável. A escalabilidade é um aspecto muito importante na simulação de grandes redes. Isso reflete diretamente nas redes P2P, que normalmente são apresentadas com uma grande quantidade de nós e intenso fluxo de dados circulando entre eles. Quanto maior a quantidade de nós no sistema, maior a complexidade de gerenciamento e certamente maior o tráfego circulando dentro da rede. Esse problema é menos evidente quando o simulador abstrai algumas camadas da rede reduzindo processamento de pacotes, porém, reduzindo também a precisão da simulação.

- Robustez e Usabilidade

  Como todo sistema computacional, o simulador deve apresentar princípios de qualidade de *software*. Entre outros aspectos, espera-se dele robustez e usabilidade. É preciso ser consistente e não se deve dar margem a falhas. Deseja-se facilidade de manuseio e aprendizado, além de ser flexível às necessidades do

usuário e oferecer a ele variadas funcionalidades e comportamentos, como por exemplo permitir que o usuário insira eventos além daqueles gerados automaticamente pelo simulador, como falhas, entradas e saídas de nós em tempos específicos.

Deve-se verificar também a forma como o usuário comum interage com o simulador. Se por meio de *scripts* de simulação usando editores comuns, ou se há linguagem e ambientes específicos para a produção de cenários. Deve-se saber se é possível utilizar padrões de linguagens de simulação ou se é permitido importação e exportação desses *scripts*.

- Informações estatísticas

  Outro aspecto fundamental é saber como o simulador oferece os resultados gerados pela simulação. Se são oferecidos arquivos de *tracing* dos eventos ocorridos, quais as informações disponibilizadas e se o usuário é capaz de ele próprio produzir com facilidade as infomações estatísticas das quais ele necessita.

  É importante também verificar se o simulador oferece visualizações gráficas ou animações que demonstram exatamente o cenário, comportamento e resultados da simulação que o usuário propôs.

- Documentação

  A disponibilidade de documentação tanto em nível de usuário como em nível de desenvolvedor é fundamental para o rápido aprendizado e contribuição com o desenvolvimento e evolução do código do simulador. É preciso que haja uma documentação organizada, completa e de qualidade. Essa documentação abrange manuais, artigos publicados, apresentações, exemplos de simulação, documentação da API, códigos bem comentados, além de suporte *online* atráves de fóruns e listas de discussões por e-mail.

Detalhamos a seguir, três simuladores de redes diretamente ligados a nosso trabalho: o Network Simulator 2, já bastante difundido no meio acadêmico; seu su-

cessor Network Simulator 3 e o simulador FFM (*Fluid Flow Model*), que mescla a simulação em nível de pacotes com a simulação em nível de fluidos.

2.2.5.1 Network Simulator 2

O NS-2 é um simulador de redes orientado à eventos, trabalha no nível de pacotes e oferece suporte à vários protocolos das camadas do modelo TCP/IP [40, 41]. Por seu longo tempo de uso, se tornou um padrão em simulação de redes na comunidade acadêmica.

O simulador é implementado usando linguagem compilada, C++, e linguagem interpretada, OTcl (*Object-oriented Tool Command Language*). Foi projetado dessa forma, pois desejava-se evitar a recompilação do simulador quando fossem feitas alterações em seus parâmetros. O escalonador de eventos e os principais componentes de rede são implementados em C++, mas também são configuráveis por meio do OTcl. Este, atua como interface de configuração e comando para o usuário, permitindo definir a topologia da rede, tráfego e eventos em geral [42]. Atualmente, o uso da liguagem de *script* é questionável, uma vez que deseja-se aumentar a escalabilidade do simulador, e hoje em dia, a necessidade de recompilar seu código a cada modificação da rede não se torna um problema.

Sua modularidade e relativa facilidade de implementação, permitiu a experimentação de diversos protocolos, dentre eles, aplicações *peer-to-peer* como Gnutella [11] e Bittorrent [25].

O Network Simulator 2 produz, como saída, arquivos de *tracing*, que apresentam os principais eventos ocorridos durante a simulação, permitindo que esses dados sejam posteriormente analisados pelo usuário. Permite também gerar animações para o interpretador NAM que representam os eventos do cenário simulado facilitando o entendimento de seu comportamento.

2.2.5.2 Network Simulator 3

O NS-3 é o sucessor do Network Simulator 2, porém, teve seu código totalmente reescrito inviabilizando, inclusive, que protocolos do simulador anterior possam ser reaproveitados. Diferentemente de seu antecessor, foi implementado apenas em linguagem compilada, C++. Elimina-se portanto, o problema de haver uma linguagem interpretada, melhorando seu desempenho. Opcionalmente, o usuário pode usar linguagem de *script* Python para configurar parâmetros de sua simulação caso queira evitar recompilações de código [6, 24].

O NS-3 é implementado visando as necessidades dos pesquisadores da área. O projeto encoraja a participação e contribuição da comunidade científica, e para isso, seu código foi projetado visando sua expansibilidade.

Sua equipe de desenvolvimento participa do mesmo projeto que deu origem ao NS-2, além de contar com milhares de pesquisadores de toda parte do mundo ativamente envolvidos em novas propostas de implementação de protocolos para serem incorporados ao NS-3.

É um simulador orientado a eventos, e por trabalhar no nível de pacotes, simula todas as camadas da pilha de rede, apresentando informações dinâmicas com relação ao descarte de pacotes, falhas, atrasos etc. Busca simular fielmente o comportamento e as propriedades da rede, ainda mais próximo da realidade que seu antecessor.

O NS-3 simula cada nó como um computador real, simulando as interfaces e *drivers* fundamentais da pilha TCP/IP. É capaz de simular topologias de rede CSMA, assemelhando-se às redes *Ethernet*, e topologias para conexões ponto a ponto. Atualmente implementa apenas filas *DropTail*. Diferentemente do NS-2, faz o controle de todo o espaço de endereçamento IPv4/IPv6. Na camada de transporte, o protocolo TCP é baseado no projeto GTNets [43], sendo uma implementação ainda mais completa de sua máquina de estados que no NS-2. Atualmente, suporta apenas controle de congestionamento Tahoe. NS-3 fornece também todo o suporte às APIs (*Application Programming Interface*) de socket da internet (Posix) com interfaces para ambos os protocolos de transporte, TCP e UDP.

O sistema de traces do NS-3 foi melhorado, permitindo que o usuário personalize as informações que ele deseja visualizar, além de incorporar um *framework* de análises estatísticas. É também capaz de gerar saídas em formato padronizado pcap (*packet capture*) para que possam ser interpretadas por outros softwares com esse fim. O NS-3 pode também ser integrado a sistemas reais atuando em modo de emulação, consumindo e emitindo pacotes para dispositivos reais.

Apesar de ainda incompleta e exigir bastante esforço do pesquisador, a documentação do Network Simulator 3 é vasta e robusta. O simulador conta também com suporte online através de listas de discussões com participação ativa de seus desenvolvedores.

Alguns estudos já foram realizados comparando o desempenho do Network Simulator 3 com demais simuladores de redes. Dentre eles, o NS-3 foi o que apresentou melhores resultados com relação ao seu tempo de simulação e consumo de memória [23].

2.2.5.3 FFM

O FFM (*Fluid Flow Model*) [4] é um simulador híbrido, mesclando um simulador de fluidos ao tradicional simulador de pacotes NS-2. O FFM ainda é um simulador limitado em relação a quantidade de protocolos que ele suporta. Atualmente permite apenas a criação de cenários com roteadores AQM e tráfegos sob conexões TCP justamente pela dificuldade de gerar novas modelagens em fluidos para o comportamento de outros protocolos. Como apresenta código fonte aberto, pode ser estendido, permitindo trabalhos futuros sobre ele.

A rede de fluidos simulada é representada por um simples nó do simulador de pacotes. Neste nó, estão definidos todos os enlaces, roteadores e filas da rede de fluidos.

Ao atravessar esse nó especial, os pacotes são inicialmente sincronizados e transformados para fluxos de fluidos por meio de enlaces também definidos em nível de implementação. As equações do modelo de fluidos, que determinam o

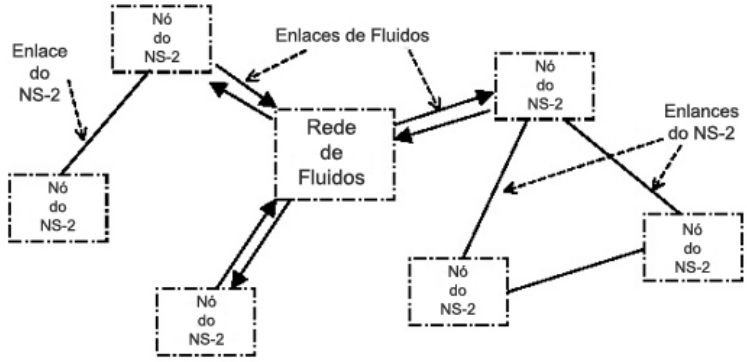


Figura 2.10: Modelo de fluidos integrado ao NS-2

comportamento e desempenho da rede, são resolvidas numericamente pelo método *Runge-Kutta* [44]. Por fim, para que sejam lançados adequadamente de volta à rede de pacotes, os dados são ressincronizados com o relógio do NS através dos enlaces de fluidos ligados aos nós comuns do Network Simulator [4].

2.3 O Protocolo TCP

O TCP (Transmission Control Protocol) é o protocolo de transporte adotado pelas redes *peer-to-peer*. Por este trabalho envolver vários conceitos aplicados a esse protocolo, viu-se a necessidade de abordar o assunto a fim de entender rotinas de estabelecimento de conexão, controle de congestionamento, *timeouts* etc., que por sua vez serão adotados em parâmetros de configuração dos simuladores utilizados nos experimentos [45, 46].

O TCP é um protocolo orientado a conexão e que fornece confiabilidade, retransmitindo datagramas perdidos e reorganizando-os na sequência correta. Para que o serviço TCP funcione, é necessário que uma conexão seja explicitamente estabelecida entre sockets localizados em cada uma das máquinas. Cada socket é constituído por um endereço IP mais um número de porta.

Toda conexão TCP é *full-duplex* e consiste em um fluxo de dados e não um fluxo de mensagens, ou seja, como a unidade entendida é o *byte*, as delimitações das mensagens não têm significado durante as transmissões. Os dados são escritos em segmentos delimitados pelo tamanho máximo do pacote IP e pela MTU (*Maximum*

Transfer Unit) da rede.

O funcionamento do TCP consiste no envio desses segmentos de dados pelo transmissor e o recebimento de confirmações de chegada enviadas pelo receptor contendo o próximo número de sequência (*byte*) que se espera receber.

As conexões TCP são estabelecidas utilizando *handshakes* de três vias. O host servidor mantém seu socket em estado *listen*, aguardando recebimento de pedidos de conexão. O lado cliente, por sua vez realiza o pedido de conexão informando as configurações da conexão que deseja estabelecer. Por fim, se o servidor aceita a conexão, um segmento de confirmação será retornado.

Uma das principais tarefas do TCP é o controle de congestionamento da rede. Ocorre congestionamento quando a carga na rede é superior a capacidade da mesma. Cabe ao TCP detectar essa ocorrência e executar procedimentos para solucionar o problema.

Todos os algoritmos TCP da Internet presumem que a ocorrência de *timeouts*, ou seja, a expiração do RTT (*Round Trip Time*) na entrega de segmentos, se deve à existência de congestionamento na rede, uma vez que para as redes cabeadas, nos dias de hoje, é rara a perda de pacotes por erros de transmissão [46]. O RTT é uma estimativa de tempo gasta para que um segmento seja enviado e confirmado. É o tempo de ida e de volta da mensagem. Para cada segmento enviado, esse temporizador é iniciado. Se o temporizador expirar antes da confirmação do segmento, o TCP interpreta a situação como a ocorrência de uma perda.

Durante a transmissão de dados, a regra básica que define o gerenciamento da janela TCP, é que o tamanho máximo do segmento que o emissor pode enviar não ultrapasse o espaço máximo anunciado e disponível no *buffer* do receptor. No entanto, quando o fator limitante não é o nó receptor, mas a rede interna, o congestionamento ainda pode ocorrer. Dessa forma, cada transmissor mantém duas janelas: a janela fornecida pelo receptor (AWND - *Advertised Window ou Announced Window*) e uma segunda janela, definida por ele com base no comportamento da rede, a janela de congestionamento (CWND - *Congestion Window*). O número de bytes que o

transmissor pode enviar é o valor mínimo entre as duas janelas.

Encontrar o valor de AWND é fácil, uma vez que é o próprio receptor quem anuncia sua capacidade atual, porém, calcular a CWND não é tão simples. Isso, porque o transmissor precisa conhecer a capacidade da rede, e isso envolve os diversos equipamentos a ela conectados. Além disso, a capacidade desses equipamentos pode variar no decorrer do tempo, havendo a necessidade de essa variável ser atualizada frequentemente. Para realizar a descoberta da janela de congestionamento são utilizados vários algoritmos como: *Slow-Start*, *Congestion Avoidance*, *Fast Retransmit* e *Fast Recovery*.

- *Slow start*: O algoritmo *slow start* corresponde a fase em que, após o estabelecimento da conexão, o transmissor ajusta a janela de congestionamento ao tamanho do segmento máximo em uso na conexão e a cada segmento confirmado antes de ocorrer um *timeout*, o transmissor incluirá o número de bytes de um segmento na janela de congestionamento. Ou seja, na prática, a cada rajada confirmada, duplica-se a janela de congestionamento mantendo-se seu crescimento exponencial até que ocorra um *timeout*. Quando isso ocorre, o parâmetro de congestionamento *ssthresh* (limitante) é atribuído à metade da janela de transmissão atual, e a janela de congestionamento é reinicalizada para um tamanho de segmento máximo, reiniciando o processo até que o *ssthresh* seja alcançado. Todas as implementações do TCP devem ser compatíveis com ele.

- *Congestion Avoidance*: A fase de *congestion avoidance* é iniciada quando a janela de congestionamento atinge seu limitante (*ssthresh*) e a partir de então, o seu crescimento deixa de ser exponencial e passa a ser linear, ou seja, aumenta de 1 segmento a cada RTT.

- Fast Retransmit: O algoritmo fast retransmit é capaz de detectar perdas antes de ocorrência do *timeout*. Isso ocorre porque quando o TCP o adota, é capaz de interpretar a recepção de 3 *ACKs* duplicados como a perda do segmento, retransmitindo-o. Três *ACKs* duplicados significam, três confirmações de seg-

mentos já enviados com o valor do campo *acknowledgement number* iguais. Um ou dois *ACKs* duplicados são considerados segmentos fora de ordem, mas três ou mais é um forte indício de descarte [45].

- Fast Recovery: O algoritmo fast recovery, por sua vez consiste na execução do *congestion avoidance* em vez do *slow start* logo após a execução do Fast Retransmit.

Várias versões do TCP surgiram fazendo a combinação e ajustes desses algoritmos. A primeira versão do TCP era chamada de Tahoe e utilizava somente os algoritmos *Slow-Start* e *Congestion Avoidance* para controle de congestionamento. Havia excessiva demora de detecção de perdas em virtude da necessidade de aguardar o *timeout* expirar. Implementações mais recentes do mesmo TCP Tahoe inseriram o *fast retransmit* permitindo que a detecção de perdas também fossem identificadas através de *acks* duplicados [47].

O TCP *Reno*, por sua vez, já foi concebido com a fase de fast retransmit. Diferentemente do Tahoe, quando a perda por *ack* duplicado é detectada, a janela de congestionamento não é reduzida para 1 segmento, entrando em ação o *fast recovery*. Quando o *ssthresh* é reduzido para a metade da janela de transmissão, a janela de congestionamento também é reduzida para o mesmo valor. TCP *Reno* funciona bem quando não há muitas perdas de pacotes dentro de uma mesma rajada, caso contrário, seu desempenho reduz e se assemelha ao do TCP Tahoe.

O TCP *New-Reno* é uma adaptação do *Reno* onde a janela de congestionamento não é reduzida múltiplas vezes quando há diversas perdas em uma mesma rajada da janela de transmissão. Esse diferencial melhora significativamente seu desempenho em relação ao *Reno*.

Capítulo 3

Metodologia

3.1 Proposta de trabalho

A simulação de grandes redes em nível de pacotes é sempre um desafio, uma vez que o grande fluxo de dados e a manipulação de uma grande quantidade de nós exige um eficiente gerenciamento de memória e uma alta capacidade de processamento. Esse cenário é facilmente identificado na simulação de redes *peer-to-peer* de compartilhamento de arquivos, o que faz delas um bom cenário de testes para a avaliação de novos simuladores que prometem maior escalabilidade de seus recursos.

Como visto anteriormente, o Network Simulator 3 é um novo simulador de redes, sucessor do NS-2, e que de acordo com trabalhos que realizaram suas primeiras avaliações, embora orientado a eventos e em nível de pacotes, promete se destacar por sua escalabilidade diante de cenários mais complexos [23].

Este trabalho se propõe a avaliar o desempenho do NS-3 diante da simulação de grandes redes, onde por sua vez, as redes *peer-to-peer* se encaixam bem nesse perfil. A fim de explorar a complexidade desses cenários, deseja-se realizar tais testes diante da simulação de ambientes P2P. Como o NS-3 ainda não possui aplicações *peer-to-peer* implementadas em seu *framework*, torna-se necessário, de imediato, o desenvolvimento de tal protocolo como parte do simulador. Para observar as diferenças de comportamento entre a simulação em nível de pacotes e a simulação em nível de

fluxos, no que se refere ao Network Simulator 3, o mesmo protocolo foi implementado em ambos os níveis de abstração.

3.1.1 Apresentação da metodologia

A existência de estudos anteriores [25] que implementaram Bittorrent para NS-2 tanto em nível de pacotes como em nível de fluxos favorece a escolha do Bittorrent como protocolo a ser desenvolvido no NS-3, pois a versão do mesmo para o NS-2 nos possibilita a validação e comparação de resultados com nossa implementação.

Como o código do NS-3 foi totalmente reescrito, não foi possível reaproveitar a implementação do Bittorrent escrita para NS-2. Além do protocolo de aplicação propriamente dito, foi necessário preparar todas as rotinas para a comunicação com as camadas inferiores da pilha TCP/IP através do uso de sockets, criação de novos cabeçalhos de pacotes, manipulação de envio e recebimento de mensagens e tratamento de erros em nível de aplicação.

A realização dos experimentos e implementações do módulo Bittorrent para o NS-3 foram realizadas utilizando a versão 3.7 disponível durante o desenvolvimento deste trabalho. Por sua vez, a versão do NS-2 adotada foi a 2.30, por mais se aproximar de compatibilidade com o módulo Bittorrent para esse simulador.

O processo de experimentação iniciou-se diretamente com a implementação do Bittorrent nos níveis de pacotes e fluxos. No entanto, os primeiros resultados obtidos mostraram-se divergentes entre o NS-2 e o NS-3. Com isso, primeiramente fez-se necessário realizar um grupo de experimentos adicionais para definir ajustes de configuração de ambos os simuladores a fim de obter-se resultados equivalentes. Entre outros ajustes, foi preciso definir igualmente em cada um deles, os principais parâmetros do protocolo TCP. Essa atividade foi realizada criando-se um cenário de rede simples e estudando os principais parâmetros de configuração aplicáveis até que ambas as simulações estivessem afinadas entre si.

Depois de concluir as implementações do simulador Bittorrent em nível de pacotes e em nível de fluxos, pôde-se observar as diferenças de desempenho em

ambos os níveis de abstração, seja com o aumento de tráfego, seja com o aumento do número de nós. Foi possível ainda, observar as questões relacionadas à precisão das informações obtidas em ambas as modalidades de simulação aplicando situações de congestionamento ou perdas de pacotes e consequentemente intensa variação da janela de transmissão do TCP.

Os resultados apresentados após as simulações do protocolo Bittorrent levaram à execução de novos experimentos que justificassem o desempenho de ambos os simuladores. Esses experimentos testaram a capacidade do NS-3 de gerenciar uma grande quantidade de nós e um intenso tráfego de dados. Criou-se então, mais uma vez, um simples cenário e utilizou-se como base para estudo um dos exemplos de simulação nativos do próprio NS-3 a fim de evitar qualquer erro de manipulação do simulador.

3.1.2 Métricas e Medições

Os testes de medição de desempenho se concentraram na variação da quantidade de pacotes circulando na rede e na variação da quantidade de *peers* em função do tempo, em segundos, necessário para que o processamento de toda a simulação fosse concluído. De acordo com o tipo de teste de precisão executado, estes, se resumiram em observar o comportamento do protocolo testado em função do tempo simulado dentro do simulador. Outra situação adotada, foi a medida do tempo (simulado) necessário para a conclusão do *download* de um determinado arquivo por um *peer* ou por todos os *peers* da rede.

Quando realizados testes de medição de desempenho, apenas uma instância dos experimentos foi colhida, uma vez que não há eventos de aleatoriedade que provoquem variação dos resultados obtidos, tornando-os bastante próximos uns dos outros.

Quando são realizados testes de precisão, apesar de o NS-3 ser um simulador determinístico, as diferentes sementes de geração de dados aleatórios, quando variadas, geram diferentes resultados para cada instância de experimentos. Além do mais, o próprio protocolo Bittorrent, quando simulado, apresenta aspectos de aleato-

riedade durante a execução de algumas de suas políticas. Neste trabalho, cada um dos experimentos de precisão são repetidos dez vezes, utilizando sementes geradoras distintas, para então, ser obtida uma média aritmética dos diferentes resultados. O intervalo de confiança no nível 95% é calculado a fim de verificar o intervalo de variação dos mesmos.

Depois de executados todos os testes, os resultados são discutidos e apresentados em tabelas ou em gráficos construídos com o auxílio da ferramenta Gnuplot [48].

3.2 Ajustando parâmetros do TCP

Durante a execução de nossas primeiras simulações, verificamos que alguns parâmetros do protocolo TCP utilizado pelo Bittorrent estavam com valores diferentes em cada um dos simuladores. A fim de ajustar igualmente esses valores e nos certificarmos de que ambos estão em sintonia e precisos entre si, definimos um cenário simples como o da Figura 3.1 e a partir dele realizamos uma pequena bateria de testes.

Alguns desses parâmetros ajustados foram a configuração da janela máxima de transmissão para 32 pacotes, o tamanho máximo do segmento TCP para 1024 *bytes*, a definição de um pacote de confirmação *ack* enviado para cada segmento recebido, e a alteração do modelo TCP para TCP Tahoe.

Figura 3.1: Cenário adotado para simples transferência de dados.

Com relação aos dois últimos parâmetros citados, o NS-3 estava configurado para enviar um pacote de confirmação *ack* a cada 2 segmentos TCP recebidos. Optamos então, por deixá-lo da mesma forma que no NS-2: um pacote de confirmação a cada segmento recebido com sucesso. O TCP Tahoe foi adotado porque este é o único modelo TCP suportado pelo NS-3. Originalmente, o módulo Bittorrent para NS-2

utilizava TCP *New-Reno*.

O cenário da Figura 3.1 foi baseado em um dos exemplos já disponibilizados nativamente pelo NS-3 e consiste de dois hosts conectados por um nó roteador, onde um deles é emissor e o outro receptor. O tráfego trocado entre eles é originado do envio de um arquivo de 100MB através de uma conexão TCP. Os canais de comunicação têm capacidade de 1Mbps e retardo de 75ms até o roteador, totalizando 150ms. Esses valores foram mantidos fixos a fim de facilitar a comparação de resultados entre ambos os simuladores. As capacidades das filas nos nós são de 100 pacotes, e durante a simulação foi induzido uma falha de 0,02% na transmissão dos dados a fim de facilitar a visualização do comportamento das janelas de transmissão do TCP, que por sua vez podem ser observadas no gráfico da Figura 3.2.

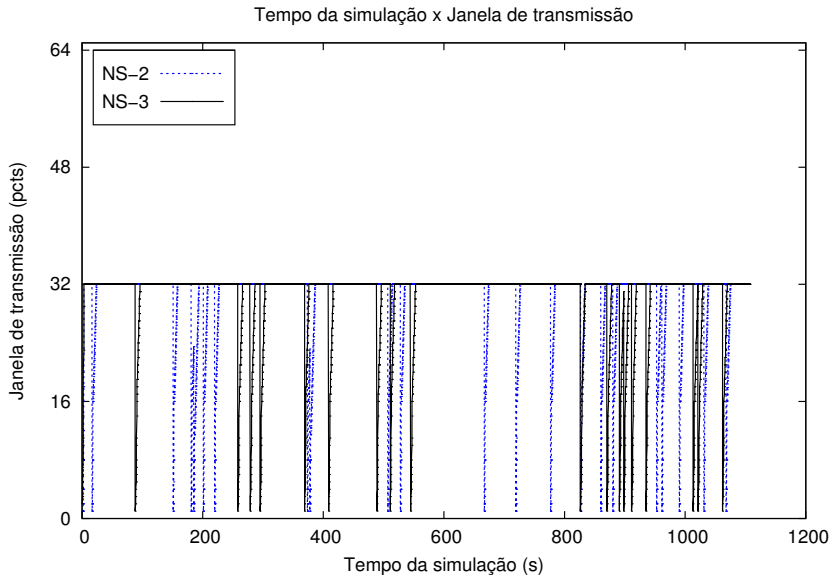


Figura 3.2: Monitoramento da janela de transmissão do TCP.

A Figura 3.3 consiste na mesma simulação da Figura 3.2, e mais uma vez representa a janela de transmissão em função do tempo simulado, porém, ampliamos sua imagem no intervalo entre 850 e 1000 segundos a fim de visualizar exatamente a atuação do protocolo TCP Tahoe. Nela, a cada detecção de perda de segmento, o TCP interpreta como a existência de um congestionamento reduzindo a janela para 1 e atribuindo o novo valor do *ssthresh* para a metade da janela naquele instante. Ao atingir esse limitante, a evolução da janela passa a ser linear até que se atinja o limite

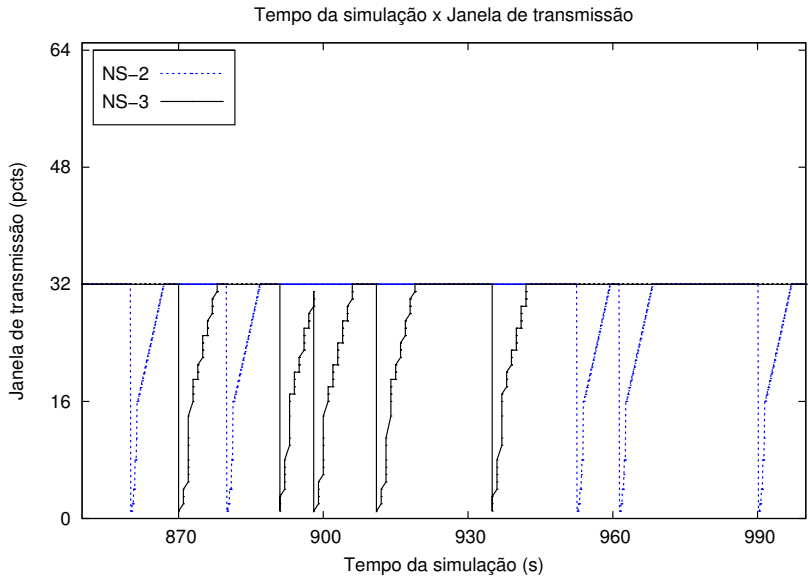


Figura 3.3: Trecho ampliado do monitoramento da janela de transmissão do TCP.

máximo de 32 pacotes.

Através desses dois gráficos pudemos nos certificar que o protocolo TCP está agindo igualmente em ambos os simuladores, o que não estava ocorrendo antes dos ajustes que precisamos realizar para dar prosseguimento aos nossos experimentos.

3.3 Arquitetura do simulador Bittorrent

A implementação do protocolo Bittorrent no NS-3 foi realizada em módulos, obedecendo a mesma estrutura de diretórios dos outros protocolos nativos do simulador. A Figura 3.4 apresenta essa estrutura e componentes do módulo Bittorrent em nível de pacotes. Uma estrutura semelhante é utilizada para representar o módulo Bittorrent em nível de fluxos. O arquivo simulacao.cc corresponde ao *script* de configuração dos parâmetros globais da simulação proposta em cada experimento, definindo o cenário, aplicações, registros de traces, etc. Por sua vez, dentro dos diretórios Bittorrent e *Tracker* estão as implementações propriamente ditas dessas duas classes. Ambas são implementadas como subclasse de *Application*, onde por sua vez, a aplicação Bittorrent depende da aplicação *Tracker*. Os *helpers* do Bittorrent definidos no diretório de

mesmo nome, são interfaces em alto nível que facilitam o uso das classes e métodos para o usuário final nos *scripts* de configuração da simulação.

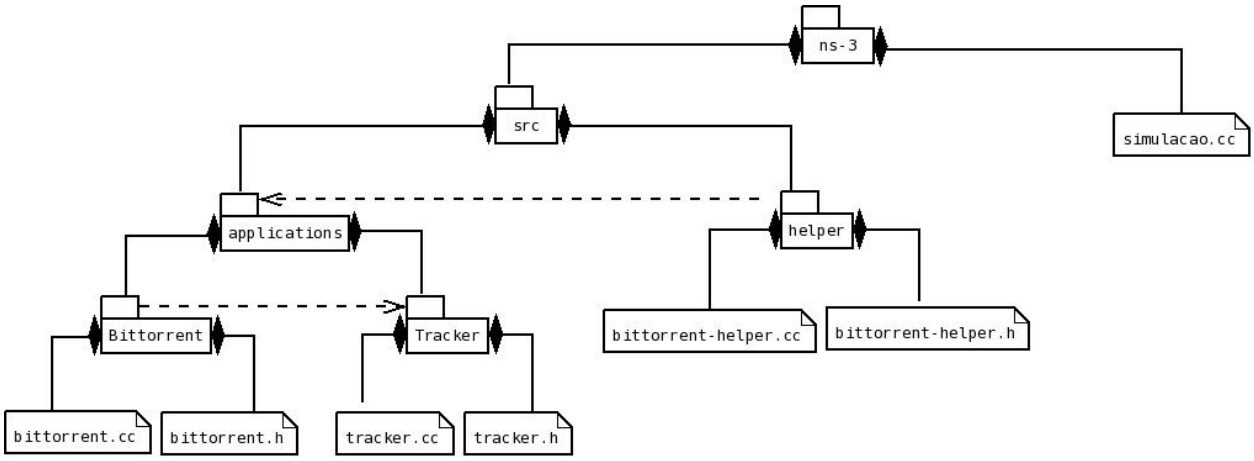


Figura 3.4: Diagrama de componentes para o módulo Bittorrent em níveis de pacotes e fluxos.

Na versão de nosso simulador em nível de pacotes, todos os recursos do NS-3 são aproveitados. Como explicado anteriormente, todas as camadas da pilha de protocolos são simuladas, assemelhando-se mais a uma rede real.

Na versão do simulador em nível de fluxos, as camadas de rede e de transporte são ignoradas de forma que as mensagens são entregues diretamente a seus destinatários. Não há, portanto, simulação de filas, riscos de congestionamentos nem descarte de pacotes. O controle é realizado apenas no nível da aplicação e, naturalmente, são respeitadas as capacidades máximas de transferências de cada *peer* mantendo-se o controle da quantidade de pacotes enviados ou recebidos.

A Figura 3.5 representa o diagrama de estados do módulo Bittorrent implementado. Todos os tipos de mensagens e funcionalidades implementadas são baseados nas especificações da versão 1.0 do protocolo Bittorrent [31].

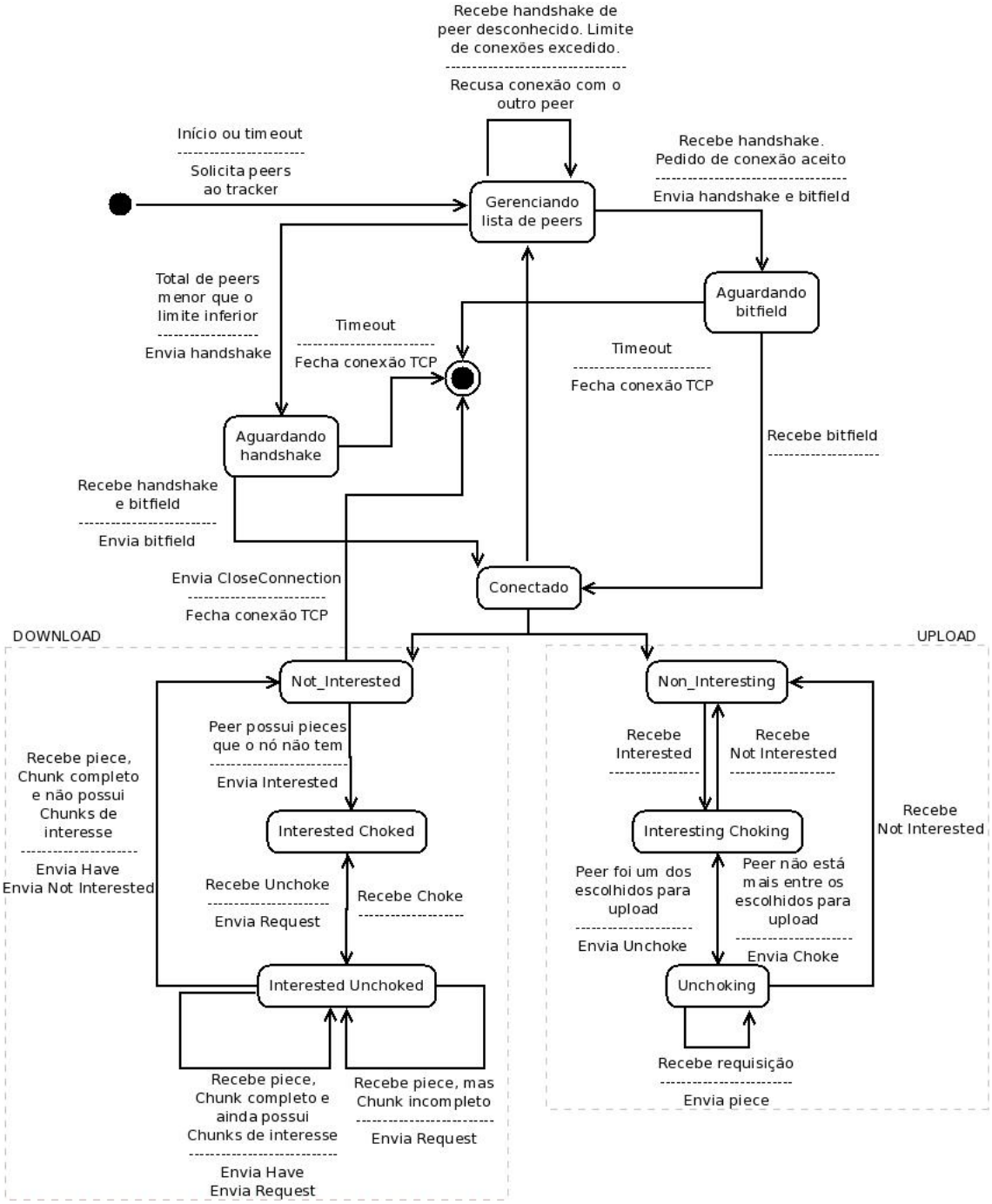


Figura 3.5: Diagrama de estados do módulo Bittorrent implementado.

Quando o nó entra na rede P2P, o mesmo solicita *peers* ao *tracker* ou aguarda receber pedidos de conexão através de mensagens *handshake*. Se o nó recebe pedido de conexão, responde com *handshake* e envia sua lista de *chunks* através da mensagem do tipo *bitfield*, passando a aguardar a lista do parceiro. Se é o nó quem inicia o pedido de conexão, então ele próprio envia o *handshake* e fica no aguardo da resposta da confirmação da conexão. Quando há recusa de conexão ou

expira-se o *timeout*, a conexão TCP é encerrada, caso contrário, estará estabelecida a conexão. A partir de então, o nó passa a possuir duas linhas de estados em paralelo: uma para realização de *downloads* e a outra referente aos *uploads*. Se o nó tem interesse em algum *chunk* do *peer*, envia mensagem do tipo *Interested* e aguarda receber mensagem de *Unchoke* para iniciar as requisições e *download* até ser bloqueado novamente com mensagem de *Choke*. A cada *chunk* completo no *download*, o nó deve informar o novo *chunk* obtido a todos seus *peers* através de mensagem *Have*. Caso não tenha mais interesse no *peer*, uma mensagem de *Not Interested* é enviada. Durante as fases de *upload*, o processo consiste em receber mensagens de *Interested* ou *Not Interested* a fim de saber se o *peer* está ou não interessado em algum *chunk* de sua lista. As requisições e envio de *pieces* para o *peer* são liberados quando mensagens *Unchoke* são emitidas, e bloqueados quando mensagens de *Choke* são enviadas.

Um pseudocódigo dos principais métodos do protocolo Bittorrent de nosso projeto é apresentado no apêndice A deste trabalho, a fim de favorecer maiores detalhes a respeito de sua implementação.

3.4 Descrição do cenário

O cenário simulado nos testes (Figura 3.6) é conhecido como *Flash crowd* [25]. Trata-se de uma situação que explora ao máximo a capacidade de distribuição de recursos do Bittorrent. Uma única *seed* está inicialmente presente na rede e deve atender às solicitações de outros *N peers* que entram simultaneamente nela. À medida que cada *peer* completa seu *download*, ele permanece na rede até que todos os *peers* tenham obtido o arquivo desejado, quando então, finaliza-se a simulação.

Foi simulado um ambiente de rede onde cada nó representa tipicamente um usuário em sua residência iniciando uma aplicação Bittorrent. Cada *peer* tem uma capacidade de transmissão de 1Mbps. O modelo TCP adotado foi o Tcp-Tahoe, o tamanho máximo do segmento foi definido para 1024 bytes e as filas nos nós ado-

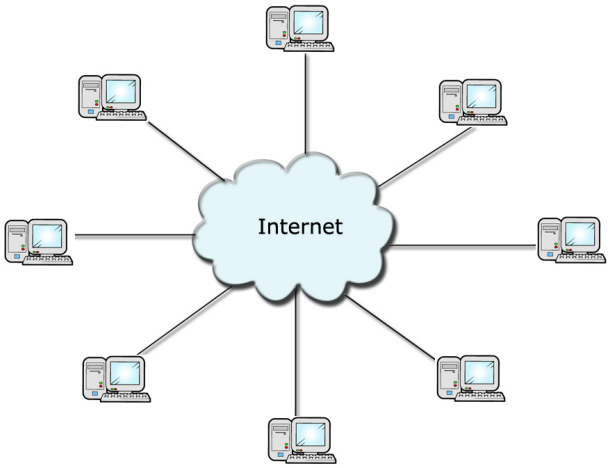


Figura 3.6: Cenário da rede *peer-to-peer* simulada.

tam política *DropTail*. A nuvem de Internet foi representada por um pequeno grupo de roteadores em estrela mas com taxa de 100Mbps e retardo de 0ms para evitar congestionamento no interior da rede. O retardo no canal de transmissão de um *peer* ao primeiro roteador da internet foi fixado em 75ms, totalizando 150ms de retardo dos *peers* entre si. Esses atributos de retardo, fila dos nós, capacidade de *download* e tamanho do segmento TCP, não se aplicam ao simulador em nível de fluxos, afinal, as mensagens são entregues diretamente ao destinatário e não há simulação de protocolo TCP.

Com relação aos parâmetros da aplicação Bittorrent, à maioria deles foram colocados valores padrões, como o intervalo de consulta ao *tracker* (300 segundos), intervalo de execução do algoritmo de checagem de *choking* (10 segundos), tamanho do *chunk* (256KB) e tamanho do *piece* (32KB).

O arquivo compartilhado tem tamanho de 100MB e o cenário *Flash crowd* foi utilizado inicialmente com 2 *peers*, depois 5, 10, 25, 50, 100, 150 e 200 *peers*. Cada instância de teste foi repetida dez vezes, cada uma delas com uma semente de geração de dados aleatória diferente, sendo obtida em seguida, a média aritmética de cada um desses grupos.

Capítulo 4

Experimentação e Resultados

4.1 Simulando Bittorrent

Concluídas as implementações do protocolo Bittorrent para NS-3 em nível de fluxos e em nível de pacotes, foi possível realizar os experimentos inicialmente propostos, abrangendo simultaneamente uma grande quantidade de nós e um grande fluxo de dados entre eles.

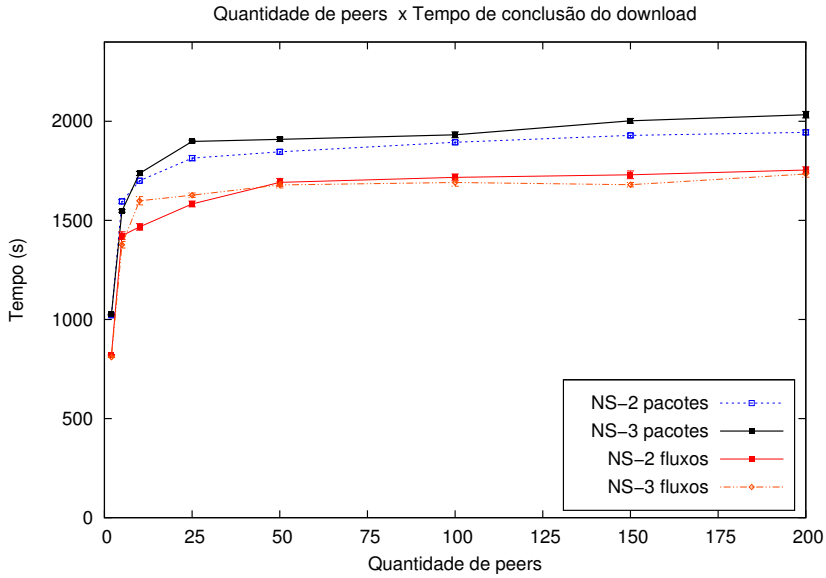


Figura 4.1: Avaliando precisão dos simuladores em nível de fluxos e em nível de pacotes.

A comparação do simulador Bittorrent para NS-3 com seu antecessor foi realizada utilizando o cenário descrito anteriormente. Os mesmos parâmetros de tes-

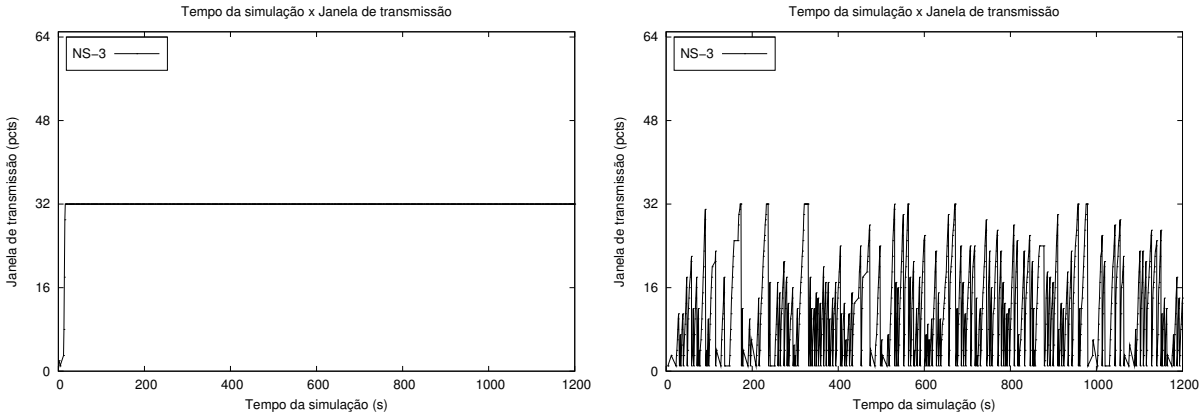


Figura 4.2: Variação da janela de transmissão x Tempo

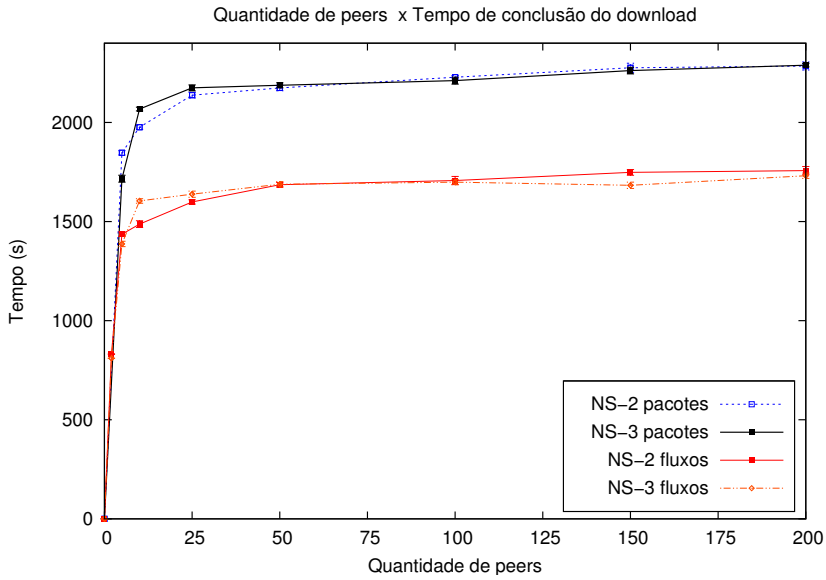


Figura 4.3: Avaliando precisão dos simuladores em nível de fluxos e em nível de pacotes quando ocorre variação da janela de transmissão.

tes foram submetidos aos 4 simuladores: aos simuladores de pacotes e de fluxos do NS-3, bem como aos simuladores de pacotes e de fluxos do NS-2. O confronto dos resultados pode ser visualizado no gráfico da Figura 4.1. Trata-se do tempo necessário para que todos os nós concluam o *download* do arquivo compartilhado em função da quantidade de *peers* na rede.

Neste primeiro experimento, aplicamos um cenário que favorece o simulador em nível de fluxos, onde de acordo com [25], as redes *peer-to-peer* apresentam uma característica particular, onde o gargalo, entre outros aspectos limitantes de processamento, estariam localizados nas bordas da rede, ou seja, nos *peers*.

Tabela 4.1: Tempo médio de conclusão de *download* do último *peer*

	5 peers	**25 peers**	**50 peers**	**100 peers**	**200 peers**
NS-2 Sem perdas (pcts)	1585s	1804s	1836s	1884s	1934s
NS-3 Sem perdas (pcts)	1557s	1908s	1919s	1941s	2043s
NS-2 Sem perdas (fluxos)	1423s	1583s	1692s	1717s	1754s
NS-3 Sem perdas (fluxos)	1377s	1627s	1679s	1691s	1734s
NS-2 Com perdas (pcts)	1846s	2137s	2174s	2228s	2285s
NS-3 Com perdas (pcts)	2116s	2174s	2187s	2211s	2289s
NS-2 Com perdas (fluxos)	1436s	1599s	1686s	1707s	1757s
NS-3 Com perdas (fluxos)	1387s	1638s	1688s	1699s	1731s

Definimos, dessa forma, um ambiente que não proporcione congestionamentos, onde há grande largura de banda nos roteadores centrais e não há perdas de pacotes nem descartes por transbordo de filas.

A Figura 4.2(a) nos mostra que, como pretendido, ao selecionarmos um par de *peers* de uma das instâncias de testes e monitorarmos sua janela de transmissão, não houve variação da mesma durante a simulação.

Como podemos perceber, portanto, nesse caso, a simulação em nível de fluxos foi capaz de gerar resultados relativamente próximos aos gerados pela simulação em nível de pacotes.

Na simulação seguinte, geramos um ambiente, em que de forma contrária a anterior, há uma intensa variação da janela de transmissão. A intenção é simular um pequeno congestionamento na rede proporcionando a atuação do controle de congestionamento do TCP. Para isso, inseriu-se um modelo em que 0,5% dos pacotes são perdidos na rede.

A Figura 4.2(b) aponta a janela de transmissão em função do tempo de simulação de um par de *peers* demonstrando a ocorrência da variação proposta.

Como podemos perceber na Figura 4.3 e na tabela 4.1, quando há variação da janela de transmissão TCP, os resultados no simulador de fluxos mantêm-se muito próximos ao da simulação anterior, afinal de contas, esse tipo de simulador não implementa diversos eventos típicos do protocolo TCP, como detecções de perdas por

ACKs duplicados, estouros de *timeout*, políticas de controle de congestionamento, etc. O simulador de fluxos, portanto, não foi capaz de acompanhar os resultados do simulador em nível de pacotes, que por sua vez, em virtude da variação das janelas de transmissão, teve um maior aumento no seu tempo necessário para a conclusão do *download* na simulação.

Tabela 4.2: Tempo de processamento para conclusão da simulação

	5 peers	**25 peers**	**50 peers**	**100 peers**	**200 peers**
NS-2 (pcts)	00:00:11	00:02:15	00:07:05	00:19:28	00:55:13
NS-3 (pcts)	00:01:39	00:15:09	00:41:54	02:19:32	09:36:10
NS-2 (fluxos)	00:00:01	00:00:17	00:00:56	00:02:14	00:05:37
NS-3 (fluxos)	00:00:01	00:00:05	00:00:12	00:00:26	00:00:52

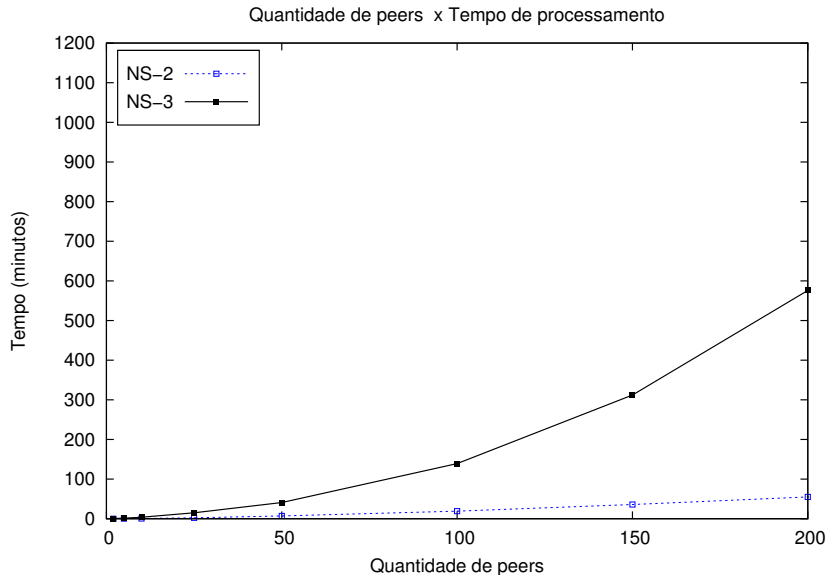


Figura 4.4: Avaliando desempenho dos simuladores em nível de pacotes.

Com relação ao desempenho de cada nível de simulação empregado, como esperado, os simuladores em nível de fluxos foram capazes de apresentar resultados muito mais rapidamente que os simuladores em nível de pacotes. Esses resultados podem ser observados na tabela 4.2.

Analisando agora o desempenho individual de cada simulador, os resultados variaram de acordo com a técnica adotada. Os mesmos podem ser visualizados nos gráficos das Figuras 4.4 e 4.5. Para a simulação em nível de pacotes, o Network

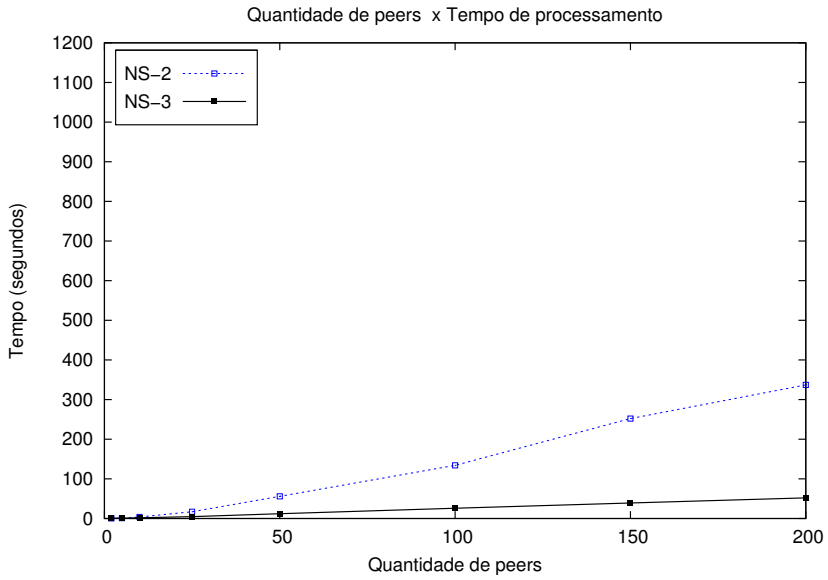


Figura 4.5: Avaliando desempenho dos simuladores em nível de fluxos.

Simulator 2 foi capaz de realizar as simulações propostas em tempo bem inferior ao do NS-3. O inverso ocorre na simulação em nível de fluxos. Nela, o Network Simulator 3 consegue apresentar resultados em menor tempo que o NS-2.

Observamos que com relação ao desempenho do NS-3 na simulação em nível de pacotes, os resultados foram contraditórios com o que se esperava de leituras de artigos anteriores [23]. Foi necessário, portanto, realizarmos um estudo a fim de investigar os motivos que levaram a tais resultados.

4.1.1 Analisando o desempenho do NS-3

Observou-se que as simulações em nível de pacotes, quando realizadas sobre o NS-3, estavam demorando exageradamente. Para isso, foram realizados alguns experimentos simplificados. O primeiro deles adotou o mesmo cenário da Figura 3.1, utilizado para os ajustes dos parâmetros do TCP. Trata-se de um nó emissor, outro receptor e um roteador fazendo a intermediação entre eles. Desta vez, porém, variou-se o tamanho do arquivo transmitido de 100MB a 500MB. O objetivo é reproduzir de forma simplificada a transferência de arquivos existente entre um par de *peers* durante a execução do protocolo Bittorrent simulado anteriormente.

Primeiramente, verificamos a precisão dos resultados de ambas as simulações

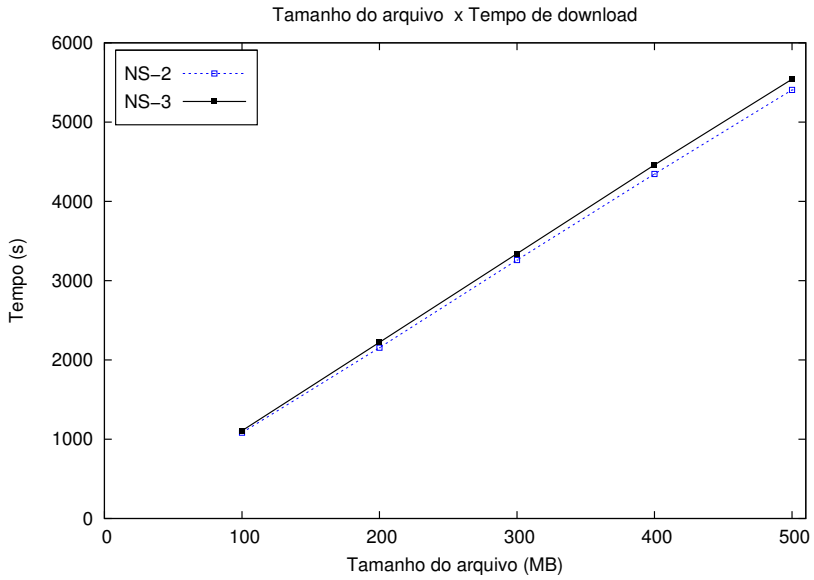


Figura 4.6: Gráfico de precisão da simulação com parâmetros ajustados.

para nos certificarmos de que estão realizando a mesma tarefa e gerando os mesmos resultados. Esses resultados, de fato precisos, podem ser vistos na imagem da Figura 4.6.

A Figura 4.7 e a tabela 4.5 nos mostram os valores obtidos neste experimento com relação ao desempenho de ambos os simuladores, ou seja, o tempo de processamento necessário para que cada um dos simuladores conclua toda a simulação que lhe foi proposta. O tempo de cada simulação foi colhido utilizando-se exatamente o mesmo ambiente computacional. Ou seja, foram utilizados um mesmo hardware (processador *dual core* 4Ghz e 2GB de memória RAM) e um mesmo sistema operacional (Linux *kernel* 2.6.21). Durante a simulação, além dos processos básicos e necessários de sistema, a simulação era o único programa em execução.

Tabela 4.3: Tamanho do tráfego TCP versus tempo de processamento dos simuladores.

	100MB	**200MB**	**300MB**	**400MB**	**500MB**
Tempo de processamento NS-2	00:03s	00:08s	00:11s	00:16s	00:19s
Tempo de processamento NS-3	00:16s	00:51s	02:07s	03:54s	06:09s

O que podemos observar é que de fato, o NS-3 precisou de um tempo bem superior que o NS-2 e com evolução não-linear para a conclusão das simulações submetidas. Diante disso, selecionamos um novo experimento a ser realizado num

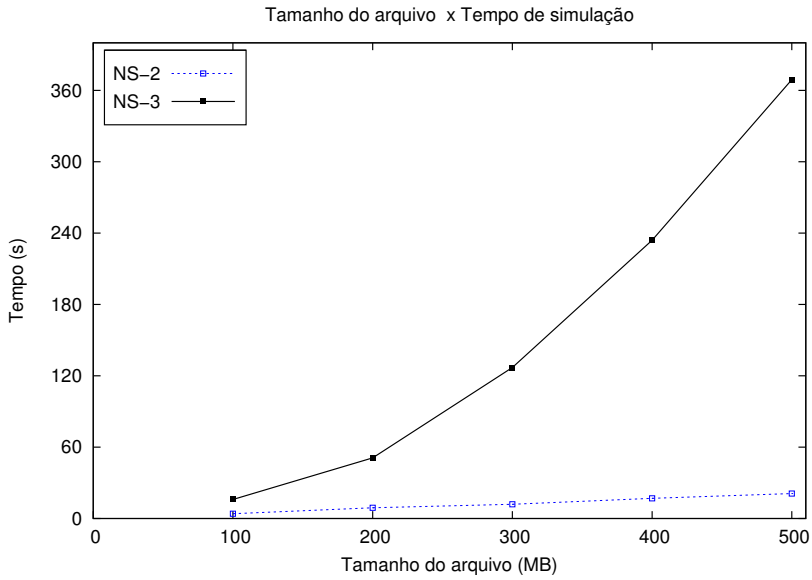


Figura 4.7: Gráfico de desempenho dos simuladores aplicando tráfego TCP.

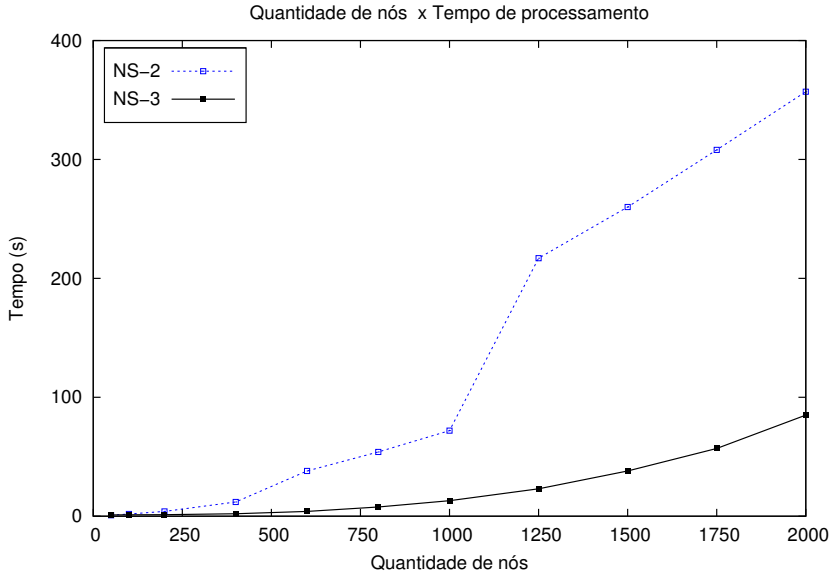


Figura 4.8: Avaliando desempenho dos simuladores em função da quantidade de nós na rede.

cenário semelhante ao que vinha sendo trabalhado, porém, desta vez, variamos a quantidade de nós na rede em topologia estrela e o tráfego circulante consistirá apenas de um pacote UDP de 1KB enviado por cada um dos nós para um outro nó específico, tendo este pacote que passar pelo nó roteador para chegar ao seu destino. O protocolo UDP foi utilizado a fim de mostrar que tais ocorrências não se limitariam ao protocolo TCP, embora o UDP seja capaz de evidenciar ainda mais os resultados destes experimentos devido a sua simplicidade e reduzida quantidade de mensagens

Tabela 4.4: Tempo de processamento da simulação versus quantidade de nós na rede.

	Tempo de processamento NS-2	**Tempo de processamento NS-3**
200 nós	4s	1,3s
400 nós	12s	2,1s
600 nós	38s	4s
800 nós	54s	8s
1000 nós	72s	13s
1250 nós	217s	23s
1500 nós	260s	38s
1750 nós	308s	57s
2000 nós	357s	85s
2500 nós	1438s	162s

em circulação.

A Figura 4.8 mostra o tempo de processamento de cada simulador em função da quantidade de nós na rede. Variamos essa quantidade de 50 a 2500 nós, e de fato conseguimos obter os mesmos resultados, em termos de desempenho, de trabalhos que se propuseram a avaliar o NS-3 [23]. Assim, da mesma forma, a tabela 4.4 mostra que o Network Simulator 3 é capaz de gerênciar com melhor desempenho, uma quantidade muito superior de nós que seu antecessor.

A fim de tentar explicar tais resultados contrastantes do NS-3, foi realizado um terceiro experimento em que o número de nós na rede é fixado em 2000, variando a quantidade de pacotes enviados por nó durante a simulação.

Podemos observar nos respectivos gráfico e tabela 4.9 e 4.5, que quando elevamos gradativamente essa quantidade de pacotes de 1 até 50, e consequentemente aumentamos o fluxo de dados na rede, o desempenho do NS-3 é prejudicado em relação ao NS-2.

Concluímos então, que o Network Simulator 3 é capaz de gerenciar, mantendo a escalabilidade, uma grande quantidade de nós, no entanto, o seu tratamento de pacotes demanda muito processamento, prejudicando seu desempenho e escalabilidade à medida que o tráfego de dados aumenta. Esse efeito não pôde ser observado

nos trabalhos relacionados [23] porque eles se limitaram a realizar seus experimentos com pacotes de pequeno tamanho e em baixa quantidade. Como sabemos, o NS-3 apresenta um *framework* complexo e que tenta simular o comportamento da rede e o processamento dos pacotes com bastante realidade. Certamente esse é o motivo do problema relatado.

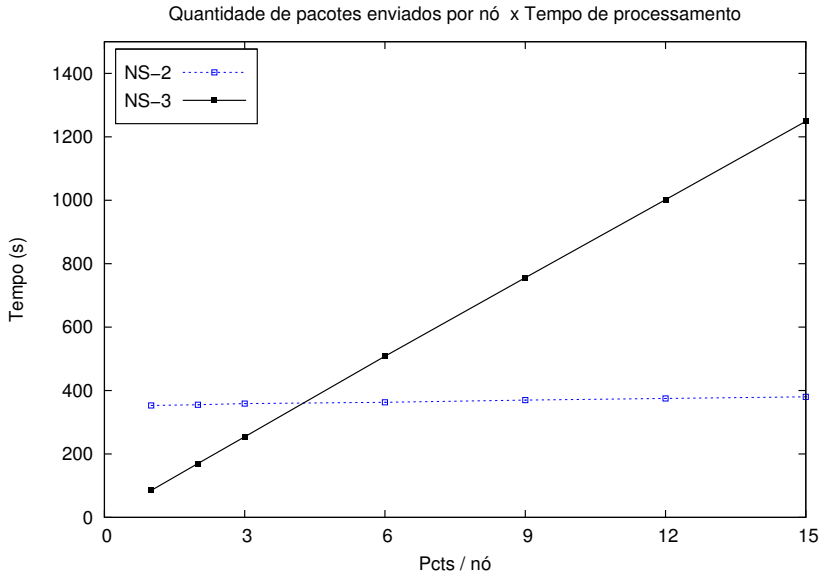


Figura 4.9: Avaliando desempenho dos simuladores em função da quantidade de tráfego na rede.

Tabela 4.5: Tempo de processamento da simulação versus quantidade de tráfego na rede.

	Tempo de processamento NS-2	**Tempo de processamento NS-3**
1 pct	354s	85s
2 pcts	355s	170s
3 pcts	359s	254s
6 pcts	363s	508s
9 pcts	370s	756s
12 pcts	375s	1002s
15 pcts	379s	1249s
50 pcts	428s	4076s

Capítulo 5

Conclusões e Trabalhos futuros

5.1 Conclusões

Abordamos nesse trabalho, alguns assuntos relacionados à avaliação de desempenho de sistemas, mais especificamente sobre algumas das principais técnicas de simulação aplicadas às redes *peer-to-peer*. Assim como a simulação de qualquer grande rede que haja intenso fluxo de dados, a simulação desse tipo de rede ainda é um desafio, uma vez que em nível de pacotes, a grande quantidade de informações a ser processada pelo simulador resulta em grande custo computacional e consequente demora para a obtenção dos resultados desejados. Por outro lado, se adotada a simulação em nível de fluxos, apesar de haver significativo ganho de velocidade de processamento, devido a redução da complexidade, o usuário é penalizado com a redução de informações detalhadas e sobretudo com a redução da precisão da simulação.

Neste trabalho, implementamos o protocolo Bittorrent para o simulador NS-3, tanto em nível de pacotes como em nível de fluxos. Inicialmente, pôde ser verificada a capacidade de precisão de ambos os modelos de simulação. Certificamos que o simulador em nível de fluxos não é capaz de detectar variações na janela de transmissão do TCP provocadas por congestionamentos ou erros de transmissão, resultando em menor capacidade de precisão em relação aos simuladores em nível de pacotes.

Com relação ao desempenho do Network Simulator 3, foi possível concluir que em nível de pacotes, apesar de o mesmo ser capaz de simular uma quantidade muito superior de nós na rede, em relação ao seu antecessor, NS-2, seu desempenho é prejudicado à medida que o tráfego de dados se intensifica.

É possível destacar dois aspectos potencialmente responsáveis por tais resultados. Primeiramente, podemos observar nas documentações e nas próprias simulações do NS-3, que o mesmo apresenta uma complexidade e detalhamento muito superior da representação das camadas de rede e de transporte em relação ao NS-2. Essa complexidade certamente colabora para o incremento das necessidades de processamento de cada pacote adicional que passa a circular na rede. Daí a queda de desempenho do NS-3 em nível de pacotes à medida que se aumenta a quantidade de tráfego na rede.

O outro aspecto que justifica os valores obtidos nos experimentos de desempenho, se dá na simulação em nível de fluxos. Havendo abstração das camadas de rede e de transporte, o NS-3 apresentou melhores resultados que o NS-2, uma vez que o primeiro leva vantagem por ser um simulador implementado em linguagem totalmente compilada (C++), diferentemente de seu antecessor que parcialmente adota linguagem interpretada (OTcl) e parcialmente linguagem compilada (C++). Daí o NS-3, em nível de fluxos, apresentar sempre melhor desempenho que seu antecessor.

5.2 Trabalhos futuros

Como vimos anteriormente, a simulação em nível fluidos é uma modelagem abstrata que representa o tráfego da rede como um fluxo de fluido contínuo, em vez de uma seqüência discreta de instâncias de pacotes. Dessa forma, um grande número de pacotes pode ser representado por um único fluido e a representação de todas as camadas da pilha de protocolos torna-se viável, mantendo-se a escalabilidade do sistema. Por se encaixar exatamente nas necessidades da simulação de redes P2P, essa técnica de simulação pode ser capaz de gerar bons resultados em trabalhos futuros.

O efeito *ripple*, no entanto, será um problema a ser contornado, uma vez que redes como essas, normalmente apresentam uma grande quantidade de filas (em virtude da quantidade de nós) e inúmeros fluxos simultâneos de dados (em virtude da troca de arquivos). Em suma, este é um método complexo e que além de exigir uma modelagem matemática para cada tipo de rede simulada, pela forma como é definido, compreende um difícil método para o gerenciamento dos diferentes tipos de mensagens que circulam na rede.

Outro aspecto que pode ser estudado em trabalhos futuros é a melhoria do simulador Bittorrent que nós desenvolvemos. O simulador certamente pode ser aperfeiçoado e incrementado com a implementação de técnicas adicionais já propostas como expansão do protocolo, dentre elas estratégias como o *Super Seeding*, *End Game*, etc.

Referências Bibliográficas

[1] LUA, E. et al. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials, Volume 7, Páginas 72-93*, 2005.

[2] FIGUEIREDO, D. et al. On the efficiency of fluid simulation of networks. *Computer Networks: The International Journal of Computer and Telecommunications Networking, Volume 50, Páginas 1974-1994. Elsevier*, 2006.

[3] HUGGARD, M.; TAILLARD, B. Using Fluid Models for AQM Evaluation. *(Dissertação) Department of Computer Science, Trinity College Dublin*, 2005.

[4] GU, Y.; LIU, Y.; TOWSLEY, D. On integrating fluid models with packet simulation. *INFOCOM 2004. 23th AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004.

[5] JAIN, R. The Art of Computer Systems Performance Analysis. *John Wiley & Sons*, 1991.

[6] NETWORK Simulator 3. Disponível em http://www.nsnam.org. Acesso em Junho de 2010.

[7] NETWORK Simulator 2. Disponível em http://www.isi.edu/nsnam/ns/. Acesso em Junho de 2010.

[8] AGUIAR, A. et al. Análise comparativa de simuladores de redes baseados em pacotes versus simuladores utilizando abstracao de fluidos. *SBRC 2008 - 26º Simpósio Brasileiro de Redes de Computadores*, 2008.

[9] MARSAN, M. et al. Using partial differential equations to model TCP mice and elephants in large IP networks. *IEEE/ACM Transactions on Networking (TON), Volume 13, Páginas 1289-1301*, 2005.

[10] QIU, D.; SRIKANT, R. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. *2004 conference on Applications, technologies, architectures, and protocols for computer communications, Páginas 367-378, ACM/SIGCOMM*, 2004.

[11] HE, Q. et al. Mapping peer behavior to packet-level details: a framework for packet-level simulation of peer-to-peer systems. *11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, 2003. MASCOTS 2003*, 2003.

[12] PATANROI, D.; LINGAPPA, G. GiaSim: GIA Implementation Using NS2 Simulator. *(Relatório técnico) Department of Computer Science Iowa State University of Science and Technology Ames, USA.*, 2004.

[13] KARAKAYA, M.; KORPEOGLU, I.; ULUSOY, O. GnuSim: a general purpose simulator for Gnutella and unstructured P2P networks. *(Relatório técnico) Department of Computer Engineering, Bilkent University, Citeseer*, 2005.

[14] SCHWETMAN, H. CSIM: a C-based process-oriented simulation language. *Proceedings of the 18th conference on Winter simulation, Páginas 387-396*, ACM, 1986.

[15] YANG, W.; ABU-GHAZALEH, N. GPS: a general peer-to-peer simulator and its use for modeling BitTorrent. *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005*, 2005.

[16] OMNET++. Disponível em http://www.omnetpp.org. Acesso em Junho de 2010.

[17] GARCIA, P. et al. PlanetSim: A New Overlay Network Simulation Framework. *Software Engineering And Middleware: 4th International Workshop, SEM 2004, Linz, Austria, Páginas 123-136*, Springer, 2005.

[18] JELASITY, M. et al. *Peersim: A peer-to-peer simulator*. Disponível em http://peersim.sourceforge.net/. Acesso em Junho de 2010.

[19] DEFUDE, B. *P2P simulation with peersim (Relatório técnico)*. 2007. Disponível em http://stromboli.itsudparis.eu/bernard/ASR/projets/soutenances/RanaivoSabourin/rapportSimulation_P2P.pdf. Acesso em Junho de 2010.

[20] BAKER, M.; LAKHOO, R. Peer-to-peer simulators. *(Relatório técnico) ACET Centre, The University of Reading, AMG*, 2007.

[21] NAICKEN, S. et al. A survey of peer-to-peer network simulators. *Proceedings of The 7th Annual Postgraduate Symposium, Liverpool, UK*, Citeseer, 2006.

[22] NAICKEN, S. et al. Towards yet another peer-to-peer simulator. *4th International Working Conference Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs 06)*, Citeseer, 2006.

[23] WEINGARTNER, E.; LEHN, H. V.; WEHRLE, K. A performance comparison of recent network simulators. *International Conference on Communications 2009 (ICC 2009)*, IEEE, 2009.

[24] HENDERSON, T. et al. Network simulations with the ns-3 simulator. *SIGCOMM 2008, Demonstration competition, Seattle, WA, USA*, 2008.

[25] EGER, K. et al. Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations. *Second workshop on Use of P2P, GRID and agents for the development of content networks*, ACM, 2007.

[26] MARTÍNEZ-YELMO, I. et al. Analysis of searching mechanisms in hierarchical p2p based overlay networks. *Universidad Carlos III de Madrid. 6th Annual Mediterranean Ad Hoc Networking Workshop*, 2007.

[27] RISSON, J.; MOORS, T. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks: The International Journal of Com-*

puter and Telecommunications Networking, Volume 50, Páginas 3485-3521, Elsevier, 2006.

[28] KINI, U.; SHETTY, S. Peer-to-Peer Networking. *Resonance, Volume 6, Number 12, 69-79, Springer*, 2001.

[29] THE Annotated Gnutella Protocol Specification v0.4. Disponível em http://rfc-gnutella.sourceforge.net/developer/stable/. Acesso em Junho de 2010.

[30] STOICA, I. et al. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking (TON), Volume 11, Páginas 17-32*, 2003.

[31] BITTORRENT, Protocol Specification v1.0. Disponível em http://wiki.theory.org/BitTorrentSpecification. Acesso em Junho de 2010.

[32] COHEN, B. Incentives build robustness in BitTorrent. *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[33] AHN, J.; DANZIG, P. Packet network simulation: speedup and accuracy versus timing granularity. *IEEE/ACM Transactions on Networking (TON), Volume 4, Páginas 743-757*, 1996.

[34] SHEIKH, A. E.; AJEELI, A. A.; ABU-TAIEH, E. Simulation and modeling: Current technologies and applications. *Igi Global*, 2007.

[35] SHANMUGAN, K. Simulation and implementation tools for signal processing and communication systems. *Communications Magazine, IEEE, Volume 32, Páginas 36-40*, 1994.

[36] LIU, Y. et al. Scalable fluid models and simulations for large-scale IP networks. *ACM Transactions on Modeling and Computer Simulation (TOMACS), Volume 14, Páginas 305-324*, 2004.

[37] KAVIMANDAN, A. et al. Network simulation via hybrid system modeling: a time-stepped approach. *14th International Conference on Computer Communications and Networks. ICCCN 2005.*, 2005.

[38] KIDDLE, C. et al. Hybrid packet/fluid flow network simulation. *17th Workshop on Parallel and Distributed Simulation.PADS 2003, Páginas 143-152*, 2003.

[39] PAN, J.; JAIN, R. A Survey of Network Simulation Tools: Current Status and Future Developments. *Department of Computer Science, Washington University in St. Louis*, 2008.

[40] LACAGE, M.; HENDERSON, T. Yet another network simulator. *WNS2'06, Volume 202, Proceeding from the 2006 workshop on ns-2: the IP network simulator*, ACM, 2006.

[41] BRESLAU, L. et al. Advances in network simulation. *USC/Information Sciences Institute, IEEE Computer 2000, Volume 33, pp 59-67*, 2000.

[42] FALL, K.; VARADHAN, K. The ns Manual (ns Notes and Documentation). *The VINT Project*, 2009. Disponível em http://www.isi.edu/nsnam/ns/. Acesso em Junho de 2010.

[43] GTNETS - Georgia Tech Network Simulator. Disponível em http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS. Acesso em Junho de 2010.

[44] CARTWRIGHT, J.; PIRO, O. The dynamics of Runge–Kutta methods. *Citeseer, J. Bifurcation and Chaos, Volume 2, Páginas 427-449*, 1992.

[45] STEVENS, W. TCP/IP Illustrated, Vol.1: The Protocols. *Addison-Wesley, Boston*, 1994.

[46] TANENBAUM, A. Redes de computadores. *Pearson educación*, 2003.

[47] PRETE, L.; SHINODA, A. Análise do Comportamento das Variações do Protocolo TCP. *XXXII Congresso Nacional de Matemática Aplicada e Computacional - CNMAC*, 2009.

[48] WILLIAMS, T.; KELLEY, C. et al. *GNUplot: an interactive plotting program*. Disponível em http://www.gnuplot.info/docs. Acesso em Junho de 2010.

Apêndice A

Pseudocódigo Bittorrent

Procedimento iniciaAplicacao

Entrada: Porta da aplicação, ponteiro para o *tracker*

Saída: -

$inicializaVariaveis()$; // Inicializa variáveis de estado, estatísticas e timers

$registraNoTracker()$; // Nó se registra no *tracker*

$reescalonaChecaPeers()$; // Escalona Timer para verificar necessidade de consulta ao *tracker*

Procedimento checaPeers

Entrada: Lista de peers conectados ao nó

Saída: Lista atualizada de peers conectados ao nó

se *quantidade de peers conectados ao nó é menor que o mínimo desejado* **então**

    $consultaTracker()$; // Solicita ips de novos peers ao *tracker*

    para cada *novo peer* **faça**

        $enviaHandshake()$; // Tenta estabelecer conexão iniciando handshake

    fim

fim

$reescalonaChecaPeers()$; // Reescalona Timer para verificar necessidade de consulta ao *tracker*

Procedimento requisitarChunk

Entrada: Índice do peer que será requisitado, nº de requisições (numReqRestantes)

Saída: Lista de requisições atualizada

se *existe algum chunk incompleto* **então**

$indiceChunk \leftarrow estrategiaStrictPriority();$ `// Seleciona primeiro o chunk incompleto`

senão se *ainda não tem chunks completos* **então**

$indiceChunk \leftarrow estrategiaRandomFirstPiece();$ `// Seleciona chunk aleatoriamente`

senão

$indiceChunk \leftarrow estrategiaRarestChunk();$ `// Dos chunks que não possui, seleciona o mais raro`

fim

enquanto $numReqRestantes > 0$ **faça**

$indiceRequisicao \leftarrow registraRequisicao(indicePeer, indiceChunk);$

$enviaRequest(indiceRequisicao);;$ `// Envia mensagem de requisição do chunk selecionado`

$numReqRestantes \leftarrow numReqRestantes - 1;$

fim

Procedimento trataHandshake

Entrada: Msg handshake, Ip do peer emissor do handshake

Saída: -

se *é o 3º handshake da sessão* **então**

$enviaBitfield();$ `// Envia mensagem Bitfield para o peer`

senão se *é o 2º handshake da sessão* **então**

se *quantidade de peers conectados a mim é menor que o máximo permitido* **então**

$enviaHandshake(ipPeer);$ `// Envia mensagem handshake para o peer`

$enviaBitfield(ipPeer);$ `// Envia mensagem bitfield para o peer`

fim

senão se *é o 1º handshake da sessão* **então**

se *quantidade de peers conectados a mim é menor que o máximo permitido* **então**

$enviaHandshake(ipPeer);$ `// Envia mensagem de handshake para o peer`

fim

fim

Procedimento trataBitfield

Entrada: Msg Bitfield, Ip do peer emissor do Bitfield

Saída: -

se *tem interesse em ao menos um chunk do peer e ainda não manifestou* **então**

$enviaInterested(ipPeer);$ `// Envia mensagem Interested`

fim

Procedimento trataInterested

Entrada: Msg Interested, Ip do peer emissor do Interested

Saída: -

se *é o primeiro peer a se interessar por este nó* **então**

$checaChoking(ipPeer)$; `// Ativa a verificação periódica de chokings`

fim

Procedimento trataUnchoke

Entrada: Msg Unchoke, Ip do peer emissor do Unchoke

Saída: -

se *o nó ainda tem interesse nesse peer* **então**

$requisitarChunk(ipPeer, numReqRestantes)$

senão

$enviaNotInterested(ipPeer)$; `// Envia mensagem Not Interested ao peer`

fim

Procedimento trataNotInterested

Entrada: Msg NotInterested, Ip do peer emissor do NotInterested

Saída: -

se *o peer está unchoke* **então**

$enviaChoke(ipPeer)$; `// Envia mensagem Choke para o peer`

fim

Procedimento trataChoke

Entrada: Msg Choke, Ip do peer emissor do Choke

Saída: -

$atualizaVariaveis()$

Procedimento checaChoking

Entrada: -

Saída: Lista de peers unchoke atualizada

se *total de peers conectados ao nó é inferior ao máximo de peers unchoke* **então**

para cada *peer interessado e choke* **faça**

$unchokePeer()$; `// Muda situação do peer para unchoke`

fim

$reescalonaChoking()$; `// Reescalona Timer para a próxima checagem de choking`

retorna

fim

$defineCandidatos()$; `// Lista peers com interesse no nó para que sejam submetidos à seleção`

$selecionaPeers()$; `// Define quais peers serão unchoke. Aplicamos apenas estratégia aleatória`

retorna

Procedimento trataHave

Entrada: Msg Have, Índice do chunk(indiceChunk), Ip do peer emissor do Have (ipPeer)

Saída: -

se *nó não tem o chunk indiceChunk* **então**

  se *nó não tem interesse no peer* **então**

    $enviaInterested(ipPeer)$; `// Envia mensagem Interested ao peer`

  fim

  se *nó não está choking para o peer e numReqRestantes é maior que 0* **então**

    $requisitarChunk(ipPeer, numReqRestantes)$

  fim

fim

retorna

Procedimento trataRequest

Entrada: Msg Request, Índice do chunk(indiceChunk), Ip do peer emissor do Request (ipPeer)

Saída: -

se *peer está unchoke* **então**

  $enviaPiece(ipPeer, indiceChunk)$; `// Envia mensagem Piece ao peer`

fim

retorna

Procedimento trataPiece

Entrada: Msg Piece, Índice do chunk, tamanho do chunk e Ip do peer

Saída: -

se *Chunk está incompleto* **então**

  $requisitarChunk(indiceChunk, 1)$; `// Continua requisições do mesmo chunk`

fim

senão

  para cada *peer conectado ao nó* **faça**

    $enviaHave(indiceChunk, ipPeer)$; `// Envia mensagem Have para cada peer`

    se *Não tem mais interesse em nenhum chunk* **então**

      $enviaNotInterested(ipPeer)$; `// Envia mensagem Not Interested ao peer`

    fim

  fim

  se *Completou todos os chunks* **então**

    $tornaSeed()$; `// Permanece servindo rede até que todos estejam completos.`

  fim

fim

Apêndice B

Simulações Bittorrent

Tabela B.1: Tempo (em segundos) de conclusão do download no NS-2 em nível de pacotes

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2 peers	1022	1022	1022	1022	1022	1022	1022	1022	1022	1022
5 peers	1584	1602	1608	1585	1573	1585	1580	1584	1555	1602
10 peers	1672	1709	1691	1686	1696	1677	1682	1712	1697	1678
25 peers	1818	1755	1817	1804	1787	1793	1820	1816	1798	1833
50 peers	1844	1823	1818	1847	1844	1847	1872	1825	1807	1833
100 peers	1891	1896	1902	1877	1872	1883	1867	1905	1884	1864
150 peers	1940	1908	1888	1929	1906	1932	1917	1928	1923	1916
200 peers	1933	1958	1937	1931	1921	1973	1922	1915	1914	1936

	Intervalo de confiança	**Média aritmética**
2 peers	0	1022
5 peers	9.62757151130159	1585.8
10 peers	8.38697677401742	1690
25 peers	13.7863185039252	1804.1
50 peers	11.5584342116049	1836
100 peers	8.86424876457135	1884.1
150 peers	9.39988411130761	1918.7
200 peers	11.6760055446491	1934

Tabela B.2: Tempo (em segundos) de conclusão do download no NS-3 em nível de pacotes

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2 peers	1026	1026	1026	1026	1026	1026	1026	1026	1026	1026
5 peers	1556	1574	1551	1563	1565	1566	1546	1565	1535	1558
10 peers	1728	1765	1772	1747	1753	1739	1730	1764	1747	1723
25 peers	1925	1903	1904	1893	1893	1924	1912	1909	1902	1915
50 peers	1944	1925	1921	1915	1931	1894	1936	1926	1894	1904
100 peers	1946	1944	1925	1941	1923	1989	1953	1962	1934	1901
150 peers	2002	2063	2019	2006	2007	2008	2007	2001	1998	2015
200 peers	2024	2044	2046	2041	2082	2000	2079	2048	2054	2014

	Intervalo de confiança	**Média aritmética**
2 peers	0	1026
5 peers	7.06342789204409	1557.9
10 peers	10.460488542134	1746.8
25 peers	6.96944134936281	1908
50 peers	10.6111934533151	1919
100 peers	14.8198854919987	1941.8
150 peers	11.6401254766403	2012.6
200 peers	16.0398036926721	2043.2

Tabela B.3: Tempo (em segundos) de conclusão do download no NS-2 em nível de pacotes (com descartes)

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
5 peers	1886	1835	1854	1852	1862	1833	1834	1838	1830	1845
10 peers	1978	1971	1954	1959	1965	1992	1996	1965	2005	1975
25 peers	2141	2127	2153	2142	2178	2143	2141	2114	2094	2143
50 peers	2138	2186	2159	2173	2180	2151	2181	2158	2213	2202
100 peers	2225	2227	2225	2210	2243	2213	2230	2216	2248	2245
150 peers	2266	2232	2260	2367	2269	2284	2237	2268	2290	2285
200 peers	2286	2252	2303	2336	2305	2309	2243	2246	2257	2313

	Intervalo de confiança	**Média aritmética**
5 peers	10.740928504865	1846.9
10 peers	10.4163271080388	1976
25 peers	13.9211109301278	2137.6
50 peers	14.3744055085121	2174.1
100 peers	8.34002484961456	2228.2
150 peers	23.142334650367	2275.8
200 peers	20.4980542957095	2285

Tabela B.4: Tempo (em segundos) de conclusão do download no NS-3 em nível de pacotes (com descartes)

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
5 peers	1734	1714	1701	1733	1710	1694	1706	1732	1723	1713
10 peers	2070	2079	2048	2043	2069	2074	2070	2051	2084	2086
25 peers	2181	2196	2179	2168	2227	2167	2152	2167	2171	2137
50 peers	2185	2207	2228	2171	2177	2174	2208	2176	2165	2184
100 peers	2215	2206	2247	2204	2239	2189	2182	2191	2258	2179
150 peers	2268	2293	2275	2266	2223	2297	2232	2235	2245	2284
200 peers	2311	2283	2292	2275	2303	2296	2277	2302	2280	2271

	Intervalo de confiança	**Média aritmética**
5 peers	8.70659448448472	1716
10 peers	9.36872808372668	2067.4
25 peers	15.1261877555795	2174.5
50 peers	12.5030409427635	2187.5
100 peers	17.4425783383519	2211
150 peers	16.4367129015341	2261.8
200 peers	8.48815023677599	2289

Tabela B.5: Tempo de conclusão do download no NS-2 em nível de fluxos

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2 peers	821	821	821	821	821	821	821	821	821	821
5 peers	1421	1461	1421	1381	1421	1411	1491	1401	1421	1401
10 peers	1481	1511	1501	1431	1431	1471	1451	1471	1461	1461
25 peers	1601	1571	1551	1601	1551	1591	1581	1611	1561	1611
50 peers	1671	1701	1701	1681	1701	1621	1721	1681	1701	1741
100 peers	1711	1731	1661	1711	1751	1741	1731	1721	1681	1731
150 peers	1731	1801	1711	1691	1741	1731	1731	1721	1691	1751
200 peers	1751	1761	1731	1721	1731	1771	1791	1781	1721	1781

	Intervalo de confiança	**Média aritmética**
2 peers	0	821
5 peers	19.5560366039401	1423
10 peers	16.3199615603381	1467
25 peers	14.5501568458004	1583
50 peers	19.9129082930286	1692
100 peers	17.0865698969542	1717
150 peers	9.6973942823843	1730
200 peers	6.5407748979951	1754

Tabela B.6: Tempo (em segundos) de conclusão do download no NS-3 em nível de fluxos

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2 peers	810	810	810	810	810	810	810	810	810	810
5 peers	1330	1400	1380	1400	1360	1360	1420	1390	1360	1370
10 peers	1590	1550	1590	1610	1570	1660	1620	1610	1560	1630
25 peers	1640	1650	1620	1620	1610	1610	1620	1630	1660	1610
50 peers	1690	1690	1690	1700	1710	1630	1690	1680	1650	1660
100 peers	1720	1690	1680	1720	1660	1710	1740	1670	1670	1650
150 peers	1690	1700	1690	1680	1640	1700	1670	1670	1690	1670
200 peers	1750	1760	1720	1730	1710	1750	1780	1730	1720	1690

	Intervalo de confiança	**Média aritmética**
2 peers	0	810
5 peers	16.280683617566	1377
10 peers	20.9572610934941	1599
25 peers	10.9516611137589	1627
50 peers	15.307808087384	1679
100 peers	18.5823697536507	1691
150 peers	11.3158573498847	1680
200 peers	16.3199615603381	1734

Tabela B.7: Tempo (em segundos) de conclusão do download no NS-2 em nível de fluxos (com descarte)

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2 peers	831	831	831	831	831	831	831	831	831	831
5 peers	1441	1461	1421	1431	1451	1431	1421	1451	1451	1401
10 peers	1471	1501	1501	1531	1471	1431	1491	1471	1501	1501
25 peers	1611	1591	1601	1621	1601	1581	1571	1621	1611	1581
50 peers	1661	1681	1661	1711	1701	1701	1671	1681	1701	1691
100 peers	1671	1681	1731	1761	1751	1701	1711	1691	1711	1661
150 peers	1741	1791	1781	1731	1711	1751	1751	1741	1731	1751
200 peers	1701	1761	1761	1711	1791	1811	1741	1771	1751	1771

	Intervalo de confiança	**Média aritmética**
2 peers	0	831
5 peers	11.4097664899705	1436
10 peers	16.8349129986655	1487
25 peers	10.8537890783774	1599
50 peers	11.0293334424521	1686
100 peers	20.4938892942857	1707
150 peers	14.6233104333495	1748
200 peers	20.7011128584263	1757

Tabela B.8: Tempo (em segundos) de conclusão (s) do download no NS-3 em nível de fluxos (com descarte)

seed / peers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2 peers	810	810	810	810	810	810	810	810	810	810
5 peers	1370	1390	1390	1420	1370	1400	1420	1380	1380	1350
10 peers	1590	1620	1590	1610	1590	1630	1610	1610	1570	1620
25 peers	1610	1640	1680	1630	1680	1630	1620	1620	1650	1620
50 peers	1670	1700	1710	1720	1670	1680	1650	1690	1680	1710
100 peers	1720	1660	1670	1690	1710	1710	1720	1690	1720	1700
150 peers	1730	1670	1680	1670	1680	1690	1710	1650	1650	1700
200 peers	1770	1730	1690	1730	1710	1750	1720	1740	1740	1730

	Intervalo de confiança	**Média aritmética**
2 peers	0	810
5 peers	13.7197479028414	1387
10 peers	11.3910465979694	1604
25 peers	15.4050897352135	1638
50 peers	13.641749841014	1688
100 peers	13.2126061866741	1699
150 peers	15.7476191442884	1683
200 peers	13.5317972721661	1731