Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

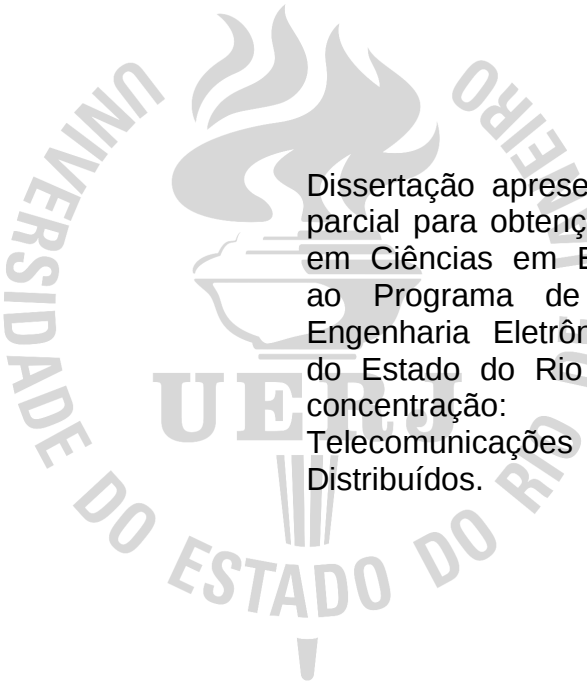
Flávia Marinho de Lima

Análise de desempenho de redes p2p com protocolo "*push/pull*" para distribuição de vídeo na presença de nós não-cooperativos

Rio de Janeiro
2010

Flávia Marinho de Lima

Análise de desempenho de redes p2p com protocolo "*push/pull*" para distribuição de vídeo na presença de nós não-cooperativos



Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações e Sistemas Distribuídos.

Orientador: Prof. Dr. Alexandre Sztajnberg

Rio de Janeiro
2010

CATALOGAÇÃO NA FONTE
UERJ/REDE SIRIUS/CTC/B

L733 Lima, Flávia Marinho de.

Análise de desempenho de redes p2p com protocolo "*push/pull*" para distribuição de vídeo na presença de nós não-cooperativos./ Flávia Marinho de Lima. - 2010.

107 f.

Orientador (es): .Alexandre Sztajnberg.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Redes de informação – Teses. 2. Internet - Teses. 3. Sistemas de transmissão de dados – Teses. 4. Engenharia Eletrônica - Teses. I. Sztajnberg, Alexandre. II. Universidade do Estado do Rio. III. Título.

CDU 004.738

_____ _____

Assinatura Data

Flávia Marinho de Lima

Análise de desempenho de redes p2p com protocolo "*push/pull*" para distribuição de vídeo na presença de nós não-cooperativos

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações e Sistemas Distribuídos.

Aprovada em 20 de julho de 2010

Banca Examinadora:

Prof. Dr. Alexandre Sztajnberg (Orientador)
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Marcélo Gonçalves Rubinstein
Faculdade de Engenharia - UERJ

Prof. Dr. Igor Monteiro Moraes
Instituto de Computação - UFF

Rio de Janeiro
2010

DEDICATÓRIA

Aos meus pais Izaneide e Severino e ao meu amor Fernando que sempre me dão força para alcançar os objetivos que tanto almejo.

AGRADECIMENTOS

Agradeço a todos os meus familiares e amigos que de alguma forma contribuíram por mais essa conquista. Um agradecimento especial aos amigos Pedro Monteiro e Rogério Maurer que nunca me negaram ajuda quando eu tinha dúvidas em Java ou Shell Script.

Agradeço a todos os professores e aos colegas de Mestrado, Felipe, Vinícius e Dalbert pelos bons momentos que passamos juntos e pelas dificuldades que superamos.

Gostaria ainda de demonstrar o meu profundo agradecimento e respeito ao meu orientador professor Alexandre que contribuiu muito com suas orientações, críticas e conselhos.

Agradeço em particular ao professor Marcelo Rubinstein e ao professor Igor Moraes por terem aceito participar da banca examinadora.

RESUMO

de LIMA, Flávia. Análise de desempenho de redes p2p com protocolo “push/pull” para distribuição de vídeo na presença de nós não-cooperativos. 107 f. Dissertação (Mestrado em Engenharia Eletrônica) - Programa de Pós-Graduação em Engenharia Eletrônica, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2010.

O uso de Internet para a distribuição de fluxos de vídeo tem se mostrado uma tendência atual e traz consigo grandes desafios. O alicerce sobre qual a Internet está fundamentada, comutação por pacotes e arquitetura cliente-servidor, não proporciona as melhores condições para este tipo de serviço. A arquitetura P2P (peer-to-peer) vem sendo considerada como infraestrutura para a distribuição de fluxos de vídeo na Internet. A idéia básica da distribuição de vídeo com o suporte de P2P é a de que os vários nós integrantes da rede sobreposta distribuem e encaminham pedaços de vídeo de forma cooperativa, dividindo as tarefas, e colocando à disposição da rede seus recursos locais. Dentro deste contexto, é importante investigar o que ocorre com a qualidade do serviço de distribuição de vídeo quando a infraestrutura provida pelas redes P2P é “contaminada” por nós que não estejam dispostos a cooperar, já que a base desta arquitetura é a cooperação. Neste trabalho, inicialmente é feito um estudo para verificar o quanto a presença de nós não-cooperativos pode afetar a qualidade da aplicação de distribuição de fluxo de vídeo em uma rede P2P. Com base nos resultados obtidos, é proposto um mecanismo de incentivo à cooperação para que seja garantida uma boa qualidade de vídeo aos nós cooperativos e alguma punição aos nós não-cooperativos. Os testes e avaliações foram realizados utilizando-se o simulador *PeerSim*.

Palavras-chave: Redes P2P, distribuição de vídeo, seleção de vizinhos, protocolo *push/pull*, mecanismo de incentivo à cooperação, difusão de informações de pedaços.

ABSTRACT

Using the Internet for video stream is becoming a trend, but it brings many challenges. The foundation upon which the Internet is based, packet switching and client-server architecture, is not suitable for this type of service. P2P (peer to peer) architecture is being considered as an infrastructure for video streams on the Internet. The basic idea is that the several members of the overlay network cooperate in the task of distributing and fowarding video chunks, making available their local resources to the network. Within this context, it is important to investigate what happens to the quality of service of the video distribution when the infrastructure provided by the P2P network is “contaminated” with free-riding nodes, which are not willing to cooperate, since the basis of this architecture is cooperation. In this work, study is initially carried out to check how the presence of uncooperative nodes can affect the quality of the distribution application of video streaming on a P2P network. Based on these results, a mechanism is proposed to encourage cooperation in order to be guaranteed a video with good quality to the cooperative nodes and some punishment for those uncooperative. The tests and evaluations were performed using the PeerSim simulator.

Keywords: P2P systems, video streaming, partner selection mechanism, push/pull protocol, cooperation incentive mechanism, information diffusion in chunk-based.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

A distribuição de vídeo pela Internet traz grandes desafios para as diversas áreas da Ciência da Computação, pois é um serviço que necessita (i) de servidores com grande capacidade de processamento e armazenamento, e (ii) grande largura de banda, quando comparada a outros serviços, baixo retardo e variação de retardo (*jitter*) e poucas perdas. É também um serviço que apresenta requisitos específicos de escalabilidade, pois se estima que uma quantidade grande de usuários o utilize, e flexibilidade, uma vez que estes usuários precisam receber os fluxos de vídeo de acordo com sua capacidade de banda e recursos computacionais para tratamento e exibição dos mesmos. Tornar disponível uma solução que contemple simultaneamente estas características é importante para que se possa oferecer ao usuário um serviço de qualidade, mas não é tarefa trivial, dado que a arquitetura original da Internet não foi concebida com esta finalidade.

Neste contexto, as redes par-a-par, ou *peer to peer* (P2P)[1], vem sendo consideradas como solução potencial para a distribuição de vídeo na Internet, devido às suas características de escalabilidade e distribuição de responsabilidades. A idéia básica da distribuição de vídeo com o suporte de P2P é a de que os vários nós integrantes da rede sobreposta distribuem e encaminham pedaços de vídeo de forma cooperativa, dividindo as tarefas, e colocando à disposição da rede seus recursos locais. Com isso diminui-se a necessidade de canais reais com grandes bandas e servidores com grande capacidade de armazenamento e processamento. Nesta abordagem, quanto mais nós existirem, maior é a capacidade de distribuição de vídeo da rede, pois serão compartilhados mais recursos, sem a dependência de um único servidor ou uma federação de servidores, o que torna as redes P2P escaláveis e robustas.

Entretanto, mesmo nas redes P2P a distribuição de vídeo também apresenta desafios. É freqüente nestas redes que alguns nós, geralmente nós com mais recursos que os demais, fiquem sobrecarregados porque nem todos os nós participantes querem cooperar, diminuindo

1 Redes sobrepostas (*overlay*), que executam sobre redes como a Internet, com nós virtuais, sendo interligados por canais de comunicação também virtuais.

os benefícios da aplicação das redes P2P para resolver o problema da distribuição de vídeo.

De certa forma, com a existência de muitos nós com comportamento não-cooperativo, as redes P2P passam a apresentar características de um sistema centralizado, como mencionado anteriormente, podendo apresentar um desempenho ainda pior, pois os nós sobrecarregados não são necessariamente especializados para a atividade de armazenamento e distribuição de vídeo, tornando-se gargalos.

Estudos empíricos têm mostrado que nós que atuam como parasitas na rede, consumindo recursos, sem contribuir, são maioria em redes P2P voltadas para o compartilhamento de arquivos (Adar e Huberman, 2000)(Saroiu et al, 2003). Apesar deste tipo de comportamento já prejudicar o desempenho deste tipo de aplicação, no caso da aplicação de distribuição de vídeo este comportamento é ainda mais nocivo, uma vez que os usuários dessas redes não estão em busca apenas da disponibilidade do arquivo contendo o vídeo, mas também esperam alta qualidade na obtenção e exibição do mesmo. A sensação de um usuário que recebe um conteúdo de vídeo com atrasos em seu tempo de reprodução, pode ser em alguns casos, a mesma de quando não o recebe.

Para solucionarmos ou minimizarmos o problema dos nós parasitas, propomos neste trabalho a criação de um mecanismo de incentivo à cooperação que introduza nos sistemas e protocolos existentes características de justiça, punindo os nós parasitas, entregando aos mesmos fluxos de vídeo com qualidade baixa, e beneficiando nós cooperativos, entregando a estes fluxos de vídeo com qualidade superior. Todos os nós ficariam estimulados a cooperar e, sendo assim, a carga seria melhor distribuída entre todos, evitando gargalos.

Uma das formas de se transmitir arquivos ou um fluxo de vídeo em redes P2P pela Internet é dividindo o seu conteúdo em pedaços ou *chunks a fim de* transmiti-los de maneira independente uns dos outros. O *BitTorrent* (Hales e Patarin, 2005) utiliza este método para compartilhamento de arquivos em redes P2P, e alguns sistemas de distribuição de vídeo, como por exemplo o *CoolStreaming* (Xinyan et al, 2005), também utilizam este método, dividindo um fluxo de vídeo em vários pedaços. Para sistemas que utilizam este método de transmissão do conteúdo, um ponto tem grande relevância, o mecanismo de como é feito a difusão desses pedaços entre os nós da rede P2P.

Basicamente, pode-se dizer que existem três modelos de referência na literatura para o mecanismo de difusão dos pedaços, que são o modelo *push*, o modelo *pull*, e o modelo

baseado no estado de cada nó. No modelo *push*, os pedaços são enviados do nó pai para os nós filhos, sem o nó pai questionar se o nó filho precisa ou não daquele pedaço. O modelo *pull* é o oposto do modelo *push*, neste o nó filho faz a requisição de um pedaço ao nó pai sem saber se ele o possui. Já o modelo baseado em estado, se baseia nas trocas de informações entre os nós para saber o que cada nó vizinho possui, e assim pedir-lhe ou enviar-lhe um determinado pedaço.

Alguns trabalhos propõem modelos híbridos. Um desses trabalhos é o modelo *Interleave* (Sanghavi et al, 2007) que combina os modelos *push* e *pull*, e adiciona um mecanismo particular de política de seleção de pedaços. Este modelo foi projetado e analisado principalmente para transferência de arquivos. O trabalho também especulava que o modelo poderia ser utilizado para distribuição de vídeo, entretanto não houve investigação e consequente comprovação disto.

O trabalho (Cigno et al, 2008) fez uma análise detalhada do modelo *Interleave* e comprovou que ele era eficiente tanto para sistemas de distribuição de arquivos, quanto para sistemas de distribuição de vídeo. O autor utilizou o simulador *PeerSim* [PeerSim] para simular e comprovar a eficiência do modelo *Interleave*, ou *push/pull*.

Neste trabalho foi utilizada a modelagem e o simulador propostos por (Cigno et al, 2008) para que o comportamento do modelo proposto por (Sanghavi et al, 2007) seja estudado no contexto de uma rede P2P cujos nós podem possuir características não-colaborativas. Além disso, este trabalho propõe a inserção de um mecanismo de incentivo à cooperação ao modelo para que sejam introduzidas características de justiça.

Objetivos

Este trabalho possui dois objetivos, o primeiro é analisar o impacto causado no desempenho do protocolo de difusão de pedaços *push/pull* proposto em (Cigno et al, 2008) , pela ação de nós com comportamento não-cooperativo na rede P2P. O segundo objetivo é propor um mecanismo de incentivo à cooperação que introduza características de justiça ao

modelo, a fim de que mesmo diante de nós não-cooperativos, o modelo garanta um bom desempenho para os nós cooperativos.

Em um primeiro momento, analisa-se o desempenho do protocolo *push/pull* ante a inserção de nós não-cooperativos à rede P2P, pois a presença deste tipo de nó, em tese, prejudicaria o sistema como um todo. Faz-se necessário comprovar se isto de fato ocorre, para então, propor algo para minimizar os prejuízos. Para analisar tal comportamento, simulações são realizadas com diferentes tipos de cenários a fim de reproduzir um cenário real, aumentando-se gradativamente o número de nós maliciosos na rede. Assim, é possível inferir a influência dessa nova variável, porcentagem de nós maliciosos ou nós não-cooperativos, no sistema como um todo.

Foi necessário fazer alterações no simulador para que fosse possível a adição de nós não-cooperativos durante o processo de criação da rede P2P. Após esta alteração, verificou-se se o simulador alterado apresentava o mesmo comportamento do simulador original quando a porcentagem de nós não-cooperativos é igual a zero. Este item foi comprovado, os resultados das simulações foram iguais nos dois simuladores, para redes P2P cuja porcentagem de nós não-cooperativos era zero. Após esta constatação, as simulações incluindo nós não-cooperativos foram realizadas. Os resultados comprovaram que o desempenho do protocolo *push/pull* diminui a medida que a porcentagem de nós maliciosos aumentou.

Após esta fase, este trabalho propôs uma nova alteração ao modelo: a possibilidade de cada nó escolher os seus vizinhos na formação do grafo, baseado em uma “reputação” inicial. Com isso, a forma com que os vizinhos de um determinado nó é escolhida deixa de ser 100% aleatória e passa a levar em consideração se o nó candidato a vizinho é ou não cooperativo. A idéia é limitar a quantidade de nós não-cooperativos como vizinhos de nós cooperativos na formação do grafo, a fim de aumentar a qualidade de serviço dos mesmos e diminuir a qualidade dos nós não-cooperativos.

Novamente simulações foram realizadas com os mesmos parâmetros anteriores, porém com a nova alteração no modelo. Os resultados apresentados no Capítulo 3 mostraram que parte do segundo objetivo foi alcançada, pois houve uma melhora com relação ao desempenho dos nós cooperativos, entretanto de maneira geral, os nós não-cooperativos continuaram obtendo melhor desempenho.

Pôde-se constatar com esta nova alteração que ocorreu um melhor “balanceamento” dos nós não-cooperativos, e isto fez com que o desempenho da rede sobreposta como um todo melhorasse. Além disso, um fenômeno que ocorria ao acaso no modelo anterior, a não ativação de um nó, que poderia ser cooperativo ou não, passou a ocorrer apenas com nós não-cooperativos. Foi observado também, que esta alteração não obteve resultados melhores devido ao fato da formação da lista de vizinhos de cada nó ser estática.

Face aos primeiros resultados obtidos, foi proposto um mecanismo de incentivo à cooperação, a fim de beneficiar ainda mais os nós cooperativos e punir os nós não-cooperativos, para atingir o segundo objetivo do trabalho. O mecanismo é capaz de atualizar a lista de vizinhos de cada nó periodicamente, considerando um *ranking* de contribuição, onde cada nó possui uma pontuação.

Organização do Texto

O trabalho está dividido da seguinte forma.

No Capítulo 1, descreve-se as principais características das redes P2P. Um estudo sobre a teoria de redes P2P é visto de maneira breve, apresentando arquiteturas, principais protocolos, tipos de estruturas e uma comparação com o modelo cliente-servidor. Topologias para distribuição de vídeo são discutidas. Também é apresentado um estudo sobre os principais protocolos de difusão de pedaços em uma rede P2P que transmita o seu conteúdo dividindo o arquivo ou o fluxo de vídeo em pedaços independentes. Um destaque é dado ao protocolo de difusão *push/pull*.

No Capítulo 2, exibe-se os trabalhos relacionados ao protocolo *push/pull*, à distribuição de vídeo e aos mecanismos de incentivo à cooperação. Além disto, é feita uma primeira análise do desempenho do protocolo *push/pull* diante de nós não-cooperativos.

Já no capítulo 3, apresentam-se as simulações e resultados relacionados ao primeiro objetivo de maneira detalhada, mostrando os cenários, métricas e parâmetros envolvidos.

O mecanismo de incentivo à cooperação proposto é visto no capítulo 4.

Por fim, no capítulo 5, a conclusão e sugestões de trabalhos futuros são apresentadas.

CAPÍTULO 1 - FUNDAMENTOS TEÓRICOS

A Internet a cada ano que passa traz inovações capazes de mudar rotinas da sociedade. *Emails*, *sites* de relacionamento, vendas *on-line*, mensageiros instantâneos e tantos outros exemplos, comprovam a capacidade que a Internet possui em proporcionar os mais variados tipos de serviço aos usuários.

Avanços na tecnologia eletrônica que proporcionaram um aumento significativo no poder computacional das máquinas de usuários, aliados ao aumento da largura de banda ofertada pelas Operadoras de Telecomunicações, mudaram o portfólio de serviços ofertados na Internet. Alguns serviços que outrora eram inimagináveis, devido a alta necessidade de qualidade de serviço (altas taxas de transmissão, baixo retardo, baixa variação de retardo, poucas perdas), passaram a ser possíveis de serem providos através da Internet, com o auxílio de novos protocolos e tecnologias.

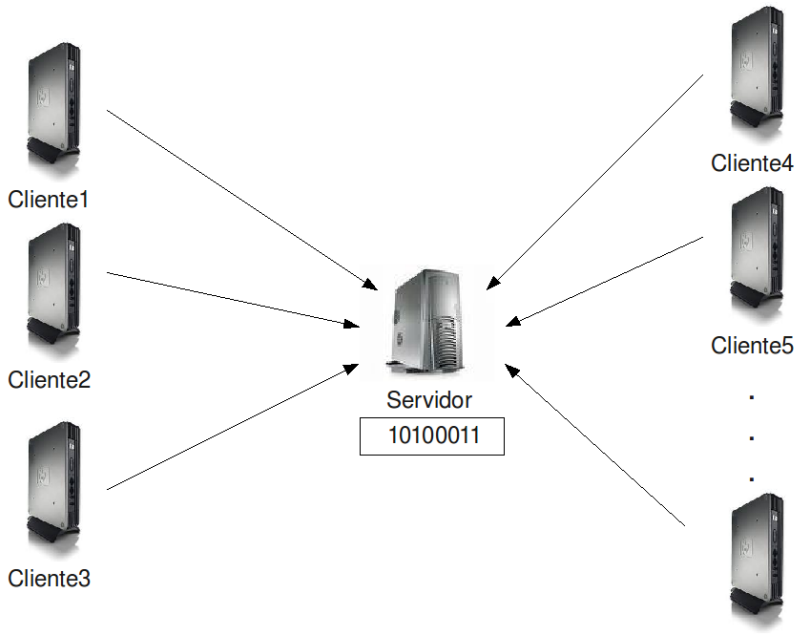
Um desses serviços inimagináveis é a distribuição de vídeo que pode ser feito por difusão, como na distribuição de TV aberta, ou sob demanda, que permite a interação do usuário na reprodução do vídeo, similar ao que existe hoje entre o espectador de um filme com um aparelho de DVD. Na transmissão por difusão, o usuário não tem controle do que está passando, não pode modificar o tempo de reprodução. Já na transmissão por demanda, o usuário pode modificar o tempo de reprodução do vídeo, através de comandos como avançar, retroceder, parar ou continuar, por exemplo.

O tráfego de vídeo é um tráfego que necessita de uma largura de banda alta quando comparado com o tráfego de dados ou voz, baixo retardo, variação de retardo pequena e pouca perda de pacotes (Moraes et al, 2003). É portanto, um tipo de tráfego que possui requisitos de qualidade de serviço bem definidos e rigorosos. Além disto, um sistema de distribuição de vídeo deve atender aos requisitos de alta escalabilidade, para atender um número arbitrário de usuários simultâneos, e a flexibilidade, para atender usuários com diferentes capacidades de recepção.

O modelo de rede atual sobre o qual a Internet se baseia não é capaz de fornecer, por si só, os requisitos mínimos necessários para garantir a qualidade de serviço desejada de um

sistema de distribuição de vídeo. Isto porque a Internet é uma rede que utiliza comutação por pacotes através do protocolo IP que se baseia em um serviço de melhor esforço na entrega, ou seja, sem garantias e sem qualidade de serviço. Além disso, a maioria das aplicações na Internet utiliza a arquitetura cliente-servidor, representada na Figura 1.

Como pode ser visto na etapa 1 da Figura 1, cada cliente estabelece uma conexão com o servidor e faz suas requisições, no caso específico todos os clientes requisitam o recurso simbólico “10100011”. Na etapa 2, o servidor responde a cada cliente com uma cópia do recurso requisitado. É fácil perceber que a arquitetura não é tão escalável em termos de banda, porque se *n* clientes requisitarem *y* de dados, o servidor gastará *ny* de banda para responder às requisições. Além disso, por ser uma arquitetura centralizada, o servidor acaba sendo um ponto único de falha. Este tipo de arquitetura cliente-servidor não é capaz de prover a escalabilidade desejada para um sistema de distribuição de vídeo, cuja ordem de grandeza esperada para o público é na ordem de milhões.



Etapa 1: Cada cliente requisita o recurso 10100011

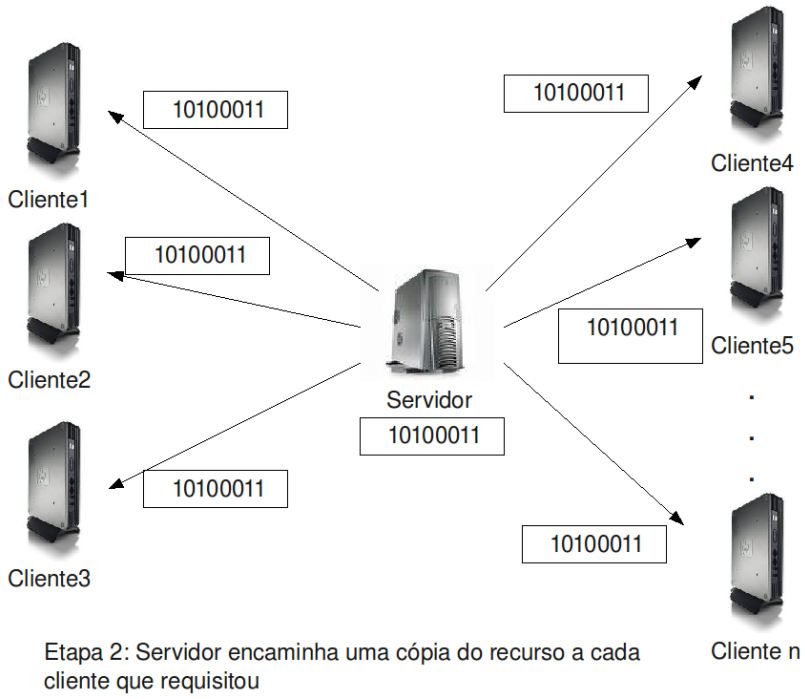


Figura 1: Representação da arquitetura cliente-servidor.

Para solucionar o problema de baixa escalabilidade que a arquitetura cliente-servidor possui, modelos de comunicação por multidifusão foram criados. Na comunicação por multidifusão, o servidor não precisa mais responder a cada uma das requisições com uma cópia do recurso requisitado, basta enviar uma cópia do recurso e todos os clientes que requisitaram a cópia do recurso a receberão, proporcionando assim, uma grande economia de banda passante.

Os primeiros modelos de comunicação por multidifusão criados, ainda em uso, são fundamentados na camada de rede. Neste modelo se faz necessário o gerenciamento de grupos, para permitir que membros entrem e saiam do grupo.

No modelo de rede, a comunicação por mutidifusão se faz pelo meio físico compartilhado. Em um grupo mais espalhado este mapeamento não é mais possível. É necessário o roteamento de mensagens que deve passar por várias redes.

Para executar o roteamento por multidifusão, cada roteador calcula uma árvore de amplitude que engloba todos os outros roteadores da sub-rede. Por exemplo, na Figura 2(a),

temos uma sub-rede com dois grupos, 1 e 2. Alguns roteadores estão associados a *hosts* que pertencem a um ou a ambos os grupos. Uma árvore de amplitude correspondente ao roteador situado mais à esquerda é mostrada na figura 2(b). Quando um processo envia um pacote de multidifusão a um grupo, o primeiro roteador examina sua árvore de amplitude e a poda, removendo todas as linhas que não levam a *hosts* que são membros do grupo. A Figura 2(c) mostra a árvore de amplitude do grupo 1 podada, e a figura 2(d) mostra a árvore de amplitude do grupo 2 podada. (Tanenbaum, 2003) Nota-se que é importante que os roteadores saibam quais de seus *hosts* pertencem a cada um dos grupos.

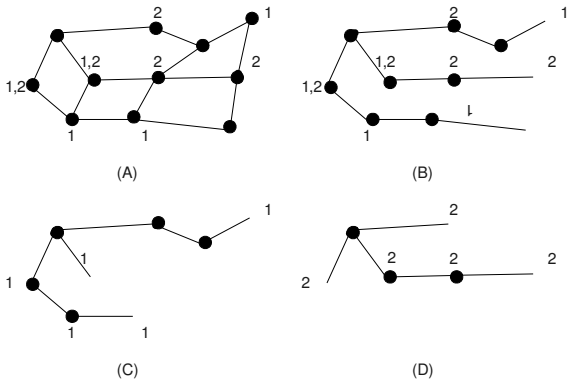


Figura 2: (a) Uma rede. (b) Uma árvore de amplitude correspondente ao roteador mais à esquerda. (c) Uma árvore de multidifusão correspondente ao grupo 1. (d) Uma árvore de multidifusão correspondente ao grupo 2. (Tanenbaum, 2003).

Os métodos por multidifusão fundamentados na camada de rede são muito eficientes em termos de banda e de diminuição de retardo, tornando-se ideais para redes LAN privadas. Entretanto, esses métodos são muito dispendiosos em termos financeiros e operacionais, uma vez que é necessária a configuração de cada elemento da rede que pertencer a um determinado grupo de destino, fazendo com que seja praticamente impossível obter este tipo de solução em redes públicas com Operadoras de Telecomunicações distintas.

A comunicação por multidifusão no nível da camada de aplicação foi criada como alternativa à comunicação por multidifusão fundamentada na camada de rede, para tentar diminuir os custos operacionais e financeiros deste, e assim como este, continuar obtendo melhor escalabilidade e flexibilidade que o modelo cliente-servidor. É importante salientar que a comunicação por multidifusão na camada de aplicação é menos eficiente em termos de

retardo e variação de retardo do que a fundamentada na camada de rede, e este é um ponto que precisa ser trabalhado nas soluções dadas.

Quando a comunicação por multidifusão é fundamentada na camada de aplicação, a comunicação é feita de forma ponto a ponto entre os nós finais, independente da topologia física da rede e dos equipamentos de interligação da mesma. Neste tipo de solução os nós são ditos autônomos e cooperativos, e possuem funções ora de servidor, ora de cliente.

As redes *peer-to-peer*, ou P2P, são exemplos de redes que utilizam a comunicação por multidifusão na camada de aplicação. Sistemas de distribuição de arquivos e sistema de distribuição de vídeo na Internet são exemplos de soluções que utilizam redes P2P para implementar a comunicação por multidifusão na camada de aplicação.

Além do modelo cliente-servidor, e do modelo P2P existem aplicações que utilizam modelos híbridos que misturam características dos dois modelos. Como exemplo pode ser citado a aplicação *BitTorrent* (Hales e Patarin, 2005), que é um sistema P2P de transferência de arquivos.

Na aplicação *BitTorrent* um arquivo é dividido em pequenos pedaços, ou *chunks*, que podem ser baixados entre os membros da rede P2P de forma paralela e independente. Pode-se dizer que um membro da rede possui o pedaço, apenas quando termina de baixar todo o pedaço. Um membro tem interesse em todos os outros membros que possuam um pedaço que ele não possua. O *BitTorrent* faz distinção entre os membros que já possuem o arquivo completo, conhecidos como *seeders*, e membros que estão baixando os pedaços do arquivo, conhecidos como *leechers*.

Quando um usuário quer obter um arquivo, ele precisa acessar um diretório global, que é um dos *sites web* conhecidos. Este diretório contém referências aos arquivos *.torrent*, que contém as informações necessárias para transferir um arquivo específico. Na realidade o *.torrent* se refere a algo conhecido como *rastreador*, um servidor que mantém uma contabilidade de quais membros ativos da rede possuem o arquivo desejado ou pedaços dele.

Quando o usuário identifica de onde os pedaços podem ser baixados, ele passa a ser um membro ativo, *leecher*, e que portanto deve ajudar outros membros, fornecendo os pedaços que possuir. Assim, é possível concluir que o *BitTorrent* combina soluções

centralizadas, a função dos rastreadores, e descentralizadas, a função de transferência de arquivos.

1.1 Redes P2P

Redes P2P são redes sobrepostas, que executam sobre redes como a Internet, com nós virtuais distribuídos, sendo interligados por canais de comunicação também virtuais, cuja principal característica é a ausência de um controle centralizado. Além disso, são sistemas auto-organizáveis, adaptáveis e escaláveis, onde todos os nós trabalham de forma colaborativa e compartilham seus recursos, assumindo ora o papel de cliente, ora o papel de servidor. Sendo assim, quanto maior o número de nós, maior é a capacidade do sistema. Esses sistemas permitem o compartilhamento de dados e recursos em uma escala grande, eliminando a necessidade de servidores e seus aparatos em infra-estrutura. De acordo com (Shirky, 2000), os aplicativos P2P são aplicativos que exploram os recursos disponíveis nos limites da Internet – armazenamento, ciclos de processamento, conteúdo e presença humana.

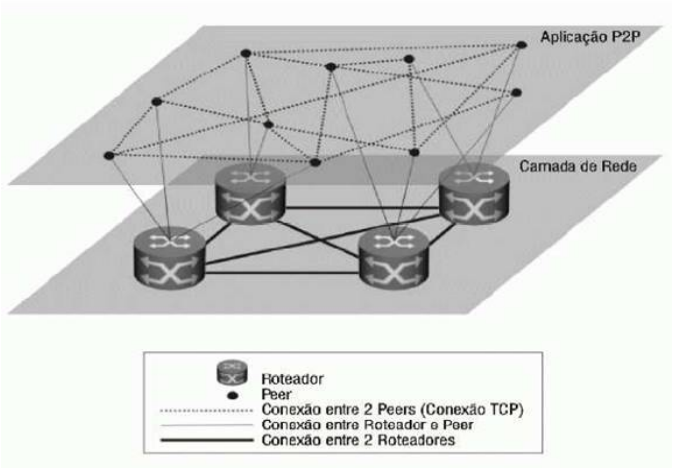


Figura 3: Representação de uma rede P2P [Wik01].

A Figura 3 ilustra uma rede P2P. A rede é composta por um conjunto de nós e por enlaces lógicos, os primeiros representam processos em máquinas finais de usuários, já os enlaces representam possíveis canais de comunicação, como conexões TCP, por exemplo. Pode ser observado que nós virtuais vizinhos não são necessariamente nós físicos vizinhos.

Diferente da arquitetura cliente-servidor, que é fundamentada em um modelo de arquitetura centralizada, onde apenas o servidor armazena e distribui os recursos, o modelo P2P é descentralizado: todos os nós que formam a rede sobreposta, ou lógica, podem armazenar e disponibilizar recursos. Esses nós ficam distribuídos na rede e cada um pode desempenhar função de cliente ou de servidor, ou ambos ao mesmo tempo.

Outro ponto importante que diferencia o modelo P2P do modelo cliente-servidor é que, se um nó desejar entrar ou sair da rede sobreposta, o impacto não será tão destruidor quanto se o servidor falhar no modelo cliente-servidor, neste último caso, simplesmente os recursos deixarão de ser distribuídos, mostrando o quanto o modelo P2P é mais robusto com relação à falhas.

Para que um nó se conecte à uma rede P2P é preciso que o mesmo se conecte logicamente a um nó que já pertença a mesma. Os nós conectados formam uma rede lógica sobreposta à rede física. Quanto a construção da rede lógica podemos classificar as redes P2P em: redes estruturadas e não estruturadas.

1.1.1 Redes estruturadas

Uma rede P2P estruturada é construída com a utilização de um procedimento determinístico. O procedimento mais usado é organizar os processos por meio de uma tabela de *hash* distribuída (*Distributed Hash Table*, DHT) (Tanenbaum e Steen, 2007)

Em uma rede baseada em DHT, os itens de dados recebem uma chave, como um identificador em um grande espaço de identificadores. Da mesma forma, os nós da rede também são representados por um identificador único. O ponto mais importante para um sistema baseado em DHT é implementar um esquema eficiente e determinístico que mapeie exclusivamente a chave de um item de dado para o identificador único de um nó, tendo como base somente alguma métrica de distância. (Balakrishnan et al, 2003)

Ao consultar um item de dado, o endereço de rede do nó responsável por aquele item de dado deve ser retornado. Na verdade, consegue-se isso roteando uma requisição para um item de dado até o nó responsável. Esse roteamento não está associado ao roteamento da

camada de rede, os responsáveis por este roteamento são os nós da rede sobreposta. Os protocolos *Chord* (Stoica et al, 2003), *CAN* (Content Addressable Network) (Ratnasamy et al, 2001), *Pastry* (Rowstron e Druschel, 2001), *Tapestry* (Zhao et al, 2004) são exemplos de protocolos de infraestrutura de roteamento de redes P2P estruturadas.

Para exemplificar o que foi dito, o protocolo *Pastry* será explicado de forma resumida. Esta explicação foi extraída de (Coulouris, Dollimore e Kindberg, 2007).

Todos os nós e objetos que podem ser acessados por meio do *Pastry* recebem identificadores globalmente exclusivos (*GUIDs*) de 128 bits. Os *GUIDs* dos nós são calculados por meio da aplicação de uma função de resumo segura na chave pública, com a qual cada nó é fornecido. As *GUIDs* de objetos como arquivos são calculados por meio da aplicação de uma função de resumo segura no nome do objeto, ou em alguma parte do estado armazenado do objeto.

Em uma rede com N nós participantes, o algoritmo de roteamento do *Pastry* direcionará corretamente uma mensagem endereçada a qualquer *GUID* em *O(log N)* etapas. Se o *GUID* identifica um nó correntemente ativo, a mensagem é enviada para esse nó; caso contrário, ela é enviada para o nó ativo cujo *GUID* é numericamente mais próximo a ele. Os nós ativos assumem a responsabilidade pelo processamento de requisições endereçadas para todos os objetos em sua vizinhança numérica.

O espaço de *GUID* é tratado de forma circular, onde o vizinho inferior do *GUID 0* é o $2^{128} - 1$. O algoritmo de roteamento completo envolve o uso de uma tabela de roteamento e um conjunto de folhas em cada nó para direcionar de forma eficiente as mensagens.

Cada nó ativo armazena um conjunto de folhas – um vetor L (de tamanho 2l) contendo os *GUIDs* e endereços dos nós cujos *GUIDs* são numericamente mais próximos nos dois lados de si próprio (*l* acima e *l* abaixo). Cada nó *Pastry* também mantém uma tabela de roteamento estruturada em árvore, fornecendo *GUIDs* e endereços IP para um conjunto de nós espalhados pela gama inteira de 2^{128} valores de *GUIDs* possíveis, com densidade de cobertura maior para os *GUIDs* numericamente mais próximos de si.

A Figura 4 mostra a estrutura da tabela de roteamento para um nó específico, e a Figura 5 apresenta as ações do algoritmo de roteamento *Pastry* para rotear uma mensagem do nó 65A1FC para o nó D46A1C. A tabela de roteamento é estruturada como segue: os *GUIDs*

são vistos como valores hexadecimais e a tabela classifica com base em seus prefixos hexadecimais. Cada entrada na tabela aponta para um dos, potencialmente, muitos nós cujos *GUIDs* têm prefixo relevante.

A tabela de roteamento mostrada na Figura 4 está localizada em um nó cujo *GUID* começa em 65A1. Os dígitos estão na base hexadecimal. As letras *n* representam pares [GUID, endereço IP] especificando o próximo *hop* a ser dado pelas mensagens endereçadas aos *GUIDs* que correspondem a cada prefixo dado. As entradas sombreadas indicam que o prefixo corresponde ao *GUID* corrente até o valor dado de *p*: a próxima linha abaixo, ou o conjunto deve ser examinado para encontrar um rota.

p=	Prefixos de GUID e manipuladores de nós n correspondentes															
0	0 n	1 n	2 n	3 n	4 n	5 n	6	7 n	8 n	9 n	A n	B n	C n	D n	E n	F n
1	60 n	61 n	62 n	63 n	64 n	65	66 n	67 n	68 n	69 n	6A n	6B n	6C n	6D n	6E n	6F n
2	650 n	651 n	652 n	653 n	654 n	655 n	656 n	657 n	658 n	659 n	65A	65B n	65C n	65D n	65E n	65F n
3	65A0 n	65A1	65A2 n	65A3 n	65A4 n	65A5 n	65A6 n	65A7 n	65A8 n	65A9 n	65AA n	65AB n	65AC n	65AD n	65AE n	65AF n

Figura 4: Primeiras quatro linhas de uma tabela de roteamento Pastry

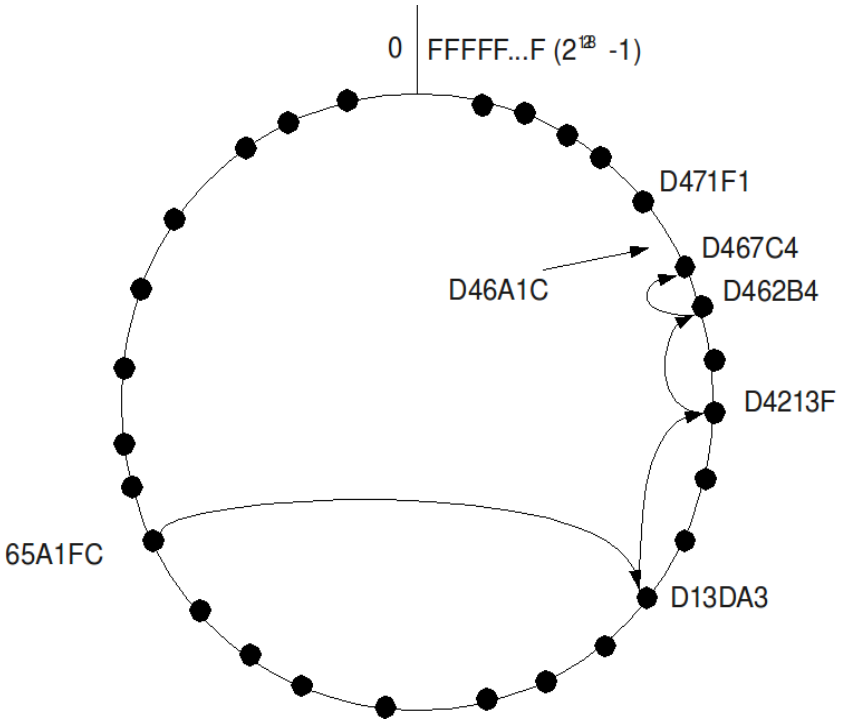


Figura 5: Exemplo de roteamento do Pastry.

Abaixo segue o pseudo-código do algoritmo (Coulouris, Dollimore e Kindberg, 2007) utilizado pelo *Pastry* para manipular uma mensagem *M* endereçada para um nó *D* (onde R[p,i] é o elemento na coluna *i*, linha *p* da tabela de roteamento):

```
Se (L-1 < D < L1) {# O destino D está dentro do conjunto de folhas ou é o nó corrente.
   Encaminha M para o elemento Li do conjunto de folhas com GUID mais
   próximo a D ou o nó corrente
}
Senão {# Usa a tabela de roteamento para enviar M para um nó com um GUID mais próximo
   Localiza p, o comprimento do prefixo comum mais longo de D e A;
   Localiza i, o (p+1)ésimo dígito hexadecimal de D.
   Se (R[p,i] < > null)
      Encaminha M para R[p,i] #Direciona M para um nó com um prefixo comum mais longe.
   Senão {# Não existe nenhuma entrada na tabela de roteamento
      Encaminha M para qualquer nó em L, ou R, com um prefixo comum
      de comprimento p, mas com um GUID numericamente mais próximo.
   }
}
```

O *Pastry* é a infra-estrutura de roteamento de mensagens implementada em várias aplicações incluindo o PAST (Druschel e Rowstron, 2001), um sistema de armazenamento de arquivos em repositório, e o *Squirrel* (Iyer et al, 2002), um serviço P2P para uso de cache *web*.

1.1.2 Redes não estruturadas

As redes não estruturadas geralmente se baseiam em algoritmos aleatórios para a construção da rede sobreposta. Cada nó possui uma lista de vizinhos, que em geral são escolhidos aleatoriamente, mas cada aplicação define como a seleção é feita. Há aplicações que quando um nó deseja ser admitido na rede sobreposta, ele primeiro deve consultar o nó fonte, que por sua vez entrega uma lista aleatória ou com base em alguma métrica de prováveis vizinhos para este nó. O nó deve se comunicar com esses prováveis vizinhos para verificar quem poderá ser de fato seu vizinho.

Além do fato da construção da rede ser aleatória, o mesmo ocorre com os dados, que também são armazenados de maneira aleatória em cada nó. Desta forma, quando um nó necessita de um dado, em geral, este nó inunda a rede com pacotes de busca. O *BitTorrent* (Hales e Patarin, 2005) e o *Gnutella* (GNU2000) são exemplos de sistema que utilizam redes P2P não estruturadas.

É possível perceber que localizar um dado relevante em um sistema P2P não estruturado pode se tornar difícil à medida que a rede cresce, pois como não existe um roteamento determinístico para localizar o dado, o uso da técnica de inundação pode se tornar ineficiente. Uma alternativa para este problema nas redes P2P não estruturadas seria a utilização de nós especiais que mantenham um índice mapeando os dados.

Em outros casos é interessante abandonar a natureza simétrica dos sistemas P2P e considerar uma colaboração diferenciada entre os nós da rede. Por exemplo, em uma rede de entrega de conteúdo (*Content Delivery Network* – CDN), os nós podem oferecer armazenamento para hospedar cópias de páginas *web*, permitindo que clientes *web* acessem páginas próximas e que, por isso, esse acesso seja rápido. Nesse caso, um nó *N* pode precisar buscar recursos em uma parte específica da rede. Se isso acontecer, usar um intermediário que coleta utilização de recurso para vários nós que estão nas proximidades uns dos outros permitirá a rápida seleção do nó que tenha recursos suficientes (Tanenbaum e Steen, 2007). Esses tipos de nós expostos, mantenedor de índices ou nós intermediários, em geral são denominados superpares (*superpeers*).

1.2 Topologias para distribuição de vídeo na Internet

A distribuição de vídeo pela Internet é uma opção alternativa a meios mais tradicionais, como distribuição por *broadcast* via satélite ou a cabo, para se visualizar programas de TVs e outros tipos de conteúdos de vídeo. A Figura 6 apresenta uma taxonomia das tecnologias alternativas. Uma das primeiras alternativas para essa distribuição era baseada na arquitetura cliente-servidor, que como foi visto, tem sérios problemas de limitação para escalabilidade, dado que o número de usuários só crescia. Esta arquitetura ainda é utilizada.

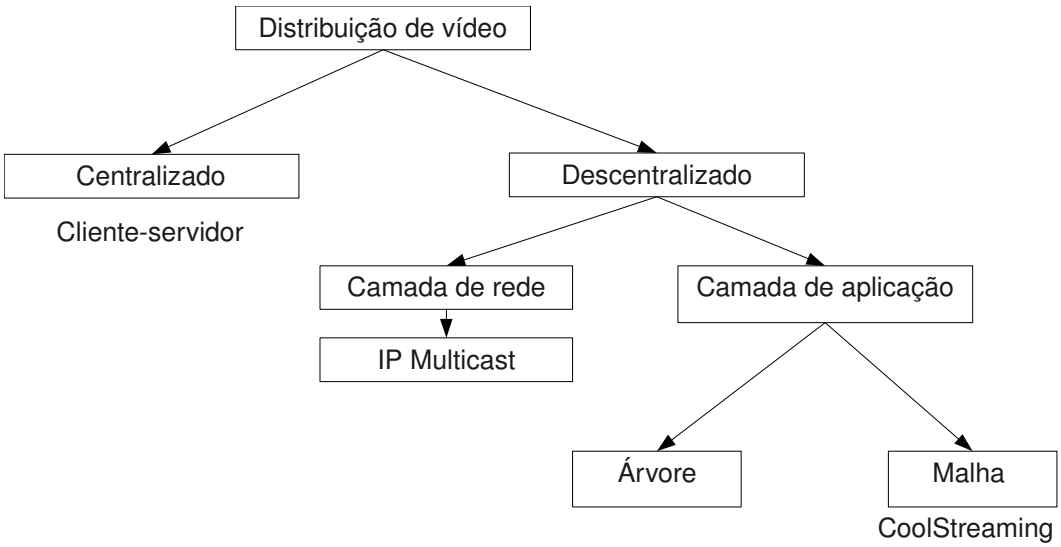


Figura 6: Distribuição de vídeo na Internet.

Uma outra opção que surgiu foi a distribuição de vídeo por multidifusão baseada na camada de redes, que como também já foi visto, tinha como desvantagem o alto custo operacional e financeiro. Uma alternativa possível é a multi-difusão baseada na camada de aplicação, onde as redes P2P se encaixam. Como foi visto, as redes P2P conseguem resolver o problema da escalabilidade e possuem um custo operacional baixo. A Figura 6 apresenta um resumo das alternativas atuais para distribuição de vídeo na Internet.

Mesmo nas redes P2P existem sub-divisões quando o assunto é a distribuição eficiente de fluxos de vídeo. Neste contexto, as redes sobrepostas podem ser divididas nas seguintes topologias: árvore e malha, discutidas nas próximas subseções.

1.2.1 Topologia em árvore

A Figura 7 mostra um exemplo de rede P2P com topologia em árvore. Nesta arquitetura, os nós se organizam na forma de uma ou múltiplas árvores, onde o nó fonte do fluxo de vídeo representa a raiz da árvore. Existe uma relação de pai e filho entre os nós, onde cada nó filho só pode possuir um nó pai. Um nó pai pode ter vários nós filhos, e um nó pode ter funções de pai e filho. O fluxo de vídeo sempre segue do nó pai para o nó filho, e nunca o contrário. A vantagem desta arquitetura é que possui um baixo nível de *overhead* no

gerenciamento, tornando-se um pouco mais próxima dos métodos de multidifusão por rede, o que garante um melhor desempenho na entrega dos pacotes. As grandes desvantagens são: a alta vulnerabilidade quanto a dinâmica da rede, pois a saída de um nó pode significar a interrupção momentânea do vídeo nos outros nós, e a baixa utilização dos recursos. Exemplos de aplicações de distribuição de vídeo que utilizam esta arquitetura: *ALMI* (Pendarakis et al, 2001), *NICE* (Banerjee et al, 2002) e *ZigZag* (Tran et al, 2003) (Utilizam uma única árvore), *SplitStream* (Castro, 2003) e *Bullet* (Kostic et al, 2003) (Múltiplas árvores).

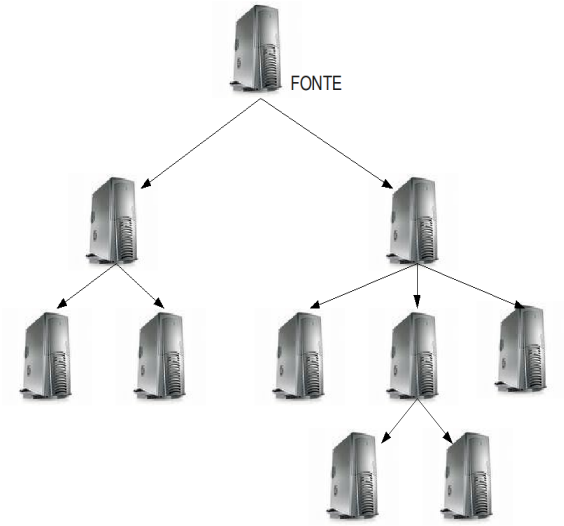


Figura 7: Topologia em árvore.

1.2.2 Topologia em malha

Já na arquitetura em malha, os nós formam uma malha de distribuição e compartilham pedaços de um vídeo que estão espalhados pela rede (Moraes et al, 2008). Nesta arquitetura, qualquer nó pode receber ou encaminhar os pedaços, e a forma como esses pedaços são encaminhados na malha depende do protocolo de difusão de pedaços escolhido.

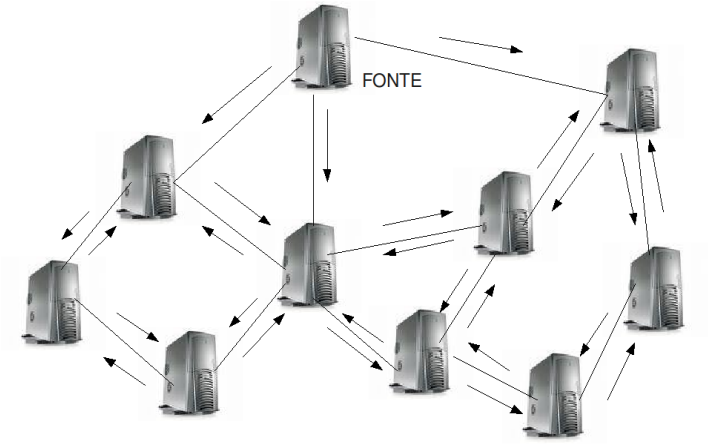


Figura 8: Topologia em malha.

As maiores vantagens desta solução são a alta utilização dos recursos e a rápida descoberta de novos nós, fazendo com que a dinâmica da rede com a entrada e saída de nós não seja tão prejudicial a manutenção do serviço. Podemos citar como exemplos de aplicações de distribuição de vídeo que utilizam este tipo de arquitetura o *Coolstreaming* (Xinyan et al, 2005), o *Promise* (Hefeeda et al, 2003) e o *GNUStream* (Jiang et al, 2003). A Figura 8 apresenta uma rede com topologia em malha.

1.3 Métodos de difusão de pedaços em redes P2P

Conforme foi visto, uma das formas de se transmitir um fluxo de vídeo em redes P2P de malha pela Internet é dividindo o seu conteúdo em pedaços a fim de transmiti-los de maneira independente uns dos outros. Para sistemas que utilizam este método de transmissão do conteúdo, um ponto tem grande relevância, o mecanismo de como é feito a difusão desses pedaços entre os nós da rede P2P.

Basicamente, pode-se dizer que existem três modelos de referência na literatura para o mecanismo de difusão dos pedaços, que são o modelo *push*, o modelo *pull*, e o modelo baseado no estado de cada nó.

No modelo *push*, os pedaços são enviados do nó pai para os nós filhos, sem o nó pai questionar se o nó filho precisa ou não daquele pedaço. Caso o nó filho, possua mais de um

nó pai, fica fácil perceber que o mesmo pedaço pode chegar várias vezes ao mesmo nó, ao passo que um determinado pedaço pode nunca chegar. O método *push* é, portanto, mais indicado em redes estruturadas, com topologia baseada em árvore (Pendarakis et al, 2001) (Banerjee et al, 2002) (Tran et al, 2004) (Castro, 2003) (Venkataraman e Francis, 2006) (Sung et al, 2006).

O modelo *pull* é o oposto do modelo *push*. Neste, o nó filho faz a requisição de um pedaço ao nó pai sem saber se ele o possui. Aqui não há problema de duplicação como no modelo *push*, mas existe o problema de *starvation*, pois um filho pode nunca encontrar um nó pai que possua o pedaço que ele esteja procurando. Em geral, o modelo *pull* está associado a sistemas não estruturados, onde um nó filho pode ser suprido por vários nós pai, diminuindo assim a probabilidade de ocorrer *starvation*. Na prática é difícil existirem sistemas que utilizem exclusivamente o modelo *pull*, em geral ele está associado a um mecanismo de trocas de informações entre os nós.

Já no modelo baseado no estado (Chu et al 2000)(Pai et al, 2005)(Xinyan et al, 2005) (Pianese et al, 2006), os nós trocam informações entre si para saber o que cada nó vizinho possui, e assim pedir-lhe ou enviar-lhe um determinado pedaço. Cada nó possui o seu mapa de *buffer* que ilustra quais pedaços do conteúdo o nó possui e ainda apresenta quais pedaços estão na eminência de serem reproduzidos, a representação pode ser *0* para pedaço ausente e *1* para pedaço disponível.

Os nós fazem a troca do mapa de *buffer* com os seus nós vizinhos, assim os mesmos serão capazes de saber quais vizinhos possuem os pedaços que ainda estão faltando. Como os nós têm consciência do que cada um possui, eles podem fazer a requisição de pedaços de maneira inteligente. Como exemplo as seguintes políticas de seleção de pedaços podem ser citadas:

- O mais raro primeiro → O algoritmo assume que é melhor um nó requisitar primeiro o pedaço que uma quantidade menor de nós vizinhos possui, pois esse pedaço em tese é mais difícil de se obter. Entre um conjunto de pedaços que um nó vizinho *V* possua e um nó *N* não tenha, nó *N* requisita o pedaço mais raro primeiro ao nó *V*.

- Em ordem → O nó requisita ao seu vizinho, o primeiro pedaço que ele não possui. Entre um conjunto de pedaços que um nó *V* possua e um nó *N* não tenha, nó *N* requisita o primeiro pedaço que não possui em primeiro lugar.

- Zipf(θ) (Carlsson e Eager, 2007)→ Combina o algoritmo mais raro com o algoritmo em ordem, assumindo diferentes probabilidades para cada um, pois o algoritmo admite que talvez não vale tão a pena perder um tempo grande aguardando um pedaço raro, ao passo que outros menos raros acabam estourando o tempo de reprodução. Para cada nova requisição, um nó *N* utiliza a seleção "baseado na ordem de reprodução" com uma probabilidade *p* e a seleção "o mais raro primeiro" com uma probabilidade (*1 -p*).

Alguns trabalhos propõem modelos híbridos para a distribuição de pedaços. Um desses trabalhos é o modelo *Interleave* (Sanghavi et al, 2007) que combina os modelos *push* e *pull*, e adiciona um mecanismo particular de política de seleção de pedaços, sem nenhuma troca de informações entre os nós sobre os pedaços que cada um possui. Este modelo foi projetado e analisado principalmente para transferência de arquivos, ele considerava que todos os nós da rede P2P eram homogêneos e sincronizados, isto é, o tempo de transmissão de um simples pedaço era o mesmo para todos os nós. Um desafio importante era provar se o modelo *Interleave* possuía bom desempenho em uma rede P2P não homogênea e dessincronizada. O trabalho também especulava que o modelo poderia ser utilizado para distribuição de vídeo, entretanto não houve investigação e consequente comprovação disto.

O trabalho (Cigno et al, 2008) analisou a performance do modelo *Interleave* em detalhe, e ampliou seu escopo para permitir redes P2P com comportamentos diversos e para comprovar se realmente era possível transmitir vídeo com este modelo de difusão de pedaços. Os resultados mostraram que era possível não só ter boa performance para distribuição de arquivos, como também para transmissão de vídeo. Este resultado, segundo o autor, mostra que um simples modelo que combine os modelos *push* e *pull*, sem manter quaisquer informações sobre os pedaços, pode obter boa performance para transmissão de vídeo. Diante deste fato, resolvemos continuar este estudo e fazer outras análises em cima deste protocolo.

1.3.1 O modelo de difusão de pedaços *Push/Pull*

O modelo de rede P2P para distribuição de vídeo proposto em (Cigno et al, 2008) possui apenas uma única fonte, que divide o conteúdo do vídeo em pedaços para serem distribuídos entre os nós da rede sobreposta de forma independente. A rede sobreposta é do tipo não-estruturada, com topologia em malha e simétrica, ou seja, se um nó *N* é vizinho de *V*, então *V* é vizinho de *N*. Os pedaços são gerados a uma taxa constante pela fonte e cada pedaço possui um identificador único que reflete a ordem de criação do mesmo.

Neste modelo, todos os nós da rede são criados ao mesmo tempo, entretanto um nó só é considerado ativo pelo protocolo de distribuição a partir do momento que recebe um pedaço de vídeo inteiro. Até então, o nó fica na rede sobreposta como um nó passivo e não pode pedir por nenhum pedaço. A partir do momento que recebe o primeiro pedaço, o nó é capaz de enviar e pedir pedaços.

Cada nó da rede tem na sua configuração uma lista de contatos, que é um número finito de vizinhos. Esta lista de contatos de um nó *N* pode ser definida como um conjunto de nós que *N* pode contactar ativamente, isto é, o nó *N* pode contactar apenas nós de sua lista de contatos. A lista apenas sofre manutenção no momento que nós saem e entram na rede sobreposta com o objetivo de manter o número de vizinhos selecionados constante, então pode-se afirmar que a lista é estática a menos que haja uma entrada ou saída de um nó.

O grafo obtido é simétrico, ou seja, um nó tanto pode enviar para um vizinho quanto pode receber do mesmo vizinho. No modelo estudado para distribuição de vídeo não há possibilidade de grafos assimétricos, embora o simulador forneça esta possibilidade. Além disso, cada nó alterna seu estado entre os modos *push* e *pull*, apenas o nó fonte não alterna o seu estado, ele permanece sempre com o estado *push*, pois o nó fonte não precisa requisitar pedaços, ele já os possui.

Durante o estado *push*, um nó *N* realiza as seguintes atividades:

1. seleciona um vizinho *V* da sua lista de vizinhos de maneira aleatória, e o pedaço *K* com o identificador mais alto, ou seja o pedaço mais recente, entre os diversos pedaços recebidos via estado *push* de seus vizinhos;

2. em seguida, envia uma mensagem ao vizinho *V* selecionado aleatoriamente, questionando se este deseja *K*;
3. se o nó *V* não tiver o pedaço *K* e tiver banda de *download* disponível, o nó *V* envia uma mensagem a *N* aceitando a oferta, e o nó *N* envia o pedaço *K* ao nó *V*;
4. caso contrário, o nó *N* aborta o processo.

Durante o estado *pull*, um nó *N* realiza as seguintes atividades:

1. envia uma mensagem de requisição de pedaço a um nó vizinho *V* aleatório, solicitando o pedaço *K* com o identificador mais baixo que não possua;
2. se o nó *V* possuir o pedaço *K* e tiver banda de *upload* disponível, o nó *V* envia uma mensagem ao nó *N* aceitando a requisição;
3. caso contrário, o nó *V* rejeita a requisição.

A política de seleção de pedaços é uma parte muito importante em sistemas de distribuição de vídeos, pois cada pedaço tem um tempo máximo de retardo a ser suportado para iniciar a sua reprodução. Uma seleção de pedaços aleatória funciona bem para transferência de arquivos, mas não funciona para transmissão de vídeo, pois na transferência de arquivos, o usuário só pode consumir o produto final após o mesmo ter chegado inteiramente, ou seja, todos os pedaços. Neste caso, não importa a ordem de chegada dos pedaços.

Já para transferência de vídeo, o requisito é outro, o usuário pode consumir o produto final antes dele chegar inteiramente, o usuário consome aos poucos e neste caso a ordem de chegada é fundamental. Se um pedaço de fluxo de vídeo chegar após o seu tempo de reprodução, haverá uma descontinuidade no vídeo, a tela provavelmente ficará congelada no último quadro reproduzido, causando um incômodo ao usuário.

Algoritmos inteligentes para selecionar os pedaços, tal qual o mais raro primeiro, não podem ser implementados apenas com modelos *push/pull*, pois conforme foi visto, é necessário conhecimento prévio do que os outros nós possuem. O modelo sugerido por

(Cigno et al, 2008) adotou a mesma política de seleção de pedaços adotada em (Sanghavi et al, 2007):

1.Durante o estado *push*, o nó *N* envia o pedaço com o identificador mais alto entre os diversos pedaços recebidos via estado *push* de seus vizinhos.

2.Durante o estado *pull*, o nó *N* requisita o pedaço com o identificador mais baixo que não possua. Essa política ajuda o nó *N* a preencher buracos na sequência de pedaços.

É possível observar que essa política de seleção não necessita de um prévio conhecimento do que outro nó possui.

Um nó muda de um estado para outro, ou depois de uma requisição ser aceita e o pedaço correspondente terminar de ser transferido, ou depois de receber um número máximo de negações às suas requisições. Este comportamento mantém o sistema rodando suavemente, e evita *starvation.*

O tempo gasto nos estados *push* ou *pull* depende do sucesso das requisições e da disponibilidade de banda, tanto do vizinho quanto do nó requisitante, logo os intervalos para o estado *push* e o estado *pull* não são fixos, e os nós são dessincronizados. Além disso, as informações trocadas pelos nós vizinhos são as mínimas possíveis, apenas mensagens de *keep-alive* e mensagens de requisições e respostas do protocolo *push/pull*.

A Figura 9 apresenta dois pseudo-códigos do algoritmo que resume as operações do protocolo *push/pull*. O primeiro código está relacionado às requisições enviadas aos nós vizinhos, o segundo são as respostas.

Os parâmetros de entrada do algoritmo relacionado às requisições são o *max_push_attempts* e o *max_pull_attempts*. O primeiro corresponde ao número máximo de tentativas que um nó pode fazer para enviar um pedaço durante o estado *push*, e o segundo corresponde ao número máximo de tentativas que um nó pode fazer para solicitar um pedaço durante o estado *pull*.

Durante o estado *pull*, o nó verifica se o número de tentativas atual é menor que o número máximo permitido e se há banda de *download* disponível. Se as duas premissas forem verdadeiras, o nó seleciona um vizinho de maneira aleatória e o pedaço com o menor ID que não possua, envia uma mensagem *"PULL"* ao vizinho selecionado e aguarda por uma resposta. Se a resposta for positiva e, novamente, se o nó tiver banda de *download* disponível,

o nó envia uma mensagem "*READY*" e inicia o processo de recebimento do pedaço. Caso contrário, o nó envia uma mensagem "*BUSY*".

Durante o estado *push*, o nó verifica se o número de tentativas atual é menor que o número máximo permitido e se há banda de *upload* disponível. Se as duas premissas forem verdadeiras, o nó seleciona um vizinho de maneira aleatória e o pedaço com maior ID que possua, envia uma mensagem "*PUSH"* e aguarda por uma resposta. Se a resposta for positiva e novamente se o nó tiver banda de *upload* disponível, o nó envia uma mensagem "*READY*" e inicia o processo de transferência do pedaço. Caso contrário, o nó envia uma mensagem "*BUSY*".

O código relacionado às respostas basicamente verifica qual é o tipo de mensagem recebida pelo nó vizinho. Se o nó receber uma mensagem do tipo "*PULL*", o mesmo verifica se tem o pedaço que o nó vizinho solicitou, e se tem banda de *upload* disponível. Se tiver, o nó envia uma mensagem de aceitação e aguarda por uma resposta. Após receber a mensagem "*READY*", o nó inicia a transferência do pedaço. Caso contrário, o nó rejeita a solicitação feita pela mensagem "*PULL*".

Se o nó receber uma mensagem do tipo "*PUSH*", o mesmo verifica se realmente não possui o pedaço, e se possui banda de *download* disponível. Caso passe pelos dois pré-requisitos, o nó envia uma mensagem de aceitação e aguarda por uma mensagem "*READY*" e inicia o processo de recebimento do pedaço. Caso o nó não preencha os pré-requisitos, o nó rejeita a solicitação feita pela mensagem "*PUSH*".

```
Algoritmo – Requisições
Entrada: Número máximo de tentativas para o estado push e para o estado pull:
max_push_attempts, max_pull_attempts

se (estado == PULL) então
       enquanto (pull_attempt < max_pull_attempts) E (banda de download
       disponível) faça
              Selecionar um vizinho e o pedaço com menor ID que não possua;
              Enviar uma mensagem PULL;
              Esperar por resposta;
              se (resposta == ACCEPT_PULL) então
                     se (banda de download disponível) então
                            Enviar mensagem READY;
                            Iniciar o download do pedaço;
                            parar; /* sair do enquanto
```

```
                    senão
                        Enviar mensagem BUSY;
                    fim
                fim
                pull_attempt++;
        fim
        estado = PUSH;
        pull_attempt = 0;
fim
se (estado == PUSH) então
        enquanto (push_attempt < max_push_attempts) E (banda de upload
        disponível) faça
                Selecionar um vizinho e o pedaço com maior ID que possua;
                Enviar uma mensagem PUSH;
                Esperar por resposta;
                se (resposta == ACCEPT_PUSH) então
                    se (banda de upload disponível) então
                        Enviar mensagem READY;
                        Iniciar o upload do pedaço;
                        parar; /* sair do enquanto
                    senão
                        Enviar mensagem BUSY;
                    fim
                fim
                push_attempt++;
        fim
        estado = PULL;
        push_attempt = 0;
fim
```

```
Algoritmo – Respostas
se (mensagem ==PULL) então
        se (pedaçoID disponível) E (banda de upload  disponível) então
                Enviar mensagem ACCEPT_PULL;
                Esperar por resposta;
                se (resposta == READY) então;
                    Iniciar o upload do pedaço;
                fim
        senão
                Enviar mensagem REFUSE_PULL;
        fim
fim

se (mensagem == PUSH) então
```

```
        se (pedaçoID faltando) E (banda de download  disponível) então
                Enviar mensagem ACCEPT_PUSH;
                Esperar por resposta;
                se (resposta == READY) então;
                        Iniciar o download do pedaço;
                fim
        senão
                Enviar mensagem REFUSE_PUSH;
        fim
fim
```

Figura 9: Pseudo-código do algoritmo *push/pull* (Cigno et al, 2008).

A maior vantagem de se utilizar protocolos de difusão baseado nos modelos *push/pull* é a sua simplicidade, relacionada ao fato deste poder trabalhar sem qualquer suposição em relação ao comportamento de cada nó. Além disso, nenhuma sinalização é necessária para coordenar os nós, fazendo com que este protocolo seja, em tese, adequado a redes muito dinâmicas, ou seja, com um volume grande de entrada e saída de nós. Por outro lado, protocolos baseado em estado buscam melhorar o desempenho e a eficiência, em detrimento da simplicidade, aumentando o risco de se tornar frágil e propenso a falhas em redes heterogêneas e dinâmicas, com nós com comportamento variável.

Para alguns casos, como transmissão de vídeo por difusão, onde, em geral, todos os usuários estão interessados no mesmo recurso em um espaço curto de tempo, talvez um protocolo mais simples seja suficiente para atender todos os requisitos do serviço de vídeo de maneira eficiente. Para outros casos, como transmissão de vídeo sob demanda, onde os usuários interessados no vídeo estão em diferentes momentos de reprodução do mesmo, provavelmente um protocolo baseado em estado seja mais eficaz, pois cada usuário saberá exatamente a quem recorrer para obter os pedaços de vídeo desejados, sem correr o risco de pedir para um usuário que não o tenha.

CAPÍTULO 2 - PROTOCOLO *PUSH/PULL* E MECANISMOS DE INCENTIVO PARA DISTRIBUIÇÃO DE VÍDEO

Este capítulo apresenta os trabalhos relacionados ao protocolo de difusão de pedaços *push/pull*, à distribuição de fluxo vídeo em redes P2P e aos mecanismos de incentivo à cooperação. Além disto, é feita uma primeira análise simplificada do desempenho do protocolo *push/pull* diante de nós não-cooperativos para avaliar se mudanças no modelo são realmente significativas.

2.1 Trabalhos relacionados a *push/pull* e distribuição de vídeo

Existem vários trabalhos relacionados ao uso de redes P2P para distribuição de vídeo (Carlsson e Eager, 2007) (Cigno et al, 2008) (Habib e Chuang, 2004) (Moraes et al, 2008) (Moraes, 2009) (Tran et al, 2003) (Xinyan et al, 2005). Nenhum deles, entretanto, faz um estudo específico sobre o comportamento de uma uma rede que utilize o protocolo de difusão *push/pull* face a presença de nós com comportamento não-cooperativo.

(Sanghavi et al, 2007) propôs o protocolo *Interleave* para difundir os pedaços entre os nós da rede. Este protocolo combina os modelos *push* e *pull*, alternando os estados dos nós entre um ou outro, e foi criado com o propósito de aumentar o desempenho de serviço de distribuição de arquivos. O cenário utilizado para estudar o novo protocolo era composto por nós homogêneos e sincronizados, onde a fatia de tempo para receber um pedaço do arquivo era a mesma para todos os nós. Além de alternar entre o estado *push* e *pull*, o autor adotou uma política de seleção de pedaços, onde durante o estado *push*, o nó *N* enviava o pedaço com o identificador mais alto entre os diversos pedaços recebidos via estado *push* de seus vizinhos. E durante o estado *pull*, o nó *N* requisitava o pedaço com o identificador mais baixo que não possuía. Essa política ajudava o nó *N* a preencher buracos na sequência de pedaços. Com essa política de seleção não havia necessidade de um prévio conhecimento do que cada nó possuía. O trabalho comprovou que o protocolo podia ser utilizado de forma satisfatória para serviço

de distribuição de arquivos, além disso, especulava que o protocolo poderia ser utilizado para distribuição de vídeo, entretanto não houve investigação e consequente comprovação disto.

Alguns trabalhos propõem a combinação dos modelos *push* e *pull*, utilizando o mecanismo *push* para distribuir rapidamente os pedaços, e o mecanismo *pull* para preencher os "buracos" no *buffer* de reprodução. Não há uma alternância entre os dois modelos. Por exemplo, (Locher et al, 2007) desenhou um modelo para distribuição de vídeo baseado em dois conceitos. O primeiro conceito define que a rede sobreposta é do tipo estruturada. O segundo conceito define o protocolo de distribuição dos dados na rede. O protocolo introduz uma combinação entre as operações *push* e *pull*. O objetivo principal do protocolo é distribuir o maior número possível de dados para um determinado número de vizinhos utilizando a operação *push*, e o que restar, utilizar operações do tipo *pull*. Segundo o autor, é preferível utilizar este método porque o uso de modelos puramente baseado em operações *pull*, traria longos retardos à reprodução.

Mais próximo de nosso trabalho, (Cigno et al, 2008) fez um estudo do protocolo *push/ pull* com a finalidade de avaliar o comportamento e desempenho do mesmo na tarefa de distribuição de fluxo de vídeo na Internet. Para atingir seus objetivos, variou parâmetros importantes da rede, como: quantidade de nós, quantidade de vizinhos por nó, quantidade de pedaços a serem distribuídos e banda de *upload*. O autor utilizou dois diferentes cenários para avaliar o protocolo. No primeiro cenário, o ambiente era simplificado, todos os nós eram homogêneos e sincronizados, o simulador era orientado a ciclos. Desta forma, em uma determinada fatia de tempo, todos os nós ficavam no estado *push*, na outra fatia, todos os nós ficavam no estado *pull*. Segundo o autor, este modelo serviu para comparar com o modelo utilizado em (Sanghavi et al, 2007).

No segundo cenário, que o autor chamou de modelo realista, os nós eram dessincronizados, a banda de download limitada e o simulador era orientado a eventos. Para simular os dois cenários utilizando o protocolo de difusão *push/pull* para distribuição de pedaços, o autor criou um novo módulo chamado *P4S* para o simulador *PeerSim* e realizou várias simulações com diferentes tamanhos de redes (100, 500 e 1000 nós), diferentes pedaços (1000, 5000 e 10000). Para avaliação dos modelos, o autor utilizou as métricas tempo máximo e máximo retardo. Os resultados apresentaram indícios de que era possível ter bom desempenho para transmissão de vídeo. E segundo o autor, nunca havia sido mostrado que um

simples modelo que combinasse os modelos *push* e *pull*, pudesse obter bom desempenho para transmissão de vídeo.

2.2 Mecanismos de incentivo em sistemas de distribuição de vídeo

Até este momento as discussões apresentadas consideraram que todos os nós atuam de forma cooperativa na rede. Grande parte dos sistemas P2P conta com isto, já que deveria ser do interesse do nó participante cooperar com os demais, pois a qualidade dos serviços que ele utiliza depende de um bom desempenho da rede como um todo.

Estudos mostram que em sistemas de compartilhamento de arquivos, o tempo total para baixar arquivos e a taxa de falhas de toda a rede aumentam a medida que os nós deixam de compartilhar seus recursos (Adar e Huberman, 2000). Em redes sem fio *Ad hoc*, a latência de pacotes e a taxa de perda aumentam para todos, quando os nós se recusam a encaminhar pacotes de controle sobre o comportamento dos outros nós (Marti et al, 2000).

Diante destes fatos, seria natural raciocinar que é mais interessante para um determinado nó cooperar para maximizar os seus resultados. Mas, isso não é o que ocorre, de fato. Estudos têm mostrado que nós que atuam como parasitas nas redes, consumindo recursos sem contribuir, são maioria em redes P2P para compartilhamento de arquivos (Adar e Huberman, 2000) (Saroiu et al, 2003).

Mas por que os usuários preferem se comportar assim? Já que deste modo, além de prejudicar a rede como um todo, acaba prejudicando a si próprio? Talvez porque os usuários não tenham este entendimento e pensem que compartilhar seus recursos (banda, CPU, memória, disco e outros) pode degradar o seu próprio desempenho. O usuário poderia pensar diferente se soubesse que o sistema possui um mecanismo de incentivo que premiasse quem cooperasse mais e punisse quem cooperasse menos.

Não apenas usuários egoístas ou parasitas prejudicam o desempenho de uma rede P2P. Usuários mal intencionados, que querem realmente degradar o serviço oferecido pela rede, ou usuários mentirosos, que querem trapacear, também prejudicam o desempenho da rede.

Nesse contexto, é importante que o sistema P2P esteja protegido destes tipos de nós, que possuam estes maus comportamentos. O desempenho do sistema deveria se manter bom e estável, mesmo diante de nós não-cooperativos, ou seja, o sistema deve ser robusto o suficiente para não ser vulnerável a este tipo de comportamento.

Existem vários trabalhos relacionados ao uso de redes P2P para distribuição de vídeo, que exploram esse problema (Habib e Chuang, 2004) (Adar e Huberman, 2000) (Feldman et al, 2004) (Hoong e Matsuo, 2008)

(Habib e Chuang, 2004) propõe um mecanismo de seleção de vizinhos baseado em um *ranking* para distribuição de vídeo na Internet com nós com interesses assimétricos. O mecanismo estimula o incentivo à cooperação através da diferenciação de serviços. Nós cooperativos da rede são recompensados com flexibilidade e escolha na seleção dos vizinhos, resultando em alta qualidade na reprodução do vídeo. Os nós não-cooperativos têm limitadas opções na seleção de vizinhos, isso quando consegue selecionar, obtendo uma baixa qualidade de vídeo. Após as simulações e medições reais, os autores verificaram que o mecanismo pode prover qualidade de vídeo próxima ao ótimo, quando todos os nós são cooperativos, desde que as fontes de vídeo não esteja próximas ao gargalo. O mecanismo dos autores permite que usuários selecionem vizinhos com pontuação igual ou menor à dele. Além disso, usuários recém chegados inicializam com pontuação zero e recebem um serviço de “melhor esforço”. Isso evita que nós não-cooperativos repetidamente se mascarem como um usuário novo para melhorar sua pontuação.

Os autores utilizaram o simulador ns-2 (NS-2) com o módulo *PROMISE* (Hefeeda et al, 2003). Os resultados mostraram que quando a rede está disponível, o sistema com ou sem mecanismo obtém bons resultados, e se a carga da rede aumenta muito, a qualidade deteriora com ou sem mecanismo. Para um número alto de sessões nas fontes, a qualidade mostra-se baixa para ambos os casos porque a rede fica muito congestionada. Para estes casos, a cooperação dos usuários não melhora o desempenho do sistema. Logo, os autores afirmam que o mecanismo de incentivo não é necessário quando a rede está totalmente disponível e não é efetiva quando o congestionamento é alto.

(Chu et al, 2004) propõe um modelo de taxação para distribuição de vídeo, onde nós com mais recursos contribuem com mais banda no sistema. Nós com recursos limitados são subsidiados pelo sistema. O modelo é aplicado no contexto de distribuição de vídeo porque o

distribuidor do vídeo impõe a taxação nos nós para obter o máximo de cooperação de cada um. Este modelo é usual para ambiente de distribuição onde um número alto de usuários estejam interessados em uma mesma sessão de vídeo, e os usuários querem cooperar de maneira síncrona com outros usuários que querem receber o mesmo conteúdo.

(Golle et al, 2001) propõe um mecanismo baseado em pagamento para distribuição de arquivos, onde cada usuário ganha recompensa se enviar conteúdo para outros usuários. A recompensa pode ser usada para um futuro *download*. O objetivo do mecanismo é alcançar a máxima cooperação entre usuários.

(Hoong e Matsuo, 2008) propõe o protocolo PALMS (*P2P Unstructured Live Media Streaming*) que faz uso dos modelos *push*, *pull* e conhecimento do estado do mapa de *buffer* de cada nó para difusão dos pedaços de vídeo em conjunto com um mecanismo de incentivo à cooperação baseado na pontuação do nó para que a rede obtenha um melhor desempenho. A pontuação é função da razão entre *bytes* transmitidos e *bytes* recebidos. A seleção do vizinho depende da pontuação do nó solicitante e do nó candidato a vizinho. Um nó solicitante só pode selecionar nós com pontuações menores ou iguais à dele. Os autores utilizaram o simulador ns-2 e obtiveram resultados satisfatórios para a qualidade de reprodução de vídeo, mesmo diante da existência de nós não-cooperativos na rede.

2.3 Análise do desempenho do protocolo *push/pull* diante de nós não-cooperativos

Para verificar se um estudo com relação ao desempenho do protocolo *push/pull* proposto em (Cigno et al, 2008) diante de nós não-cooperativos é relevante ou não, foi feita uma análise simples. Criou-se uma pequena rede em malha simétrica com 9 nós, sendo 8 nós comuns e um nó fonte. O nó fonte possui todos os pedaços (*k*) e apenas os envia, os nós comuns não possuem pedaços de início, e podem enviar e recebê-los. Para que um nó seja ativado nesta pequena rede, ele deve receber um pedaço. Para simplificar o exemplo, dado que um nó recebeu um pedaço no instante t_i, ele distribui para os seus vizinhos no instante t_{i+1}. Além disso, apenas 2 pedaços são distribuídos, a banda de cada nó é ilimitada e os nós ficam no estado *push (a maioria dos pedaços do protocolo push/pull são transmitidos via push)*. No

primeiro caso, todos os nós são cooperativos, nos casos seguintes há presença de nós não-cooperativos.

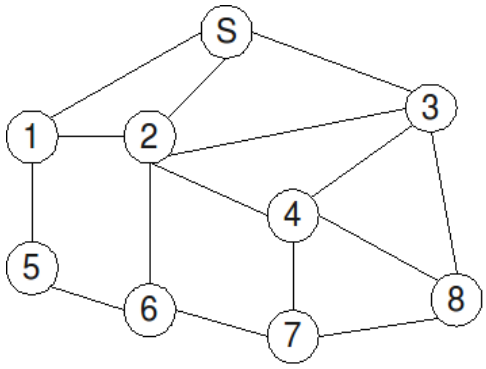


Figura 10: Rede sobreposta sem modificação.

A Figura 10 representa a rede sem modificação, a Figura 11 representa a rede sobreposta com alteração após o nó N5 detectar que é melhor escolher o nó N2 como parceiro ao invés do nó N1, e o nó N7 detectar que é melhor ter o nó N3 do que o nó N4.

Caso 1 – Rede sobreposta da figura 10 com todos os nós cooperativos

t_1 → S entrega k_1 para N_1, N_2 e N_3

t_2 → S entrega k_2 para N_1, N_2 e N_3; N_1 entrega k_1 para N_5; N_2 entrega k_1 para N_4 e N_6; N_3 entrega k_1 para N_8

t_3 → N_1 entrega k_2 para N_5; N_6 entrega k_1 para N_7; N_2 entrega k_2 para N_6; N_3 entrega k_2 para N_4 e N_8

t_4 → N_4 entrega k_2 para N_7.

Caso 2 – Rede sobreposta da figura 10 com todos os nós cooperativos, exceto N_1 e N_4.

t_1 → S entrega k_1 para N_1, N_2 e N_3

t_2 → S entrega k_2 para N_1, N_2 e N_3; N_2 entrega k_1 para N_4 e N_6; N_3 entrega k_1 para N_8

t_3 → N_6 entrega k_1 para N_5; N_8 entrega k_1 para N_7; N_2 entrega k_2 para N_4 e N_6; N_3 entrega k_2 para N_8

t_4 → N_8 entrega k_2 para N_7; N_6 entrega k_2 para N_5.

Caso 3 – Rede sobreposta da figura 10 com todos os nós cooperativos, exceto N_1, N_3 e N_6.

t_1 → S entrega k_1 para N_1, N_2 e N_3

t_2 → S entrega k_2 para N_1, N_2 e N_3; N_2 entrega k_1 para N_4 e N_6

t_3 → N_4 entrega k_1 para N_7 e N_8; N_2 entrega k_2 para N_4 e N_6

t_4 → N_4 entrega k_2 para N_7 e N_8.

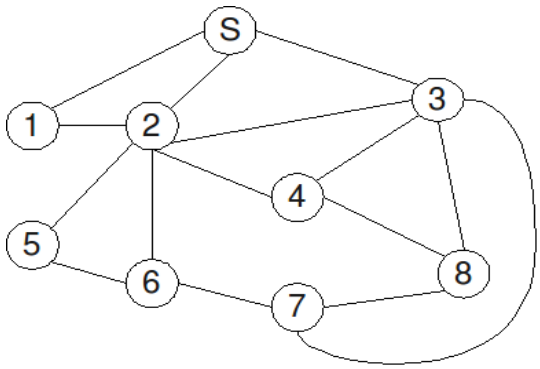


Figura 11: Rede sobreposta após modificação.

Caso 4 – Rede sobreposta da figura 11 com todos os nós cooperativos, exceto N_1 e N_4.

t_1 → S entrega k_1 para N_1, N_2 e N_3

t_2 → S entrega k_2 para N_1, N_2 e N_3; N_2 entrega k_1 para N_5 e N_6; N_3 entrega k_1 para N_4, N_7 e N_8

t_3 → N_2 entrega k_2 para N_4, N_5 e N_6; N_3 entrega k_2 para N_7 e N_8

Caso 5 – Rede sobreposta da figura 11 com todos os nós cooperativos, exceto N_1, N_3 e N_6.

t_1 → S entrega k_1 para N_1, N_2 e N_3

t_2 → S entrega k_2 para N_1, N_2 e N_3; N_2 entrega k_1 para N_4, N_5 e N_6

t_3 → N_4 entrega k_1 para N_8; N_6 entrega k_1 para N_7; N_2 entrega k_2 para N_4, N_5 e N_6

t_4 → N_4 entrega k_2 para N_8; N_6 entrega k_2 para N_7.

A Tabela 1 apresenta uma comparação entre os tempos finais de cada nó para os cinco casos diferentes. Através do resultado é possível observar que melhorias são alcançadas quando se altera um vizinho devido a não cooperação deste. É certo que foi utilizado uma rede muito pequena, e que não foram considerados problemas relacionado a banda, entretanto esta tendência é um bom começo para se investigar este aspecto no protocolo.

Tabela 1: Resumo comparativo entre os cinco casos estudados

Nós	Caso1	Caso2	Caso3	Caso4	Caso5
N_1	t_2	t_2	t_2	t_2	t_2
N_2	t_2	t_2	t_2	t_2	t_2
N_3	t_2	t_2	t_2	t_2	t_2
N_4	t_3	t_3	t_3	t_3	t_3
N_5	t_3	t_4	Não é ativado	t_3	t_3
N_6	t_3	t_3	t_3	t_3	t_3
N_7	t_4	t_4	t_4	t_3	t_4
N_8	t_3	t_3	t_4	t_3	t_4

O caso 4 apresentou melhor desempenho do que o caso 2, com ambos possuindo os mesmos nós não-cooperativos. Todos os nós do caso 4 terminaram de receber os dois pedaços k até o instante t_3, o que não ocorreu no caso 2. Já o caso 5 que possuía os mesmos nós não cooperativos que o caso 3, teve praticamente o mesmo desempenho em quase todos os nós, exceto o nó N_5, este no caso 3 não conseguiu sequer ser ativado, já no caso 5, o mesmo finaliza no instante t_3.

Outra observação importante a ser feita é que a distribuição dos pedaços teve um desempenho pior nos casos 2 e 3 em relação ao caso 1, onde nenhum nó era não-cooperativo. Mostrando que o protocolo é vulnerável a presença de nós não-cooperativos.

CAPÍTULO 3 - AVALIAÇÃO DO PROTOCOLO *PUSH/PULL*

Um dos pontos fundamentais deste trabalho trata do estudo do comportamento de uma rede sobreposta em malha que utilize o método de difusão de pedaços baseado no modelo push*/pull* diante de nós que podem ou não estarem dispostos a cooperar. O objetivo é mensurar o quanto a presença de nós maliciosos pode prejudicar o desempenho da rede P2P, principalmente quando o serviço ofertado depende tanto de requisitos rigorosos, como é o caso de aplicações de distribuição de vídeo.

Fazer este tipo de estudo em uma rede "viva" não é tarefa trivial, já que este tipo de experimento envolve uma escala grande de nós. Por isso, ao invés de utilizar uma rede "viva", optou-se por simular a rede sobreposta em um simulador de rede P2P.

Este capítulo apresenta informações sobre o simulador utilizado, *Peersim* (PeerSim) com o módulo P4S (P4S), mostra também as métricas utilizadas para mensurar o desempenho das redes e os resultados obtidos com seus respectivos comentários para o primeiro objetivo do trabalho. Também será visto um mecanismo de avaliação baseado na reputação do nó e seus respectivos resultados, esta segunda análise ainda faz parte do primeiro objetivo deste trabalho.

3.1 O simulador

A simulação foi feita no simulador *PeerSim (PeerSim)* utilizando o módulo P4S (P4S). O simulador *PeerSim* foi criado para lidar com redes sobrepostas de grande escala (milhões de nós) e suportar dinamismo, o *PeerSim* foi desenvolvido na linguagem Java com orientação a objeto, contendo módulos que podem ser modificados de forma independente uns dos outros. Essa propriedade faz com que protótipos de protocolos sejam criados, modificados e/ou estudados de maneira mais fácil e rápida, sem interferir nos demais módulos.

O *PeerSim* suporta dois modelos de simulação: o modelo orientado a ciclos e o modelo orientado a eventos. O primeiro modelo é mais simples que o segundo, alcança alta

escalabilidade e desempenho, com o custo de um pouco de perda de realismo. No modelo orientado a ciclos, não existe simulação da camada de transporte e nem concorrência, em outras palavras, os nós se comunicam entre si diretamente. Alguns protocolos conseguem tolerar esta perda de informação sem maiores problemas, outros não, então é necessário saber que tipo de modelo é mais adequado para cada tipo de experiência. Para este trabalho foi escolhido o modelo de simulação orientado a eventos, pois era preciso que o ambiente fosse o mais próximo possível da realidade.

O primeiro passo da simulação é a leitura do arquivo de configuração, onde os parâmetros da rede são passados, tais como: quantidade de nós na rede, número de pedaços, quantidade de vizinhos, porcentagem de nós não-cooperativos, porcentagem de vizinhos não-cooperativos, protocolos a serem ativados, bandas de *upload* e *download*. A seguir, o simulador inicializa os nós na rede e os protocolos envolvidos, responsáveis pela construção da rede sobreposta, pelo gerenciamento da banda nos nós ou pela difusão dos pedaços entre os nós por exemplo.

O grafo criado pode ser assimétrico, quando *N* é vizinho de *V*, mas não necessariamente *V* é vizinho de *N*. Na prática, *N* pode enviar pedaços para *V*, mas *V* não tem *N* na sua lista de vizinhos e portanto não envia para *N*. A classe *peersim.dynamics.WirekOut* é um exemplo de classe que implementa o grafo assimétrico. Há a possibilidade de criação de grafos simétricos, onde o nó *N* possui o nó *V* na sua lista de vizinhos e o nó *V* possui o nó *N* na sua lista também, assim, ambos podem enviar e receber para o outro.

Em nossas simulações utilizamos grafos simétricos. Para isso, utilizamos a classe *peersim.dynamics.WirekOutUnd*, a mesma utilizada pelo trabalho (Cigno et al, 2008) para distribuição de vídeo. Esta classe cria um grafo simétrico com cada nó possuindo um número fixo de vizinhos, e a escolha de cada vizinho é feita de forma aleatória.

Cada nó tem os mesmos tipos de protocolos. As instâncias de nós e protocolos são criados através de um processo de clonagem. Apenas uma instância é construída usando o construtor do objeto, que serve como protótipo, e todos os nós na rede passam a ser criados através deste protótipo. Após a inicialização da rede, os eventos são gerados ou cada ciclo com seus componentes é executado até que um componente de controle decida que a simulação chegou ao fim. Os dois controles utilizados para finalizar a simulação foram: todos os nós obtiveram todos os pedaços ou tempo limite esgotado.

Como o trabalho visa o estudo do protocolo de difusão baseado no modelo *push/pull*, optou-se por utilizar o módulo P4S, que foi criado para simular tanto distribuição de arquivos quanto distribuição de vídeo em redes P2P em malha. O módulo P4S criou a classe *p4s.core.Alternate* que simula um protocolo de difusão que combina os modelos *push* e *pull*, além disso, o P4S utiliza o módulo *Bandwidth Aware Protocol (*P4S) para gerenciamento de alocação de banda utilizando política *FIFO*, onde a primeira conexão consegue obter o máximo de recursos possíveis.

3.2 Simulação e resultados

3.2.1 Impacto causado ao protocolo *push/pull* devido à presença de nós não-cooperativos

A primeira parte da simulação analisa o impacto causado no desempenho do protocolo de difusão de pedaços *push/pull* proposto em (Cigno et al, 2008), devido a ação de nós com comportamento não-cooperativo na rede P2P. Antes de iniciar a simulação foi necessário criar nós com comportamento não-cooperativo, já que os nós que eram criados até então possuíam sempre o comportamento esperado pelo protocolo.

Para modelar um nó com comportamento não-cooperativo, modificamos apenas o algoritmo do estado *push* porque de acordo com os resultados de (Cigno et al, 2008) o número de pedaços recebidos via *push* é ligeiramente maior do que recebido via *pull*, com isso o nó não-cooperativo nesta simulação prejudica seus vizinhos porque não envia pedaços de forma voluntária aos mesmos, entretanto quando é solicitado a enviar um pedaço, o nó não-cooperativo age corretamente. Pode-se dizer então, que o nó não-cooperativo neste caso não é 100% prejudicial, não tem a finalidade de prejudicar totalmente a rede e sim de se beneficiar em relação aos outros nós, de certa forma age de maneira egoísta.

Para que fosse possível a adição de nós não-cooperativos durante o processo de criação da rede P2P foi necessário fazer uma alteração no simulador. Foi feita uma alteração na classe *peersim.core.GenericNode* com a adição de um atributo para informar se o nó é um nó cooperativo ou não-cooperativo. A classe *peersim.core.Network* também foi alterada para

que fosse possível inicializar na rede sobreposta nós não-cooperativos. Por fim, o algoritmo da classe *p4s.core.Alternate* responsável pelo modelo *push/pull* foi alterado para que nós não-cooperativos tivessem comportamento diferente de nós cooperativos durante o estado *PUSH*. A Figura 12 apresenta o pseudo-código do algoritmo *push/pull* modificado.

```
Algoritmo – Requisições
Entrada: Número máximo de tentativas para o estado push e para o estado pull:
max_push_attempts, max_pull_attempts
se (estado == PULL) então
       enquanto (pull_attempt < max_pull_attempts) E (banda de download
       disponível) faça
              Selecionar um vizinho e o pedaço com menor ID que não possua;
              Enviar uma mensagem PULL;
              Esperar por resposta;
              se (resposta == ACCEPT_PULL) então
                     se (banda de download disponível) então
                            Enviar mensagem READY;
                            Iniciar o download do pedaço;
                            parar; /* sair do enquanto
                     senão
                            Enviar mensagem BUSY;
                     fim
              fim
              pull_attempt++;
       fim
       estado = PUSH;
       pull_attempt = 0;
fim

se (estado == PUSH) então
       enquanto (push_attempt < max_push_attempts) E (banda de upload
       disponível) E (IsMalicius == False) faça
              Selecionar um vizinho e o pedaço com maior ID que possua;
              Enviar uma mensagem PUSH;
              Esperar por resposta;
              se (resposta == ACCEPT_PUSH) então
                     se (banda de upload disponível) então
                            Enviar mensagem READY;
                            Iniciar o upload do pedaço;
                            parar; /* sair do enquanto
                     senão
                            Enviar mensagem BUSY;
                     fim
              fim
              push_attempt++;
       fim
```

```
       estado = PULL;
       push_attempt = 0;
fim
```

```
Algoritmo – Respostas
se (mensagem == PULL) então
       se (pedaçoID disponível) E (banda de upload  disponível) então
              Enviar mensagem ACCEPT_PULL;
              Esperar por resposta;
              se (resposta == READY) então;
                     Iniciar o upload do pedaço;
              fim
       senão
              Enviar mensagem REFUSE_PULL;
       fim
fim

se (mensagem ==PUSH) então
       se (pedaçoID faltando) E (banda de download  disponível) então
              Enviar mensagem ACCEPT_PUSH;
              Esperar por resposta;
              se (resposta == READY) então;
                     Iniciar o download do pedaço;
              fim
       senão
              Enviar mensagem REFUSE_PUSH;
       fim
fim
```

Figura 12: Pseudo-código do algoritmo *push/pull* alterado.

Pode ser notado que a única diferença entre o algoritmo da Figura 12 para o algoritmo da Figura 9, já explicado na seção 1.3.1, é a adição de mais uma condição no laço do estado *push*. Agora, o nó que vai enviar voluntariamente um pedaço deve, além de possuir um número de conexões ativas menor que o número máximo de conexões possíveis e banda de *upload* disponível, ser também um nó cooperativo. Caso contrário, se o nó for não-cooperativo, o mesmo passa para o estado *pull* diretamente, sem enviar nenhum pedaço.

Era necessário verificar se o simulador alterado tinha o mesmo comportamento do simulador sem alteração quando a porcentagem de nós não-cooperativos fosse igual a zero. Este item foi comprovado, pois os resultados das simulações foram exatamente iguais nos dois simuladores, para redes P2P cuja porcentagem de nós não-cooperativos era zero.

Após esta “calibração” do novo simulador, inicializou-se as simulações das redes propostas. Inicialmente, estudamos o comportamento da rede variando-se a porcentagem de nós não-cooperativos e fixando-se outros parâmetros durante as rodadas de simulação.

A simulação foi feita da seguinte forma. A rede sobreposta é composta por um nó fonte que gera os pedaços a uma taxa constante, que pode ser, por exemplo, a taxa de um vídeo. Os nós são inicializados ao mesmo tempo e seus vizinhos são escolhidos de forma aleatória e simétrica, conforme já explicado. Os nós entram na rede como nós passivos, ou seja, não podem solicitar nenhum pedaço a um de seus vizinhos até que fiquem no modo ativo.

O nó torna-se ativo apenas após receber o primeiro pedaço. O trabalho (Cigno et al, 2008) não cita como essa forma de entrar na rede seria observada pelo usuário em termos práticos. Uma idéia seria o usuário ter um endereço de página *web* para acessar e fazer o pedido para adquirir o vídeo. Ao fazer o pedido o usuário receberia um aviso informando que sua solicitação seria processada, e quando tornar-se um usuário de fato ativo no sistema, receberia um aviso de usuário ativo, pronto para receber e enviar pedaços de vídeo. Esta discussão é importante para que se entenda que existe uma distinção entre o tempo de inicialização do nó, quando o usuário recebe o aviso de que sua solicitação está sendo processada, e o tempo de ativação do nó, quando o usuário recebe o aviso de que está pronto.

Após a inicialização dos nós, o nó fonte começa a enviar pedaços para os seus vizinhos, ativando-os. A partir daí, outros nós também são ativados e portanto ficam aptos a pedir e enviar pedaços. A simulação chega ao fim quando todos os nós obtiverem todos os pedaços ou o tempo limite estiver esgotado.

Dois tipos de vídeo foram escolhidos para serem distribuídos nas redes, um vídeo de entretenimento de 5 minutos e um vídeo educacional de 20 minutos. Os parâmetros desses vídeos foram retirados de (Costa et al, 2004) que utilizou arquivos de carga dos servidores da TV UOL e do servidor *eTeach*, um servidor de conteúdo educacional localizado na Universidade de Wisconsin, EUA. Esses dois vídeos também foram utilizados no trabalho de (Moraes, 2009).

Os seguintes parâmetros, resumidos na Tabela 2, foram utilizados nas duas fases de simulação. Assim como o trabalho (Moraes, 2009), assumiu-se que a taxa de reprodução dos

vídeos era igual a 350 kb/s e que um pedaço contém 10 s de vídeo. Com isso, um pedaço de vídeo tem tamanho igual a 437,5 kB. Os vídeos de entretenimento e educativo possuem, respectivamente, 30 e 120 pedaços. O número de nós na rede sobreposta era igual a 200.

Nas primeiras rodadas de simulações, a banda de *upload* era igual em todos os nós, 600 kb/s. Isso foi feito para que nenhum outro parâmetro pudesse mascarar o desempenho do protocolo de difusão escolhido diante de nós não-cooperativos. O número de vizinhos de cada nó foi igual a 12, este número foi definido baseado nos resultados de (Cigno et al, 2008), onde a rede tornava-se estável com valores de vizinhos maiores ou iguais a 12. Variou-se o parâmetro porcentagem de nós não-cooperativos da rede em 0%, 5%, 10%, 15% e 20%.

Tabela 2: Parâmetros considerados em todas as simulações.

	Entretenimento	**Educativo**
Número de nós participantes	200	200
Tamanho do vídeo (pedaços)	30	120
Tamanho do vídeo (minutos)	5	20
Taxa de transmissão do vídeo (Kb/s)	350	350
Duração do pedaço de vídeo (s)	10	10
Tamanho do pedaço do vídeo (kB)	437,5	437,5

3.2.1.1 Métricas escolhidas

Quatro métricas foram escolhidas para verificar o desempenho do protocolo *push/pull* em presença de nós não-cooperativos durante a simulação.

A primeira métrica escolhida foi o tempo máximo que o último nó a terminar de reproduzir leva para obter todos os pedaços. Esta métrica é de grande relevância para distribuição de arquivos, onde o produto só pode ser utilizado quando o mesmo está inteiro no destino final. Esta métrica não é tão relevante para distribuição de vídeo, mas foi utilizada no

trabalho (Cigno et al, 2008), onde o protocolo foi apresentado, então foi decidido apresentá-la neste trabalho também.

As outras três métricas foram utilizadas para mensurar o desempenho do protocolo em aplicações de distribuição de vídeo, são elas: qualidade do sistema de vídeo, índice de continuidade e atraso inicial. A métrica qualidade do sistema de vídeo é definida em (Habib e Chuang, 2004) como:

$$Q = \frac{\sum_{i=1}^{T} Zi}{T} \tag{1}$$

onde T é o número total de pedaços envolvidos na distribuição de vídeo e Z_i é a variável que representa o fato do pedaço ter chegado ou não antes do seu tempo de reprodução. Caso o pedaço tenha chegado antes do seu tempo de reprodução, a variável Z_i recebe 1, do contrário Z_i recebe 0, o que representa descontinuidade no vídeo. Esta métrica captura implicitamente outras métricas, tais como: atraso do pedaço, perda do pedaço e variação de retardo.

O índice de continuidade definido em (Moraes, 2009) é uma métrica importante para sistema de distribuição de vídeo, pois ela representa o quanto o usuário obteve o vídeo de forma contínua, ou seja, sem interrupções. Quando um pedaço chega após o seu tempo de reprodução, significa que o usuário ficou provavelmente com a imagem do último quadro congelada até que este pedaço atrasado chegue ou que um temporizador estoure, ou seja, pedaços atrasados geram descontinuidade na reprodução do vídeo e consequentemente desconforto ao usuário.

O tempo total em que a reprodução do vídeo fica parada aguardando o pedaço atrasado chegar para reproduzi-lo é chamado de tempo de espera. De posse do tempo de espera do nó é possível calcular o índice de continuidade do mesmo. Tem-se que:

$$c = \frac{tr - te}{tr} \tag{2}$$

onde t_r é o tempo total de reprodução do vídeo. Quanto maior o índice de continuidade, mais tempo em que o usuário assistiu ao vídeo sem interrupções. Um índice igual a 100% significa que não houve interrupções na reprodução de vídeo de um dado nó.

A última métrica a ser avaliada é o atraso inicial na reprodução do vídeo. A estratégia definida para o momento certo de inicializar a reprodução de um vídeo tem importância fundamental no sucesso da qualidade do vídeo ao longo da reprodução. O artigo (Carlsson e Eager, 2007) apresenta algumas políticas de como determinar o melhor momento para iniciar o vídeo.

Algumas políticas são mais simples, como a política "ao menos (*b*)": Começa a reproduzir o vídeo quando *b* pedaços tiverem sido recebidos pelo nó destino, e uma dessas pedaços for o pedaço com o identificador *0*. Outras são mais complexas, como as políticas "LTA (*b*)" e "EWMA (*b*, α)" apresentadas no artigo (Carlsson e Eager, 2007), que em resumo agem da seguinte forma, cada uma a seu modo. As politicas definem um "*in-order buffer*" que contém todos os pedaços até o primeiro pedaço ausente e definem a taxa de ocupação deste *buffer* a medida que o mesmo vai aumentando. Essa taxa de ocupação inicialmente é menor do que a taxa de *download* dos pedaços, pois pedaços também chegam fora de ordem e portanto não ficam dentro deste *buffer*. Entretanto esta taxa de ocupação pode exceder a taxa de *download* a medida que os buracos no *buffer* vão sendo ocupados. Logo, é mais seguro inicializar a reprodução de vídeo uma vez que o valor corrente da taxa de ocupação permita que o "*in-order buffer*" preencha os buracos com o tempo que ele levaria para reproduzir o arquivo inteiro, se a taxa de reprodução se mantivesse ao longo do tempo. Com *k* pedaços no "*in-order buffer*" a taxa de ocupação deveria ser pelo menos (K – k) / K vezes maior que a taxa de reprodução do vídeo.

O simulador *P4S/PeerSim* não possui um escalonador para a reprodução de vídeo. Ele se preocupa com a transmissão dos pedaços, de como e em quanto tempo os pedaços chegarão até o seu destino final, e não de como este vídeo será reproduzido neste destino. Como não era escopo deste trabalho o desenvolvimento desta funcionalidade no simulador, mas ao mesmo tempo era importante pelo menos visualizar a tendência do atraso inicial e da perda de continuidade da rede P2P em face a adição de nós não-cooperativos, definiu-se uma política simples para calcular o atraso inicial e, consequentemente, todo o restante da reprodução. Esta política é aplicada aos arquivos de resultado das simulações.

Definiu-se da seguinte forma a política para inicializar o vídeo: aguardar a chegada dos três primeiros pedaços, ou seja, pedaços *0*, *1* e *2*. A Figura 13 explica com um exemplo numérico como foi feito o cálculo para determinar se cada pedaço chegou antes ou após o seu tempo de reprodução, dada a política citada.

Na Figura 13 existem duas tabelas, a primeira tabela possui duas colunas: (i) o tempo de chegada (TC) de um pedaço ao nó destino e (ii) a identificação do próprio pedaço (P). A segunda tabela possui quatro colunas: (i) tempo em que a reprodução do pedaço deve iniciar (TR), (ii) a identificação do pedaço, (iii) teste que verifica se o tempo de reprodução é maior ou igual ao tempo de chegada e (iv), por fim, a coluna resultado do teste. Supondo que o a duração da reprodução de cada pedaço seja de 2 segundos, que a ordem de chegada dos pedaços esteja representada na primeira tabela e que o atraso inicial seja igual ao tempo de chegada do último dos três primeiros pedaços, temos que:

$$TR_{inicial} = TC_1 = 4 \text{ s} \quad (3)$$

Para os três primeiros pedaços não há teste a ser feito, pois foi definido que a condição para o vídeo iniciar é que os três tenham chegado. Como TR inicial é igual a 4 segundos e o pedaço *0* levará 2 segundos para terminar de reproduzir, o pedaço *1* começará no instante 6 segundos e terminará no instante 8 segundos. O pedaço *2* começará em 8 segundos e terminará em 10 segundos. Já o quarto pedaço (p = 3) precisa começar a ser reproduzido no instante igual a 10 segundos, logo TR_3 é igual a 10 segundos, verifica-se o tempo de chegada do pedaço 3, TC_3 é igual a 2 segundos. TR_3 é maior ou igual a TC_3? Sim, então o pedaço chegou antes do seu tempo de reprodução. Esse cálculo é feito para todos os pedaços. Nota-se que o pedaço 8 chegou após o seu tempo de reprodução, fazendo com que haja descontinuidade de 1 segundo na reprodução do vídeo.

TC	P
1	0
2	2
2	3
3	4
4	1
6	5
9	7
10	6
14	10
17	9
21	8
22	12
25	11

TR	P	Teste do escalonador	Resultado
4	0	não precisa	
6	1	não precisa	
8	2	não precisa	
10	3	10 ≥ 2	antes
12	4	12 ≥ 3	antes
14	5	14 ≥ 6	antes
16	6	16 ≥ 10	antes
18	7	18 ≥ 9	antes
20	8	20 ≥ 21	após
21	8		
23	9	23 ≥ 17	antes
25	10	25 ≥ 14	antes
27	11	27 ≥ 25	antes
29	12	29 ≥ 22	antes

TR de cada pedaço = 2s

Figura 13: Exemplo numérico para o escalonador de reprodução definido.

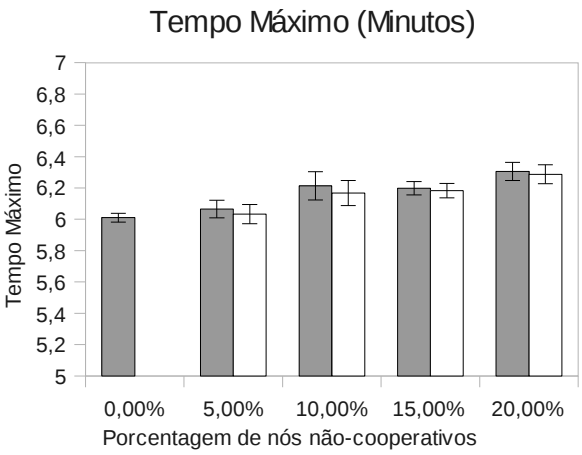
Conforme pode ser percebido, o atraso inicial interfere e muito no restante do desempenho da reprodução. Quanto maior o atraso inicial, maior a probabilidade de não haver descontinuidade no vídeo, entretanto um atraso muito grande pode gerar irritação ao usuário que solicitou o vídeo. Já um atraso inicial muito pequeno pode comprometer a reprodução do vídeo, fazendo com que o usuário tenha uma boa primeira impressão, mas uma péssima avaliação da reprodução. Enfim, existe uma relação de custo-benefício entre as duas métricas.

Um ponto interessante que poderia ser estudado é qual o tamanho ideal do pedaço para manter uma boa relação entre as métricas, pois pedaços pequenos podem acarretar em atraso inicial pequeno, mas desempenho pobre com várias pequenas interrupções. Já pedaços grandes podem acarretar em atraso inicial alto, mas um bom desempenho ao longo da reprodução.

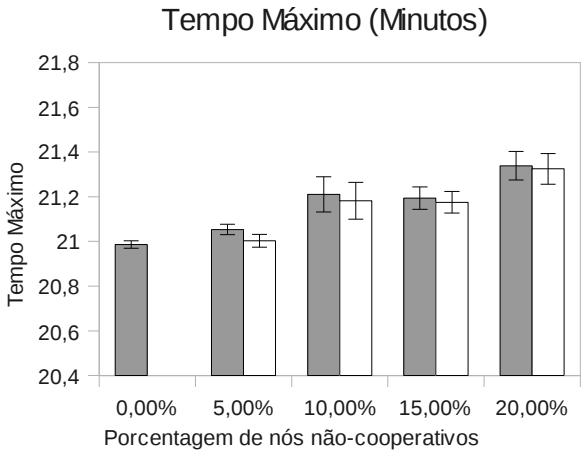
3.2.1.2 Resultados

As Figuras 14.a, 15.a, 16a e 17.a apresentam os gráficos de desempenho das métricas tempo máximo, índice de continuidade (ICO), qualidade do sistema de vídeo (Q) e atraso inicial respectivamente, para uma rede sobreposta que possui os seguintes parâmetros: banda de *upload* igual a 600 kb/s , 30 pedaços (C) para serem distribuídos e número de nós (N) igual a 200, com 0, 5, 10, 15 e 20% de nós não cooperativos. Já as Figuras 14.b, 15.b, 16.b e 17.b apresentam os gráficos de desempenho da mesma rede, mas aumentando o número de pedaços para 120.

Para cada configuração de parâmetros (N, C, banda e porcentagem de nós não-cooperativos) foram realizadas 15 rodadas de simulação, totalizando 150 simulações. O eixo *y* dos gráficos das figuras 14, 15, 16 representam as médias das métricas avaliadas obtidas pela rede com um intervalo de confiança de 90%, já o eixo *x* representa a porcentagem de nós não-cooperativos na rede sobreposta. Cada gráfico possui dois grupos de barra. O grupo de barras cinzas corresponde ao desempenho dos nós cooperativos, já o grupo de barras brancas corresponde ao desempenho dos nós não-cooperativos.



(a) Tempo máximo para rede com N=200, C=30 e *upload* = 600 kb/s.



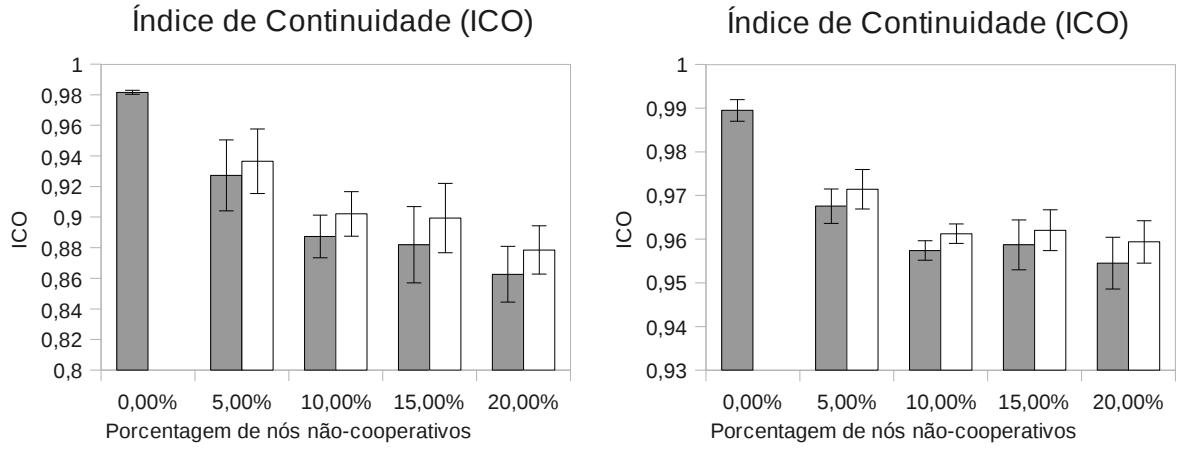
(b) Tempo máximo para rede com N=200, C=120 e *upload* = 600 kb/s.

Figura 14: Gráficos de desempenho das redes com relação à métrica tempo máximo (minutos).

Conforme pode ser observado nas Figuras 14.a e 14.b, o tempo máximo dos nós cooperativos e não-cooperativos são muito próximos, com uma vantagem muito pequena para os nós não-cooperativos. É possível notar em ambas que o melhor desempenho dos nós da rede ocorreu quando a quantidade de nós não-cooperativos foi igual a zero, e que o desempenho foi piorando a medida que o número de nós não-cooperativos na rede foi aumentando. Esse resultado era esperado conforme foi visto na seção 2.3 e foi confirmado.

Para a Figura 14.a o melhor valor ficou próximo a 6 minutos e o pior valor próximo a 6,3 minutos, ou seja, no melhor caso, todos os nós terminaram em até 6 minutos e no pior caso em até 6 minutos e 18 segundos, lembrando que o vídeo era de 5 minutos e que há um atraso inicial para a reprodução do vídeo. Já na Figura 14.b, no melhor caso todos terminaram em até 21 minutos e no pior caso em até 21 minutos e 21 segundos, para este caso, o vídeo era de 20 minutos.

As Figuras 15.a e 15.b apresentam o desempenho da rede com relação a métrica ICO, uma das mais importantes métricas em se tratando de distribuição de vídeo. Como pode ser observado, para ambos os casos, houve degradação do ICO à medida que a porcentagem de nós não-cooperativos foi aumentando. Assim como na métrica anterior, o melhor desempenho também foi para a rede sem nós não-cooperativos.



(a) Índice de continuidade da rede com N=200, C=30 e *upload* = 600 kb/s.

(b) Índice de continuidade da rede com N=200, C=120 e *upload* = 600 kb/s.

Figura 15: Gráficos de desempenho das redes com relação à métrica índice de continuidade (ICO).

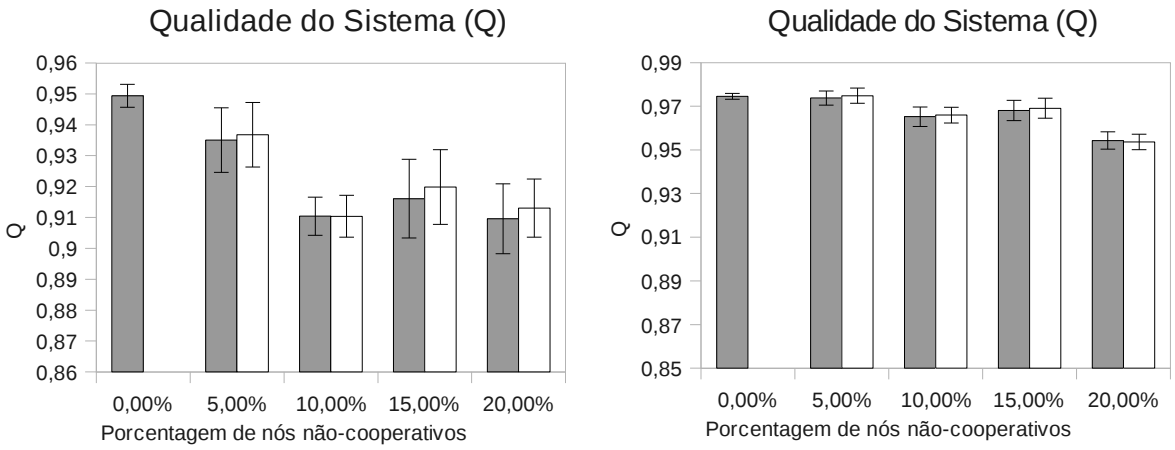
No primeiro caso, Figura 15.a, a rede atingiu um ICO igual a 0,98 para o melhor caso, isso significa de que em 5 minutos de vídeo reproduzido, houve 6 segundos de interrupção em média. Já para o pior caso que corresponde a rede com 20% de nós não-cooperativos, a rede obteve um ICO igual a 0,86, ou seja, uma média de 42 segundos de interrupção. Para esta situação, observa-se uma diferença de 12 pontos percentuais.

No segundo caso, Figura 15.b, o ICO para o melhor caso foi igual a 0,99, ou seja, em 20 minutos de reprodução de vídeo, houve em média 12 segundos de interrupção. Já para o pior caso, que também corresponde a rede com 20% de nós não-cooperativos, a rede obteve um ICO igual a 0,95, ou seja, média de 60 segundos de interrupção. Para esta situação verifica-se uma diferença de 4 pontos percentuais.

Para os dois casos, os nós não-cooperativos obtiveram um melhor desempenho que os nós ditos normais. As curvas dos nós não-cooperativos e normais possuem tendências muito parecidas, mostrando que o bom desempenho de um leva ao bom desempenho do outro, e vice-versa. Isto nem sempre é o desejado, principalmente em se tratando de comportamentos tão distintos na rede, o sistema deveria beneficiar aos nós cooperativos e punir os nós não-cooperativos.

O melhor desempenho da métrica ICO para os nós não-cooperativos era esperado, pois como os mesmos não perdem tempo com o estado *push*, onde eles deveriam entregar pedaços de maneira voluntária aos demais nós, ao trocar de imediato para o estado *pull*, o nó não-cooperativo tende a preencher de maneira mais rápida os “buracos” do seu *buffer* de reprodução, desde que, obviamente, possua nós cooperativos como seus vizinhos. É provável que para números bem próximo a 0% de maliciosos, a média obtida pelos nós não-cooperativos seja até mesmo melhor do que a média obtida pelo nós de uma rede com apenas nós cooperativos, fazendo valer à pena se utilizar de um mau comportamento.

As Figuras 16.a e 16.b apresentam o desempenho das redes com relação a métrica qualidade (Q). Percebe-se que para ambos os casos, as curvas dos nós não-cooperativos e cooperativos ficam praticamente sobrepostas. Além disso, ocorre uma degradação da qualidade a medida que a porcentagem de nós não-cooperativos aumenta, conforme ocorreu com as métricas anteriores. Novamente o melhor desempenho foi para a rede sem nós não-cooperativos para ambos os casos.



(a) Qualidade do sistema de vídeo da rede com N=200, C=30 e *upload* = 600 kb/s.

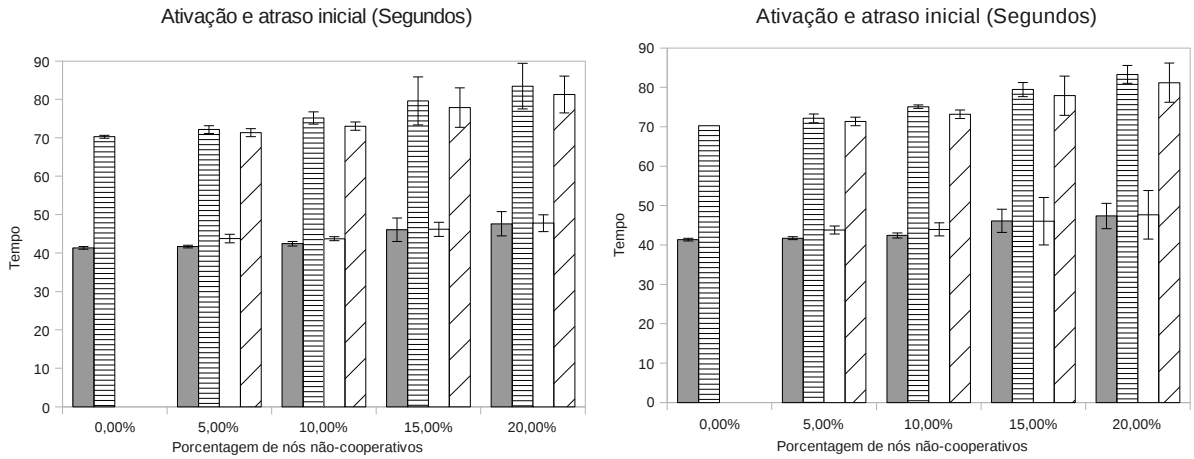
(b) Qualidade do sistema de vídeo da rede com N=200, C=120 e *upload* = 600 kb/s.

Figura 16: Gráficos de desempenho das redes com relação à métrica qualidade do sistema de vídeo.

Na Figura 16.a, o melhor Q foi igual a 0,95, o que significa que de 30 pedaços, em média 1,5 pedaços chegaram com atraso em cada nó. Já no pior caso, que ocorreu na rede com 20% de nós não-cooperativos, o Q foi igual a 0,91, ou seja, de 30 pedaços em média 2,7 chegaram atrasados. Ocorreu uma perda de 4 pontos percentuais.

Na Figura 16.b, o melhor Q foi igual a 0,97, ou seja, de 120 pedaços apenas 3,6 pedaços chegaram atrasados em cada nó. Para o pior caso, novamente quando a rede possuía 20% de nós não-cooperativos, Q foi igual a 0,95, isto é, de 120 pedaços em média 6 pedaços chegaram atrasados em cada nó. Para este caso, houve uma perda de 2 pontos percentuais.

Um ponto a ser comentado é que se as métricas ICO e Q forem analisadas em conjunto, poder-se-ia dizer que na melhor situação do primeiro caso, tem-se que em média 1,5 pedaços atrasados foram responsáveis por 6 segundos de interrupção em cada nó. Já no segundo caso, para a melhor situação, tem-se que em média 3,6 pedaços atrasados foram responsáveis por 12 segundos de interrupção em cada nó.



(a) Atraso inicial da rede com N=200, C=30 e *upload* = 600 kb/s.

(b) Atraso inicial da rede com N=200, C=120 e *upload* = 600 kb/s.

Figura 17: Gráficos de desempenho das redes com relação à métrica atraso inicial.

As Figuras 17.a e 17.b apresentam o desempenho das redes com relação à métrica atraso inicial. Para analisar esta métrica, decidimos inserir também o comportamento da métrica tempo de ativação dos nós, pois conforme foi explicado na seção 3.2.1 o nó só começa a participar da rede de forma efetiva no momento que o mesmo recebe o primeiro pedaço, a partir daí o nó pode solicitar e enviar pedaços. Por isso, acreditávamos que o tempo de ativação influenciava a métrica atraso inicial, só não sabíamos o quanto.

Sendo assim, cada gráfico da Figura 17 possui quatro grupos de barras, o grupo com barras cinzas corresponde ao desempenho dos nós ditos normais com relação à métrica ativação, o grupo com barras brancas corresponde ao desempenho dos nós não-cooperativos com relação a mesma métrica, já os grupos com barras hachuradas horizontalmente e com barras hachuradas com ângulo de 45 graus correspondem ao desempenho dos nós normais e não-cooperativos respectivamente, com relação à métrica atraso inicial.

O primeiro ponto que chama a atenção nos gráficos é que os grupos das Figuras 17.a e 17.b possuem valores praticamente iguais, independente da quantidade de pedaços a serem distribuídos pela rede. Isto ocorre devido ao fato de termos utilizados as mesmas sementes nas rodadas de simulação, que faz com que o comportamento inicial dificilmente seja alterado, apenas porque o número de pedaços a ser distribuído foi alterado.

Percebe-se que o desempenho das redes com relação a esta métrica vai se degradando suavemente à medida que a porcentagem de nós não-cooperativos aumenta. As curvas dos nós cooperativos e não-cooperativo são muito similares. O melhor desempenho das redes novamente foi visto quando a quantidade de nós não-cooperativos foi igual a zero.

Para as duas Figuras 17.a e 17.b, o melhor valor para tempo de ativação ficou próximo a 40 segundos e o pior caso, com 20% de nós não-cooperativos, próximo a 50 segundos, ou seja, uma diferença de 10 segundos. O melhor valor para o atraso inicial ficou próximo a 70 segundos e o pior valor ficou próximo a 82 segundos, uma diferença de 12 segundos.

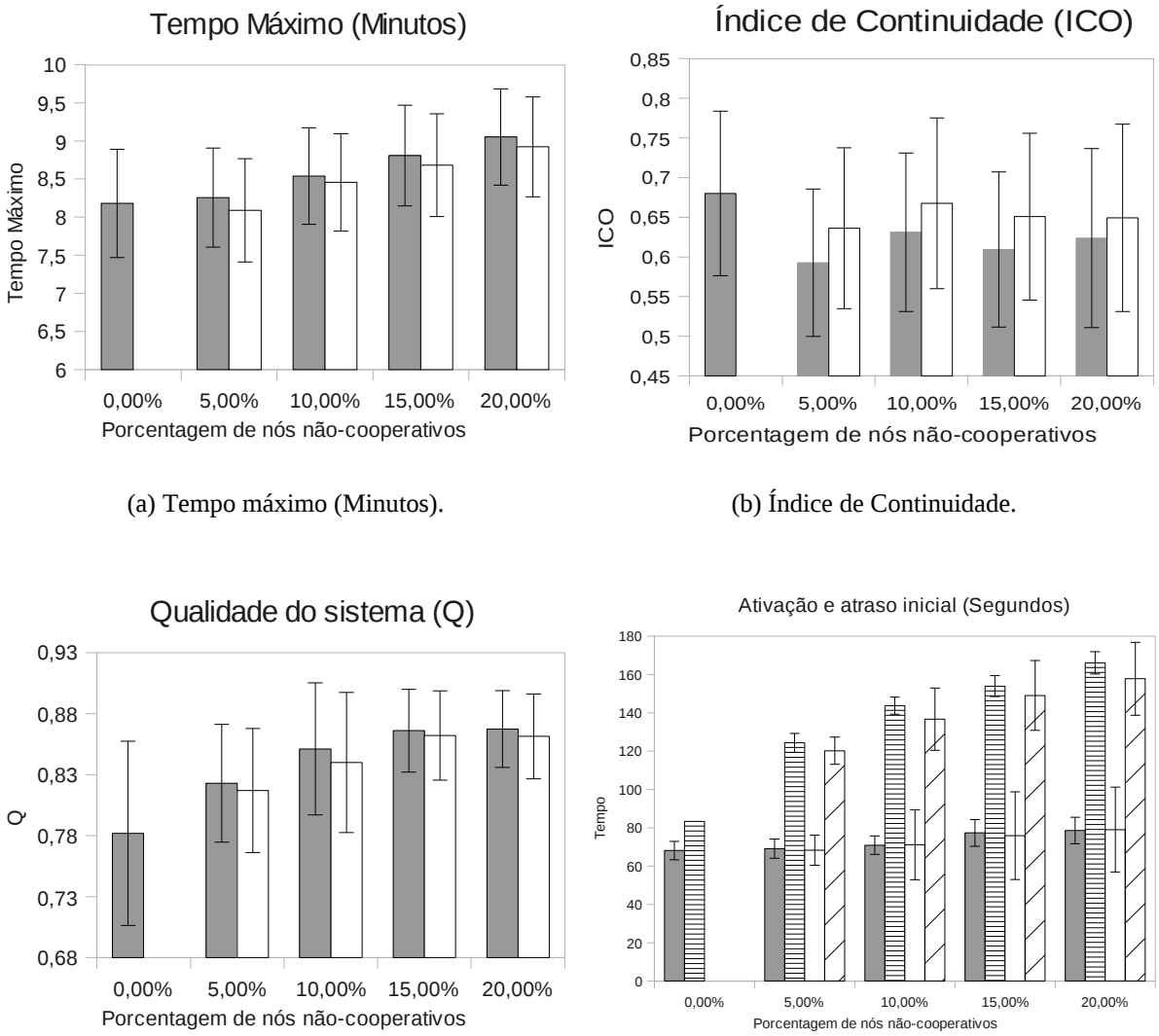
Pode ser notado que o tempo de ativação de um nó é responsável por mais da metade do atraso inicial do mesmo, o que é muito representativo. Uma boa contribuição futura seria realizar um estudo para tentar modificar o mecanismo de ativação do nó para otimizar o atraso inicial.

O último ponto a ser comentado sobre métrica atraso inicial é que como a mesma se comportou de maneira praticamente igual para os dois casos, 30 e 120 pedaços, pode se dizer que o vídeo menor ficou prejudicado. O usuário que assistiu ao vídeo de 5 minutos, no melhor caso, ficou aguardando 70 segundos para começar a reproduzir um vídeo de 300 segundos de duração, ou seja, a reprodução do vídeo representou 81% do total de 370 segundos gastos. Para o vídeo de 120 pedaços, o usuário aguardou, para o melhor caso, 70 segundos para começar a assistir o vídeo e assistiu 20 minutos de vídeo (1200 segundos), ou seja, a reprodução do vídeo representou 94,5% do tempo total. Para o cálculo dos dois casos, desconsideramos o tempo de descontinuidade.

Após analisar o desempenho do protocolo de difusão em redes P2P na presença de nós não-cooperativos, com os nós possuindo banda de *upload* fixa de 600 kb/s, resolveu-se verificar o desempenho do protocolo de difusão nas mesmas condições anteriores, exceto que a banda deixou de ser fixa e passou a ser variável com os seguintes possíveis valores: 200, 360, 600 e 1000 kb/s. Cada nó pode adquirir de forma aleatória um desses 4 valores para banda de *upload*.

A Figura 18 apresenta o gráfico de desempenho das métricas tempo máximo, índice de continuidade (ICO), qualidade do sistema de vídeo (*Q*) e atraso inicial da rede sobreposta que possui os seguintes parâmetros: número de nós *(N*) igual a 200, banda de *upload* variável e 30

pedaços (*C*) para serem distribuídos. Já a Figura 19 apresenta o gráfico de desempenho da rede composta por 200 nós, banda de *upload* variável e 120 pedaços.



(a) Tempo máximo (Minutos).

(b) Índice de Continuidade.

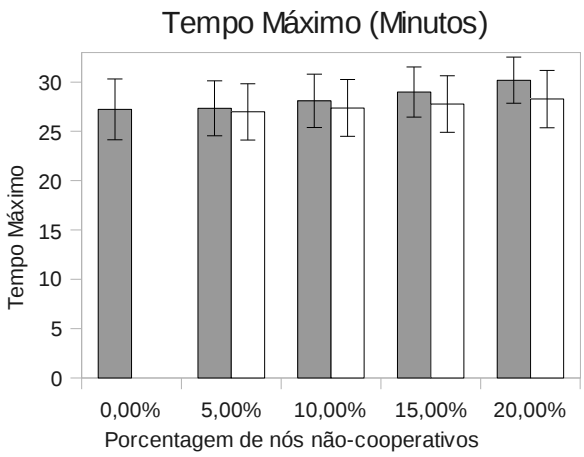
(c) Qualidade do sistema de vídeo.

(d) Atraso Inicial.

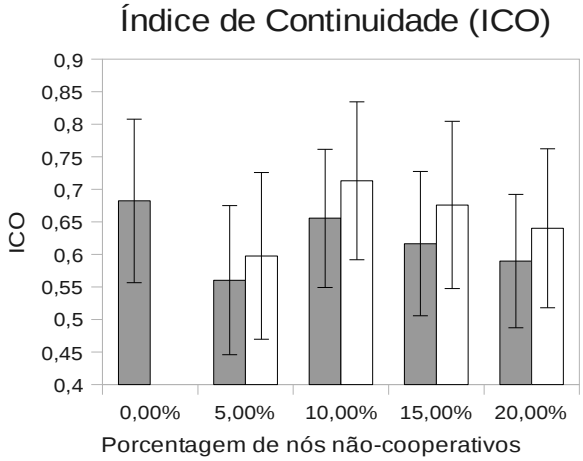
Figura 18: Gráficos de desempenho da rede sobreposta com os parâmetros N=200, C=30 e banda de *upload* variável.

Para cada conjunto de parâmetros (*N*, *C*, banda e porcentagem de nós não-cooperativos) foram realizadas 15 rodadas de simulação, totalizando 150 rodadas. O eixo *y* dos gráficos das Figuras 18 e 19 representam as médias das métricas obtidas pela rede com um intervalo de confiança de 90%, já o eixo *x* representa a porcentagem de nós não-cooperativos na rede sobreposta. Novamente, cada gráfico possui dois grupos de blocos,

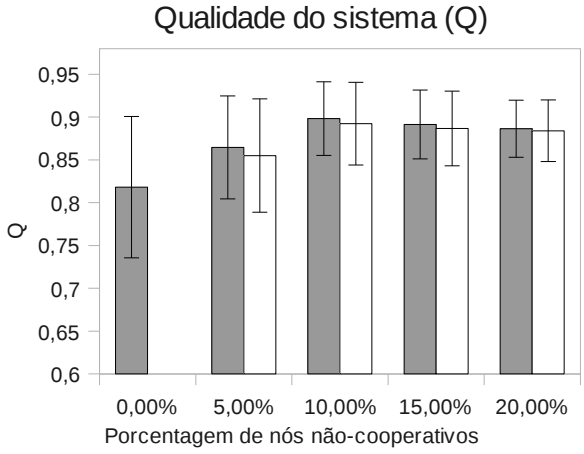
exceto o gráfico do atraso inicial (18d e 19d) que possui quatro grupos, o grupo de barras cinzas corresponde ao desempenho dos nós cooperativos, já o grupo de barras brancas corresponde ao desempenho dos nós não-cooperativos.



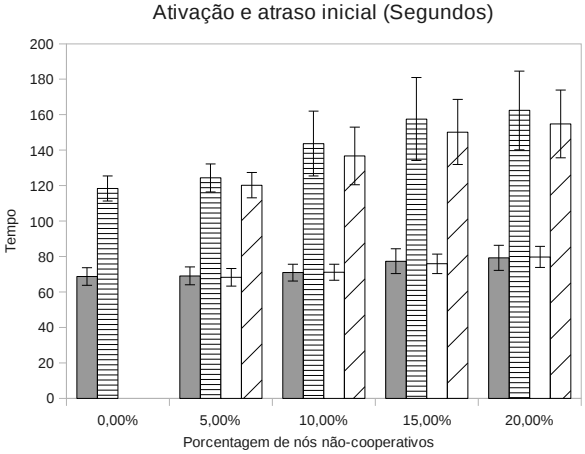
(a) Tempo máximo (Minutos).



(b) Índice de Continuidade.



(c) Qualidade do sistema de vídeo.



(d) Atraso Inicial.

Figura 19: Gráficos de desempenho da rede sobreposta com os parâmetros N=200, C=120 e banda de *upload* variável.

A tendência das curvas são similares às curvas apresentadas para banda de *upload* fixa: melhor desempenho para as redes sem nós não cooperativos e desempenho degradado à medida que se aumenta a porcentagem de nós não-cooperativos. Entretanto dois pontos chamam atenção. O primeiro ponto é o fato do desempenho das redes terem caído muito, e o segundo ponto é que o desvio padrão é alto.

Fazendo algumas comparações com os números anteriores, verifica-se que o tempo máximo para o melhor caso da rede que distribuía 30 pedaços foi de 8 minutos e 11 segundos contra 6 minutos com banda fixa em 600 kb/s, o pior caso obteve 9 minutos e 3 segundos contra 6 minutos e 18 segundos. Houve um aumento de 2 a 3 minutos em média. Para as redes que distribuíram 120 pedaços, o melhor caso obteve tempo máximo de 27 minutos contra 21 minutos, e o pior caso 30 minutos e 19 segundos, contra 21 minutos e 21 segundos, ou seja, uma degradação muito alta.

Com uma análise breve nos gráficos, percebe-se que as outras métricas também ficaram muito prejudicadas. Na verdade esta alta variação e desempenho ruim do modelo com relação a banda já eram esperados, e não é culpa do protocolo de difusão e sim da forma que é criado o grafo e principalmente de como ele é mantido durante a simulação, de fato o grafo é mantido fixo no simulador durante todo o tempo da simulação.

Sendo assim, dependendo de como o grafo for inicialmente criado, o desempenho pode variar muito. Se os nós ficarem bem balanceados em termos de banda e obtiverem aleatoriamente melhores bandas, melhor será o desempenho, caso contrário, pior será o desempenho. Importante salientar que das quatro bandas possíveis, uma (600 kb/s) era igual ao caso estudado anteriormente, uma (1000 kb/s) está acima e duas (200 e 360 kb/s) abaixo. Isto levou a um desempenho ruim, pois se todas as bandas fossem acima de 600 kb/s, o desempenho provavelmente seria bom, na média, mas com uma variação alta. Entretanto, a realidade brasileira ainda não é esta: todos os usuários possuindo banda de *upload* acima de 600 kb/s.

Vale a pena comentar outros dois pontos observados durante as simulações anteriores com banda fixa e com banda variável. Com valores mais altos para porcentagem de nós não-cooperativos, como 50% por exemplo, a simulação fica impraticável, pois vários nós sequer são ativados, além do desempenho das redes ficar bastante degradado.

Além disso, em poucos casos para simulações com 15% ou 20% de nós não-cooperativos, ocorreu de nós cooperativos não ativarem, o que é um ponto bastante negativo. Faz-se necessário então, criar uma forma de impedir que este problema continue a ocorrer com os nós cooperativos. Neste sentido, com o objetivo de avaliar se um mecanismo de admissão de vizinhos no momento de formação do grafo seria benéfico, fizemos as simulações discutidas na seção a seguir.

3.2.2 Avaliação de mecanismo de admissão de vizinhos

Nesta segunda parte das simulações propõem-se um mecanismo de controle de admissão de vizinhos na formação do grafo, onde nós cooperativos possam selecionar vizinhos cooperativos e rejeitar vizinhos não-cooperativos. A idéia básica é criar um mecanismo que possibilite o nó fazer sua escolha baseada em uma reputação prévia do nó candidato a vizinho, neste caso, ou um nó cooperativo (*0*) ou um nó não-cooperativo (*1*).

De acordo com (Moura, 2009), os mecanismos de reputação agregam opiniões sobre o comportamento passado das entidades com o objetivo de estimar o seu comportamento futuro. Essa estimativa é baseada em valores que exprimem o grau de confiança nas entidades.

Baseado nesta premissa, utilizamos o atributo criado anteriormente que identificava se o nó era um nó cooperativo ou não-cooperativo como a reputação de cada nó, deixando de fazer uma seleção de vizinhos 100% aleatória.

Para cada nó cooperativo foi criado um contador para contabilizar a quantidade de vizinhos não-cooperativos correntes no momento da seleção dos mesmos, durante a formação do grafo, quando este contador atinge um determinado limiar, o nó cooperativo passa a rejeitar nós não-cooperativos como vizinhos.

Para que o mecanismo pudesse ser criado, modificou-se o método *WirekOutUnd* da classe *peersim.graph.GraphFactory* para que ao invés dos vizinhos de um determinado nó fossem escolhidos de forma 100% aleatória, passassem a levar em consideração se o nó candidato a ser vizinho era ou não cooperativo. A Figura 20 apresenta o pseudo-código para formação do grafo de maneira simplificada.

```
Algoritmo – Formação de grafo simétrico
Pseudo-código do algoritmo para formação do grafo simétrico levando em consideração
se o nó é ou não não-cooperativo

n → Número de nós na rede
k → Número de vizinhos

1ª etapa:
Criar vetor de nodes de tamanho n, com valores de 0 a n-1 de forma ordenada. Depois
misturar o conteúdo.
nodes[]
```

```
2ª etapa:
Escolher os vizinhos de cada nó.

para todo i variando de 0 a n faça
       enquanto j < k faça
              se (Node(i) isMalicius) então
                     Escolher uma posição aleatória do vetor nodes[] para vizinho (v)
                     se (Node(v) isMalicius) então
                            Node(i) vizinho Node(v)
                            j++
                     senão
                            se (%_atual_vizinho_mal(v) < %_max_vizinho_mal) então
                                   Node(i) vizinho Node(v)
                                   j++
                            senão
                                   j++
                            fim
                     fim
              senão
                     Escolher uma posição aleatória do vetor nodes[] para vizinho (v)
                     se (Node(v) isMalicius) então
                            se (%_atual_vizinho_mal(i) < %_max_vizinho_mal) então
                                   Node(i) vizinho Node(v)
                                   j++
                            senão
                                   j++
                            fim
                     senão
                            Node(i) vizinho Node(v)
                            j++
                     fim
              fim
       fim
fim
```

Figura 20: Pseudo-código do algoritmo de criação do grafo.

A condição *(%_atual_vizinho_mal(i) < %_max_vizinho_mal)* imposta no algoritmo para formação do grafo, permite que cada nó cooperativo informe a porcentagem de nós não cooperativos que está disposto a tolerar. Desta forma a seleção dos vizinhos deixa de ser 100% aleatória, pois beneficia os nós cooperativos a medida que pode calibrar a quantidade de vizinhos não-cooperativos na sua lista.

Um outro ponto é importante ser destacado, como a seleção deixou de ser 100% aleatória, o problema de um nó cooperativo não conseguir nem sequer ser ativado porque sua lista de nós vizinhos só possui nós não-cooperativos deixa de existir, à medida que a variável

%_max_vizinho_mal tenha valor baixo. Se a variável *%_max_vizinho_mal* tiver valor alto, próximo a 100%, provavelmente o comportamento do algoritmo modificado será próximo ao comportamento do algoritmo original, com probabilidade de um nó cooperativo nem sequer ser ativado na presença de nós não-cooperativos. Estes dois pontos comentados precisam ser confirmados nos resultados das simulações.

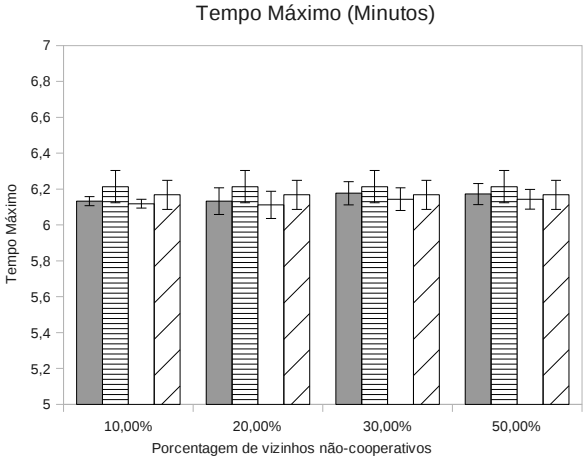
Da mesma forma que a primeira parte da simulação, uma vez formada a lista de vizinhos de cada nó, a mesma não é mais alterada durante toda a simulação. Novamente simulações são realizadas com os mesmos parâmetros apresentados na tabela 2. As métricas tempo máximo, qualidade do sistema de vídeo, índice de continuidade e atraso inicial são avaliadas e comparadas com os valores obtidos sem a alteração do método *WirekOutUnd* que forma o grafo simétrico.

3.2.2.1 Resultados

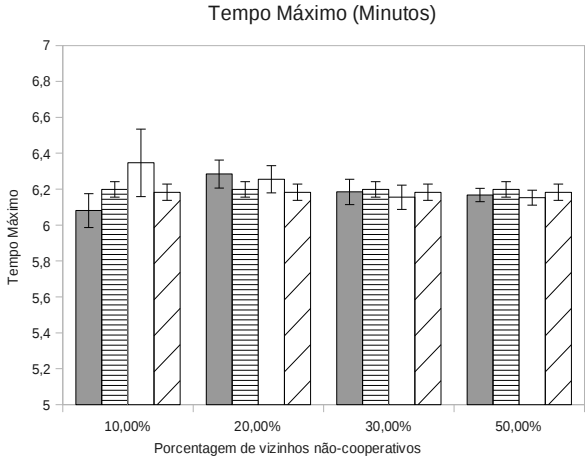
As Figuras 21 e 22 apresentam os gráficos de desempenho das redes que distribuem 30 e 120 pedaços respectivamente, com relação à métrica tempo máximo. Como já verificamos que o modelo ainda não está preparado para bandas de *upload* variável, decidimos fixar a banda novamente em 600 kb/s para que não haja influência de nenhuma outra variável durante a simulação.

Para cada conjunto de parâmetros (N, C, banda e porcentagem de nós não-cooperativos e porcentagem de vizinhos não-cooperativos) foram realizadas 15 rodadas de simulação, totalizando 480 rodadas. O eixo y dos gráficos das Figuras de 21 a 28 representam as médias das métricas obtidas pela rede com um intervalo de confiança de 90%, já o eixo *x* representa a porcentagem máxima de vizinhos não-cooperativos permitidos na lista de cada nó cooperativo.

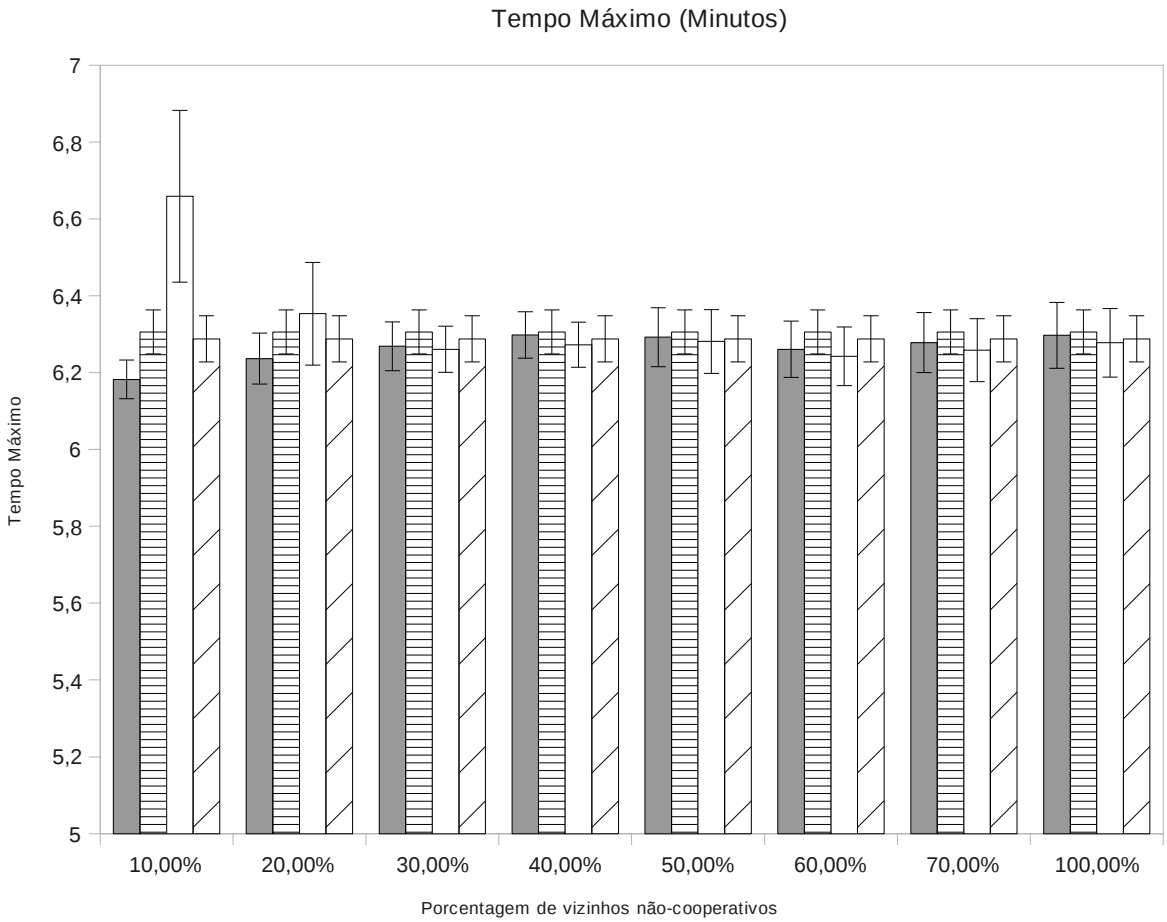
Cada gráfico possui quatro grupos de blocos, o grupo de blocos cinzas corresponde ao desempenho dos nós cooperativos. já o grupo de blocos brancos corresponde ao desempenho dos nós não-cooperativos ambos com o mecanismo, os grupos com barras hachuradas horizontalmente e com barras hachuradas com ângulo de 45 graus correspondem ao desempenho dos nós normais e não-cooperativos respectivamente, sem o mecanismo.



(a) Tempo máximo para uma rede com N=200, C= 30, upload = 600 kb/s e 10% dos nós não-cooperativos.

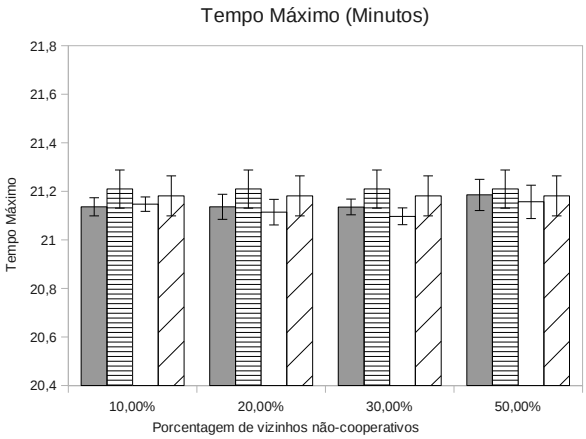


(b) Tempo máximo para uma rede com N=200, C= 30, upload = 600 kb/s e 15% dos nós não-cooperativos.

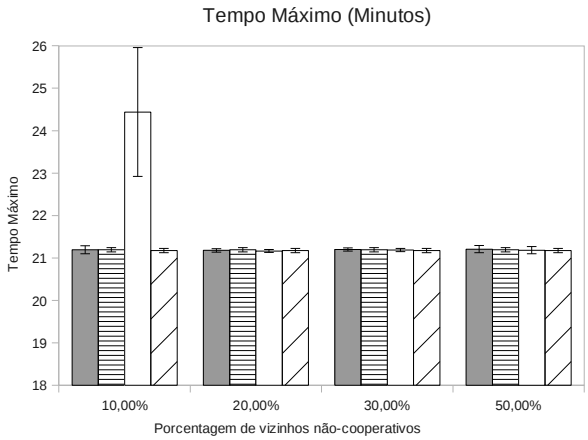


(c) Tempo máximo para uma rede com N=200, C= 30, upload = 600 kb/s e 20% dos nós não-cooperativos.

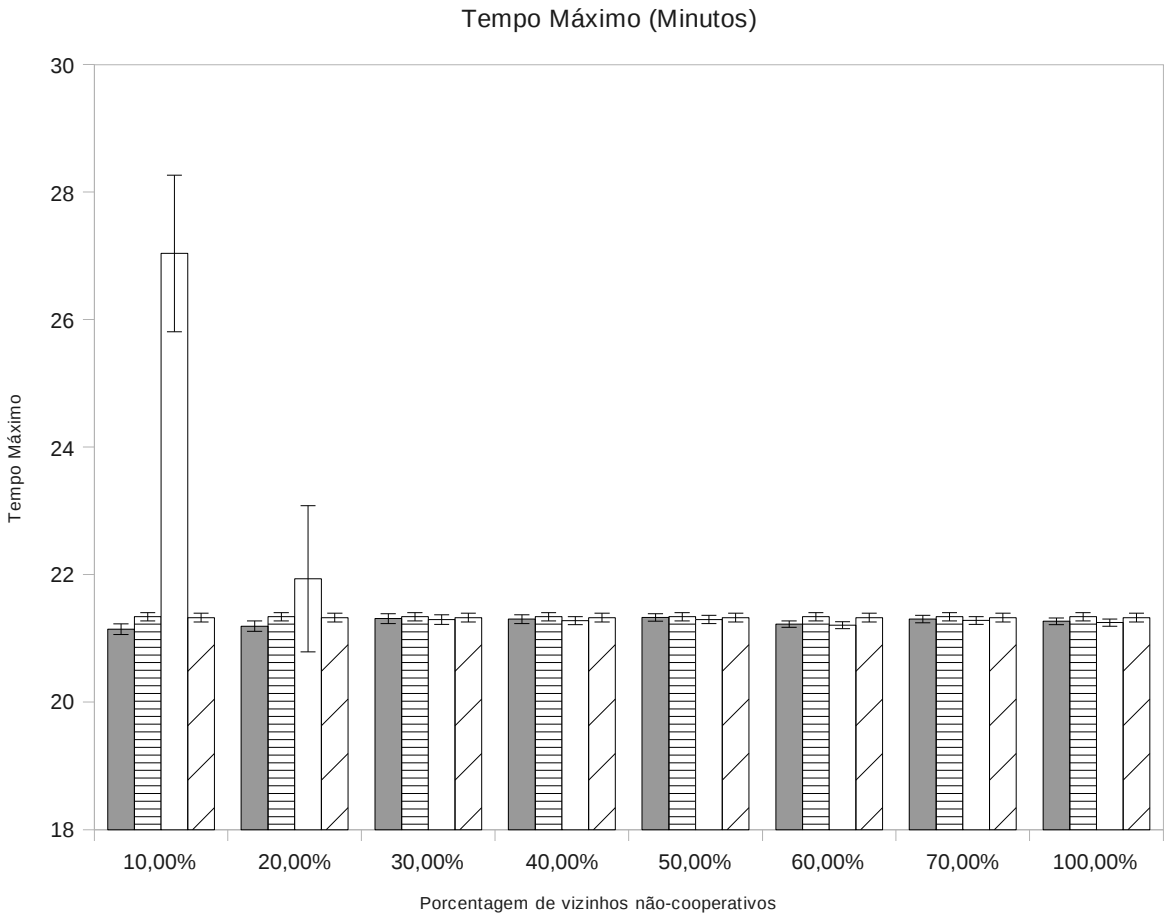
Figura 21: Tempo máximo para redes sobrepostas com os parâmetros N=200, C=30, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.



(a) Tempo máximo para uma rede com N=200, C= 120, upload = 600 kb/s e 10% dos nós não-cooperativos.



(b) Tempo máximo para uma rede com N=200, C= 120, upload = 600 kb/s e 15% dos nós não-cooperativos.



(c) Tempo máximo para uma rede com N=200, C= 120, upload = 600 kb/s e 20% dos nós não-cooperativos.

Figura 22: Tempo máximo para redes sobrepostas com os parâmetros N=200, C=120, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.

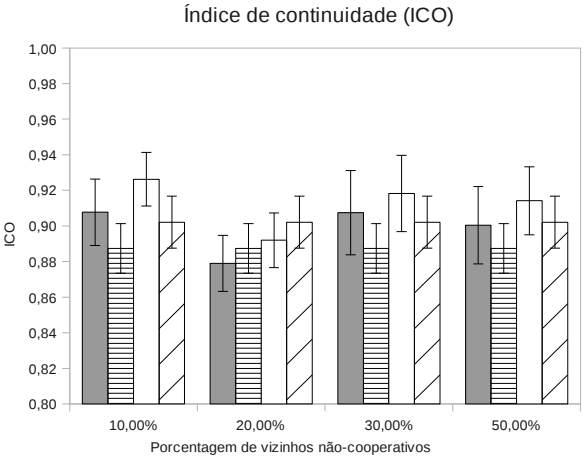
Conforme pode ser observado nas Figuras 21 e 22, de uma maneira geral houve uma pequena melhora de desempenho na rede como um todo, salvo para os nós não-cooperativos quando a porcentagem máxima de vizinhos não cooperativos permitida era baixa, menor que 20%.

Percebe-se que para esses valores baixos de porcentagem de vizinhos com porcentagem alta de nós não-cooperativos na rede (15% e 20%), o tempo máximo para nós maliciosos aumentou em comparação com as simulações anteriores sob mesmas condições. Por exemplo, para C=120 e porcentagem de nós não-cooperativos igual a 20%, Figura 22 c, o tempo máximo ficou próximo a 27 minutos, podendo variar 1 minuto para cima ou para baixo, ao passo que nas simulações sem mecanismo o valor era próximo a 21 minutos.

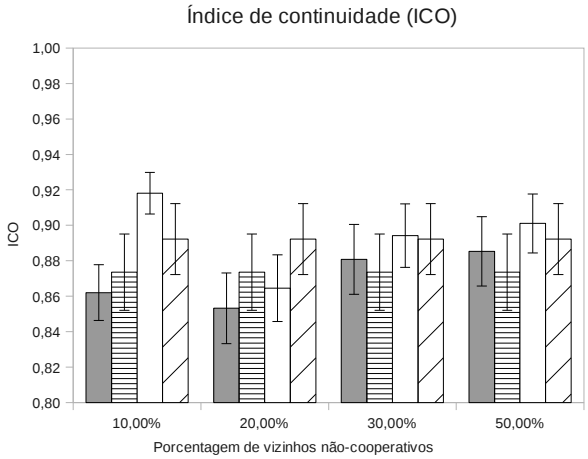
Esse comportamento já era esperado, pois em um universo de 200 nós com 20% de nós não-cooperativos, sobram apenas 160 nós cooperativos para serem distribuídos. Como apenas 10% de 12 vizinhos podiam ser nós não-cooperativos, ou seja, apenas 1 vizinho, a probabilidade de alguns nós não-cooperativos terem poucos ou nenhum nó cooperativo era alta. Com isso, o desempenho médio dos nós não-cooperativos que fossem ativados, seria prejudicado por aqueles com poucos nós cooperativos como vizinhos.

Outro ponto que pode ser notado, principalmente pelos gráficos com 20% dos nós sendo não-cooperativos, é que o modelo com mecanismo tende ao mesmo desempenho que o modelo sem mecanismo de admissão à medida que a porcentagem de vizinhos não-cooperativos tende a 100%.

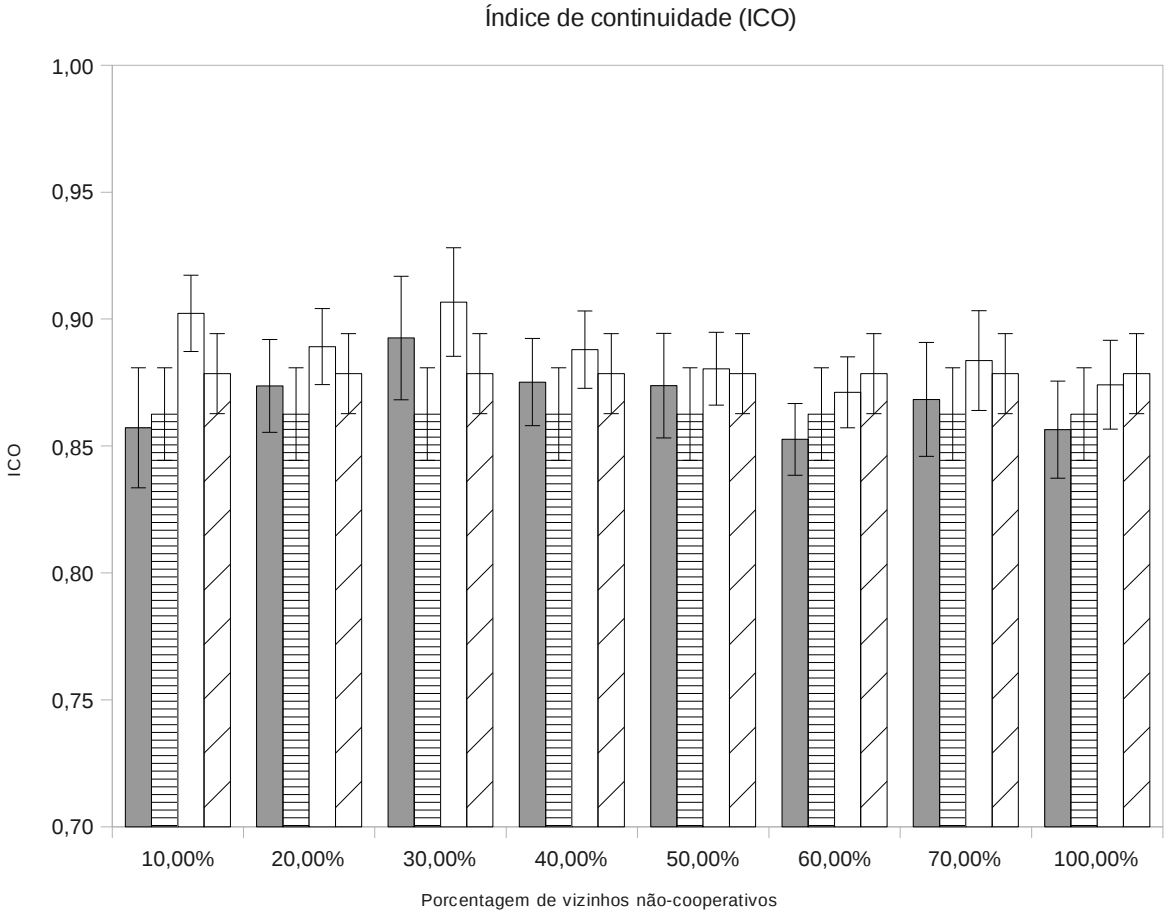
As Figuras 23 e 24 apresentam o desempenho das redes que distribuem 30 e 120 pedaços, respectivamente. Como pode ser visto, em todos os gráficos as curvas dos nós não-cooperativos foram melhores que as curvas dos nós cooperativos. Os nós das redes que distribuíram apenas 30 pedaços tiveram pior desempenho no ICO do que os nós das redes que distribuíram 120 pedaços, este fato também ocorreu nos resultados de ICO da seção 3.2.1.2.



(a) ICO para uma rede com N=200, C= 30, upload = 600 kb/s e 10% dos nós não-cooperativos

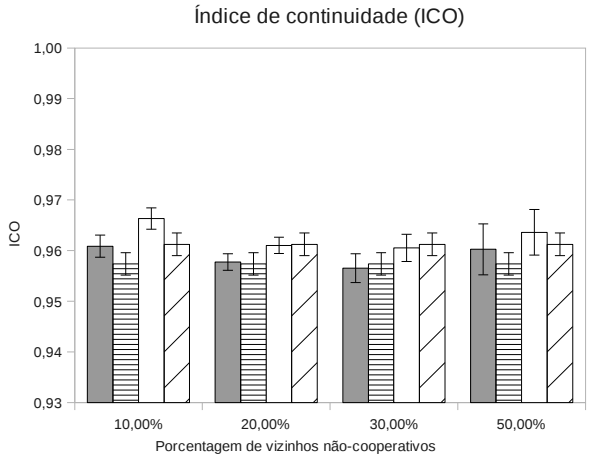


(b) ICO para uma rede com N=200, C= 30, upload = 600 kb/s e 15% dos nós não-cooperativos

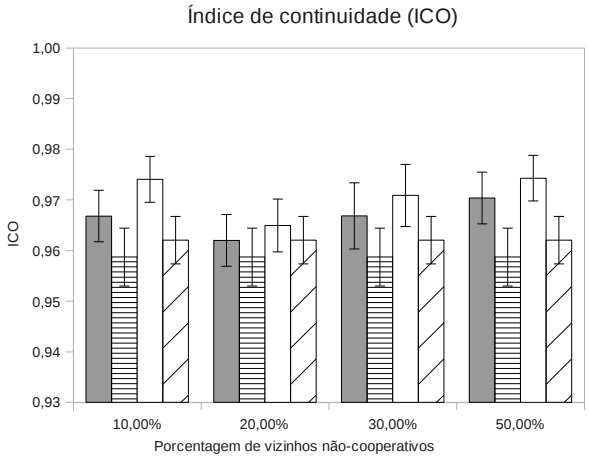


c) ICO para uma rede com N=200, C= 30, upload = 600 kb/s e 20% dos nós não-cooperativos

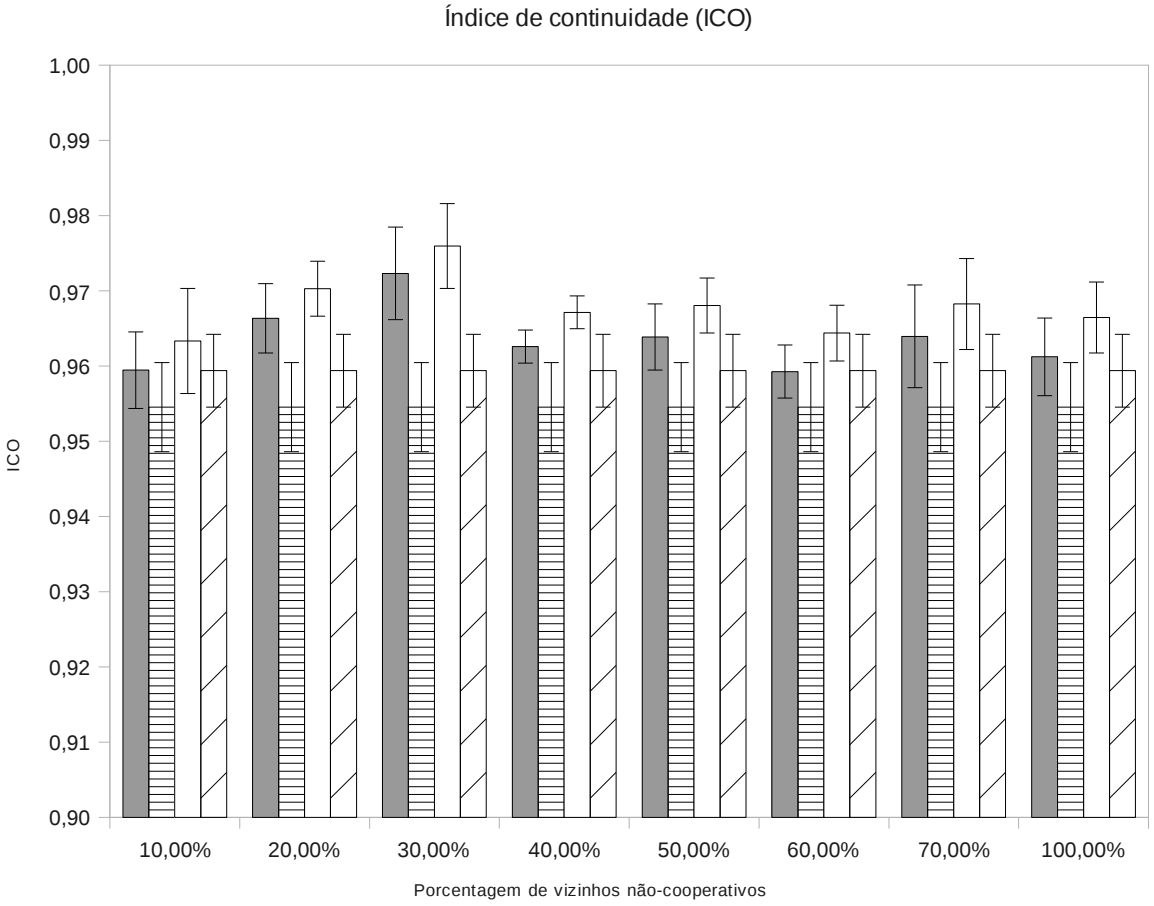
Figura 23: Índice de continuidade para redes sobrepostas com os parâmetros N=200, C=30, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%



(a) ICO para uma rede com N=200, C= 120, upload = 600 kb/s e 10% dos nós não-cooperativos



(b) ICO para uma rede com N=200, C= 120, upload = 600 kb/s e 15% dos nós não-cooperativos

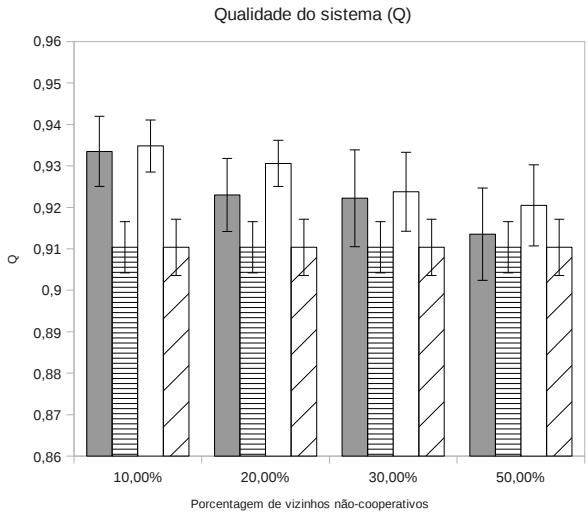


(c) ICO para uma rede com N=200, C= 120, upload = 600 kb/s e 20% dos nós não-cooperativos

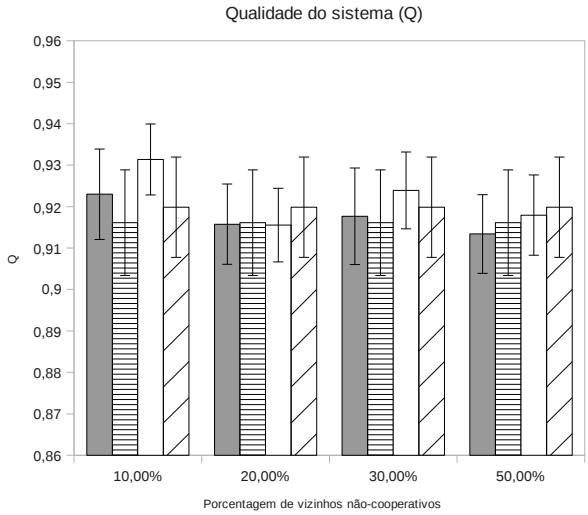
Figura 24: Índice de continuidade para redes sobrepostas com os parâmetros N=200, C=120, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.

O interessante é que ao fazer a análise dos 30 primeiros pedaços da rede que distribuiu 120 pedaços, observou-se que o desempenho ficava muito próximo do desempenho da rede que distribuiu 30 pedaços. Isto mostra que talvez a escolha da política de inicialização de vídeo não tenha favorecido os primeiros pedaços.

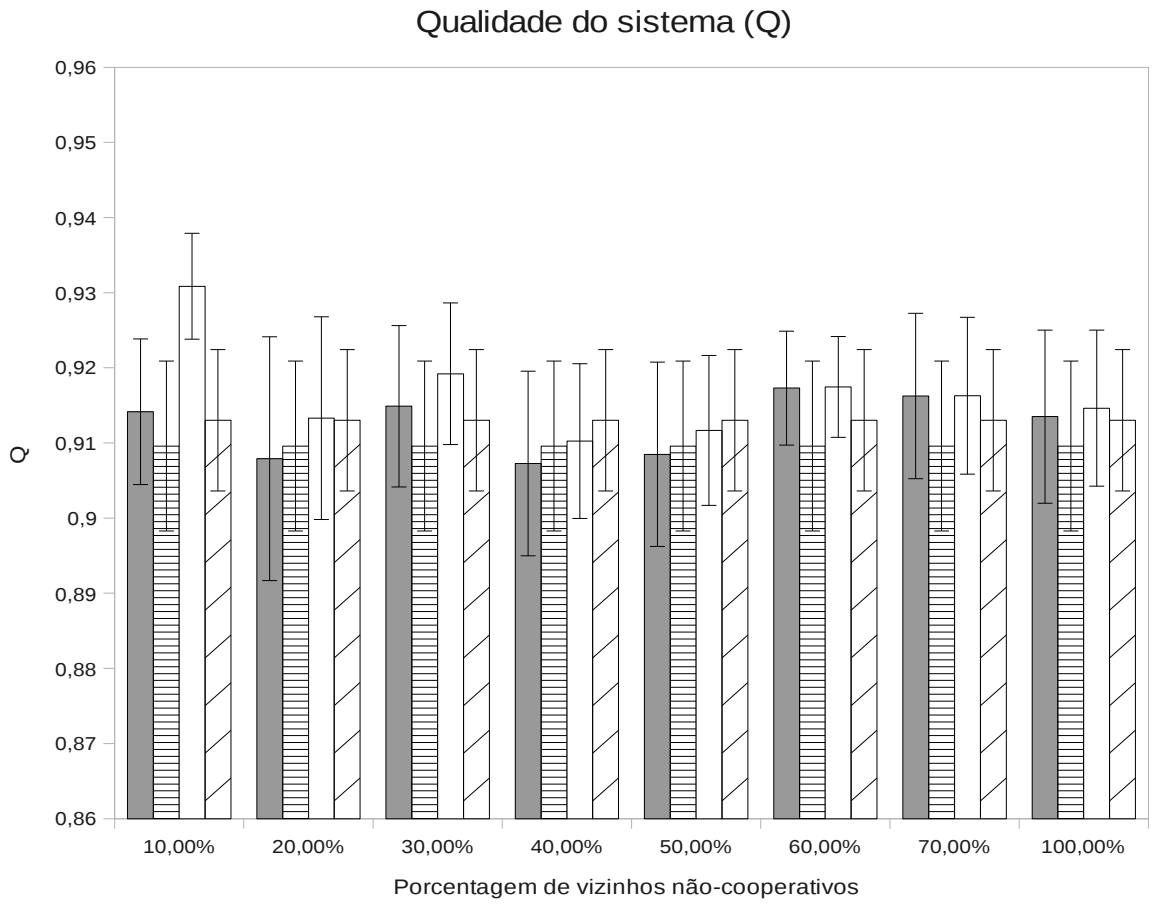
Para os nós da rede que distribuiu 120 pedaços, Figura 24, o ICO teve uma suave melhora com a adição do mecanismo de admissão. Já para a rede que distribuiu 30 pedaços não há melhora significativa, o desempenho foi muito próximo ao desempenho do modelo sem mecanismo, levando-se em consideração o intervalo de confiança de 90%.



(a) Qualidade do sistema para uma rede com N=200, C= 30, upload = 600 kb/s e 10% dos nós não-cooperativos.

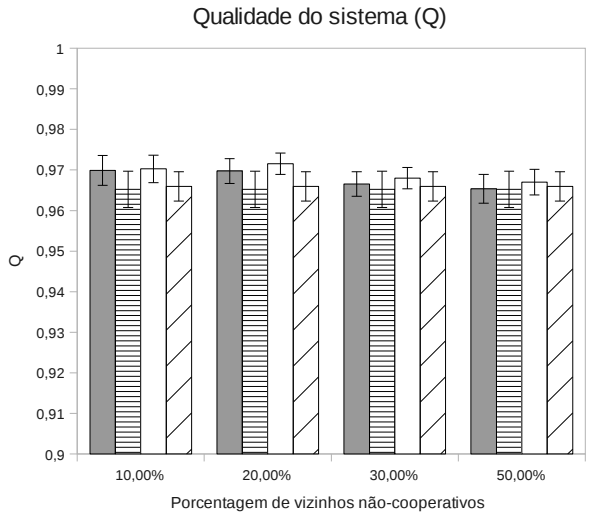


(b) Qualidade do sistema para uma rede comN=200, C= 30, upload = 600 kb/s e 15% dos nós não-cooperativos.

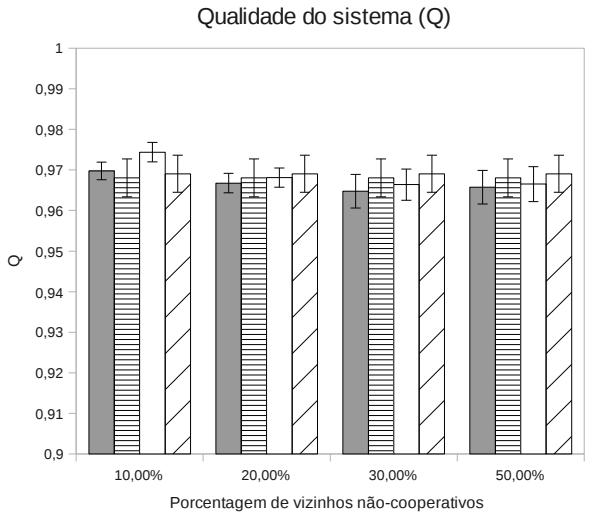


(c) Qualidade do sistema para uma rede com N=200, C= 30, upload = 600 kb/s e 20% dos nós não-cooperativos.

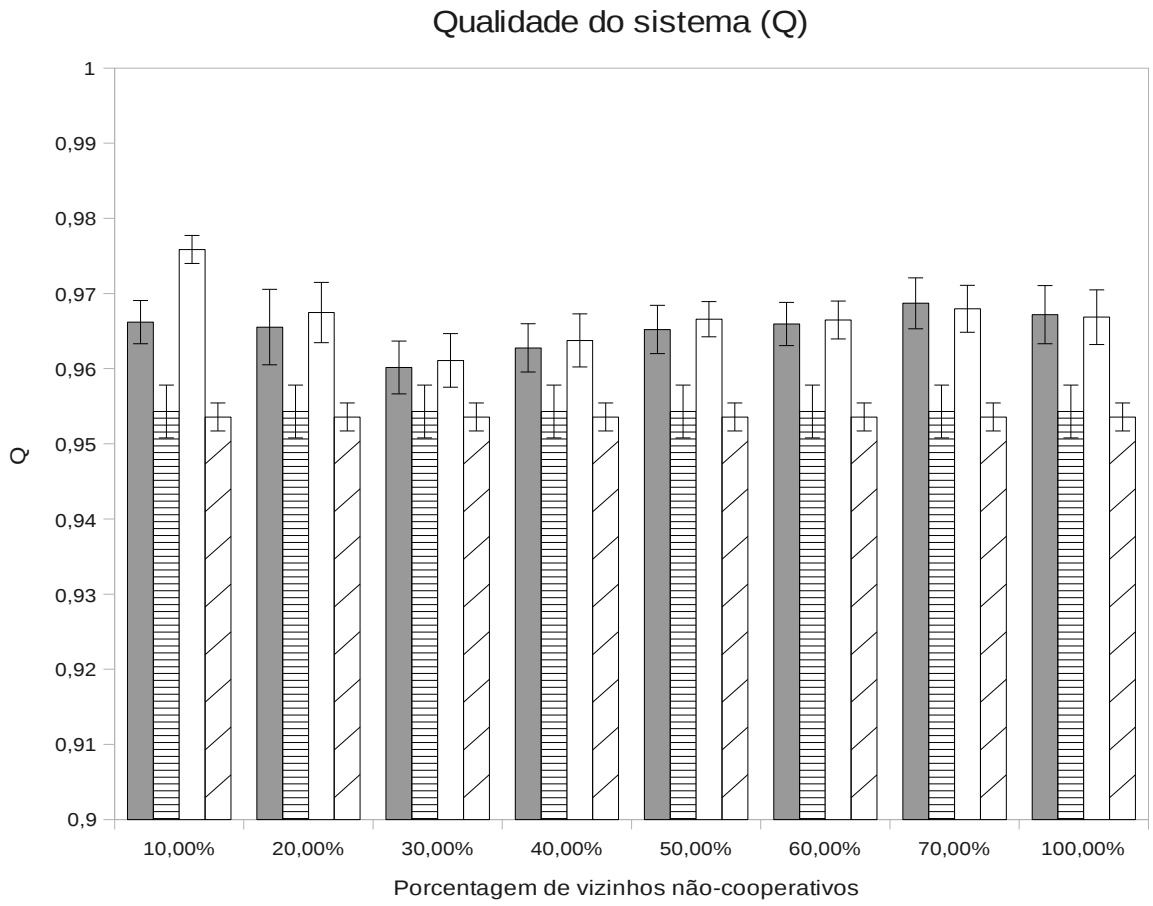
Figura 25: Qualidade do sistema de vídeo para redes sobrepostas com os parâmetros N=200, C=30, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.



(a) Q para uma rede com N=200, C= 120, upload = 600 kb/s e 10% dos nós não-cooperativos.



(b) Q para uma rede com N=200, C= 120, upload = 600 kb/s e 15% dos nós não-cooperativos.



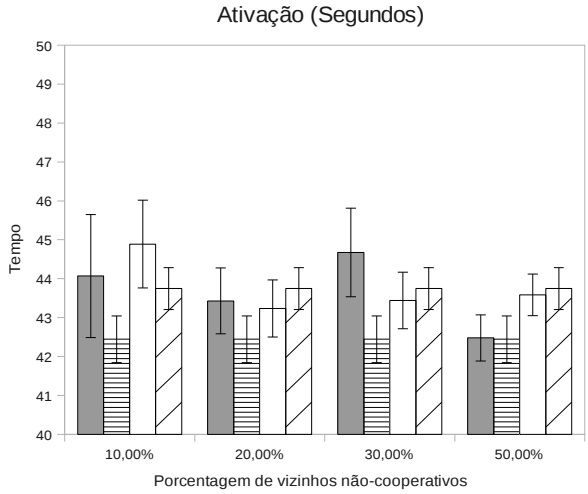
(c) Q para uma rede com N=200, C= 120, upload = 600 kb/s e 20% dos nós não-cooperativos.

Figura 26: Qualidade do sistema de vídeo para redes sobrepostas com os parâmetros N=200, C=120, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.

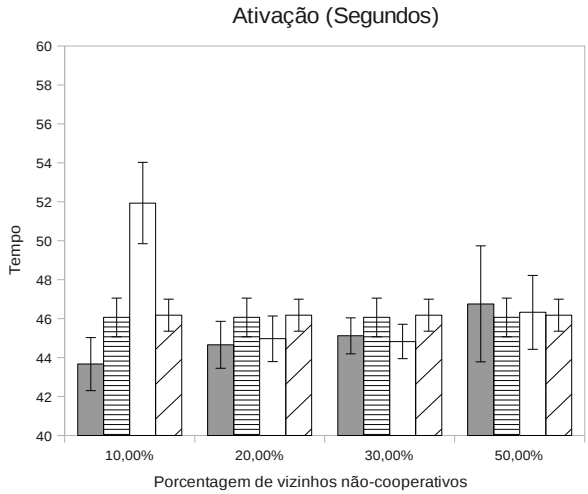
As Figuras 25 e 26 apresentam o desempenho da rede sob o ponto de vista da métrica qualidade do sistema, esta métrica obteve um comportamento muito similar à métrica ICO. Em geral, os nós não-cooperativos com melhor desempenho que nós cooperativos. A diferença substancial é que houve uma melhora significativa da rede para ambos os casos, e não apenas para a rede que distribuiu 120 pedaços, para porcentagem baixa de vizinhos não-cooperativos, igual a 10%.

As Figuras 27 e 28 apresentam o desempenho da rede sob o ponto de vista da métrica atraso inicial. Para que o gráfico não ficasse muito “poluído”, decidimos mostrar

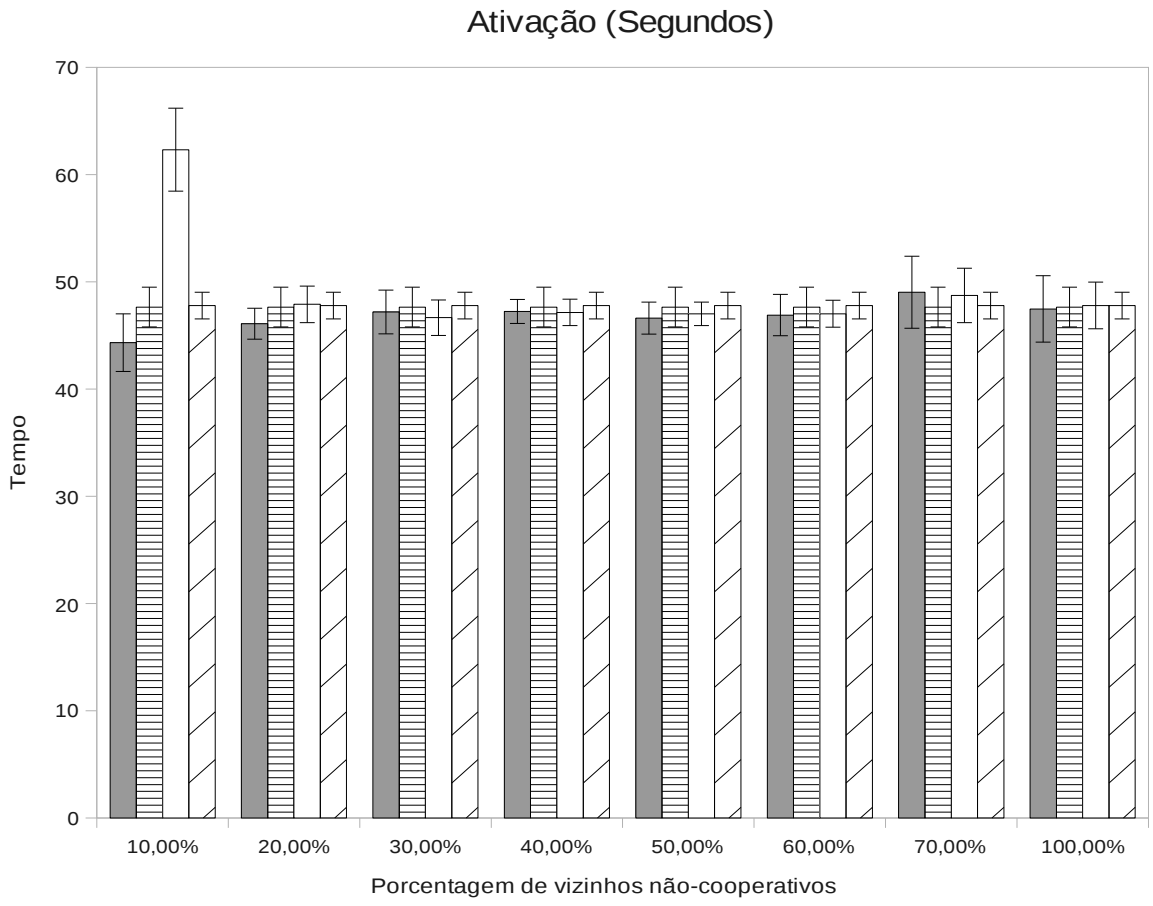
separadamente o tempo de ativação da rede, já que conforme vimos, o tempo de ativação influencia muito o atraso inicial e portanto também deve ser apresentado. Decidimos apresentar apenas os gráficos relacionados a rede que distribuiu 30 pedaços, pois conforme foi visto, os gráficos para 30 e 120 pedaços ficaram praticamente iguais, o mesmo comportamento ocorreu nestas simulações.



(a) Ativação para uma rede com N=200, C= 30, upload = 600 kb/s e 10% dos nós não-cooperativos.

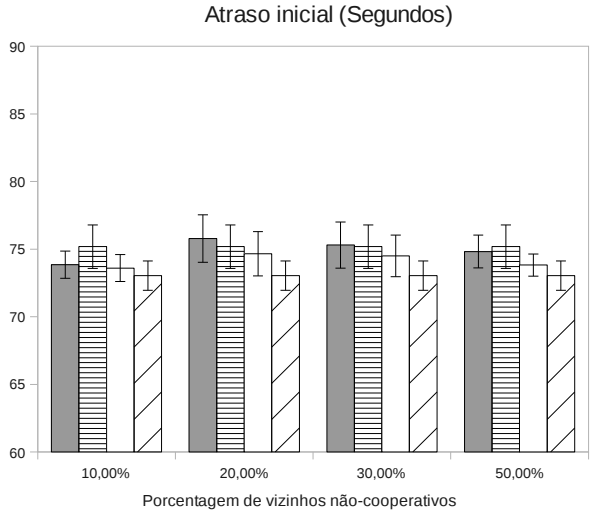


(b) Ativação para uma rede com N=200, C= 30, upload = 600 kb/s e 15% dos nós não-cooperativos.

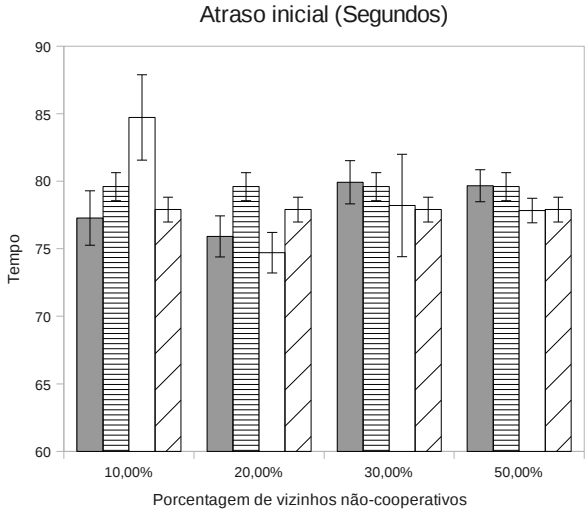


(b) Ativação para uma rede com N=200, C= 30, upload = 600 kb/s e 20% dos nós não-cooperativos.

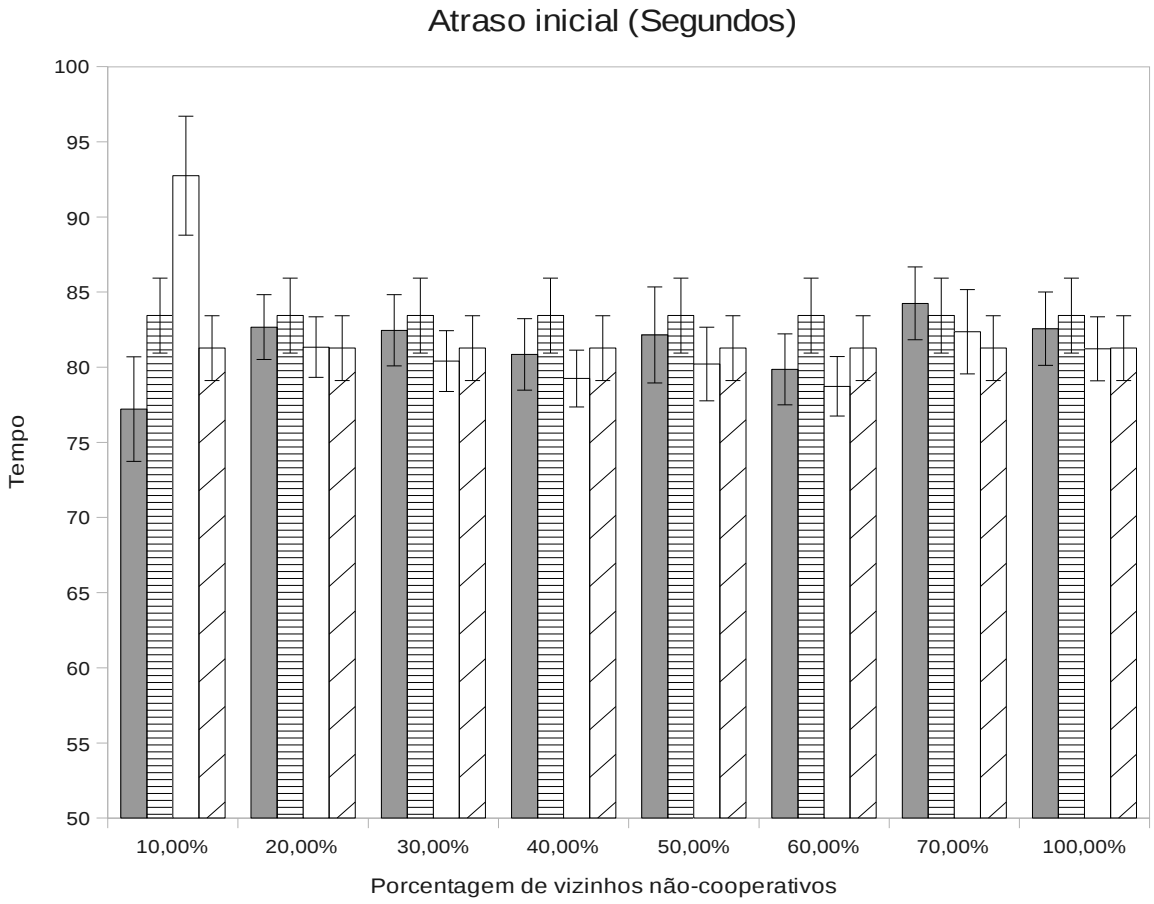
Figura 27: Tempo de ativação para redes sobrepostas com os parâmetros N=200, C=30, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.



(a) Atraso inicial para uma rede com N=200, C= 30, upload = 600 kb/s e 10% dos nós não-cooperativos.



(b) Atraso inicial para uma rede com N=200, C= 30, upload = 600 kb/s e 15% dos nós não-cooperativos.



(c) Atraso inicial para uma rede com N=200, C= 30, upload = 600 kb/s e 20% dos nós não-cooperativos.

Figura 28: Atraso inicial para redes sobrepostas com os parâmetros N=200, C=30, banda de upload = 600 kb/s e nós não cooperativos 10, 15 e 20%.

As Figuras 27.a e 28.a apresentam o comportamento das métricas ativação e atraso inicial respectivamente, das redes com 10% de nós não-cooperativos. Para este caso, quase não há mudanças entre as redes que utilizaram o mecanismo de admissão e as rede que não utilizaram o mecanismo. Entretanto, para redes com 15% dos nós não-cooperativos (Figuras 27.b e 28.b) e redes com 20% dos nós não-cooperativos (Figuras 27.c e 28.c) há momentos em que os nós não-cooperativos são punidos e os nós cooperativos são beneficiados.

Na Figura 27.b, quando a porcentagem de vizinhos é igual a 10%, temos que o tempo de ativação para os nós cooperativos ficou próximo a 44 segundos variando 1 segundo para mais ou para menos, e para nós não-cooperativos o tempo ficou próximo a 51 segundos

variando 2 segundos para mais ou para menos de acordo com o intervalo de confiança adotado. Sem o mecanismo, o tempo de ativação para ambos ficou próximo a 46 segundos.

Na Figura 28.c, para porcentagem de vizinhos igual a 10%, temos que o atraso inicial dos nós cooperativos ficou próximo a 77 segundos, variando 3 segundos para mais ou para menos, e dos nós não-cooperativos ficou próximo a 93 segundos variando 4 segundos para mais ou para menos. Sem o mecanismo, o atraso inicial para os nós cooperativos ficou próximo a 83 segundos e para os nós não-cooperativos próximo a 81 segundos.

Para os dois casos citados, os nós cooperativos obtiveram uma pequena melhora de desempenho e os nós não-cooperativos acabaram sendo punidos.

CAPÍTULO 4 - PROPOSTA PARA MECANISMO DE ATUALIZAÇÃO DOS VIZINHOS

Como foi visto no capítulo anterior, o protocolo *push/pull* tem seu desempenho prejudicado à medida que o número de nós não-cooperativos aumenta na rede sobreposta. Com o novo mecanismo de admissão de vizinhos, o desempenho dos nós cooperativos melhorou para valores baixos de nós vizinhos não-cooperativos e não ocorreu mais a situação de um nó cooperativo não ser ativado, apenas nós não-cooperativos tiveram este problema. Entretanto, os nós não-cooperativos que conseguiram ser ativados continuaram tendo melhor desempenho que os nós cooperativos na maioria das situações.

O mecanismo de admissão conseguiu atingir um ponto importante, a melhora do desempenho da rede como um todo, principalmente quando a quantidade máxima de vizinhos não-cooperativos admitidos por um nó era menor que 20%. O controle admissional de vizinhos tornou a rede mais balanceada em termos de nós não-cooperativos, ou seja, nenhum nó cooperativo ficou sobrecarregado com nós não-cooperativos.

O fato de um nó cooperativo possuir um nó não-cooperativo como vizinho não é de todo ruim, visto que definimos que um nó não-cooperativo não entrega pedaços de forma voluntária, mas entrega um pedaço quando solicitado. Desta forma, possuir poucos nós não-cooperativos como vizinhos, pode até ser benéfico, pois os mesmos têm a tendência de possuir pedaços de forma mais rápida. Entretanto esta afirmação não é válida para alguns casos, por exemplo: quando um nó cooperativo é o único nó cooperativo de um nó não-cooperativo.

Neste caso o nó cooperativo tem seu desempenho prejudicado, principalmente em métricas que medem atrasos, como ICO e qualidade. Isto ocorre porque ao fazer uma requisição *pull*, o nó seleciona de forma aleatória um vizinho para fazer a requisição, se o vizinho escolhido for este nó não-cooperativo, ele dificilmente terá o pedaço desejado, já que o único nó que envia pedaços de maneira voluntária ao mesmo é o próprio nó requisitante. Com isto, este nó cooperativo perderá tempo na maioria das vezes que solicitar pedaços a este nó não-cooperativo e terá que fazer uma nova requisição a outro nó.

Pode ser percebido que o mecanismo de admissão já consegue impor alguma justiça com relação ao comportamento dos nós e a qualidade obtida pelos mesmos, entretanto, deveria haver um mecanismo mais dinâmico, capaz de ver os erros cometidos na formação das vizinhanças de cada nó e corrigi-los. O fato da formação do grafo de vizinhança ser estático, atrapalha o desempenho da rede como um todo, pois os nós acabam não sendo estimulados a cooperar, já que não há um incentivo para isto, pois a condição inicial pré-estabelecida não se altera.

Diante disso, resolvemos propor um mecanismo dinâmico de atualização dos vizinhos para cada nó, que deverá ser executado de tempos em tempos durante a simulação. Isto ajudaria a agrupar nós cooperativos, aumentando a qualidade de vídeo recebida por cada um. A idéia básica é criar um *ranking* de pontuação central baseado nos pedaços recebidos e transmitidos por cada nó, e posicionar cada nó neste *ranking*, de forma que seja possível comparar o posicionamento de cada um. Se no momento da atualização do conjunto de vizinhos de um determinado nó, um dos vizinhos não preencher o requisito desejado, o mesmo será removido do conjunto de vizinhos. Após esta etapa, haveria a etapa de adição de vizinhos que também deveria preencher certos requisitos.

4.1 Mecanismo

O sistema funcionaria da seguinte forma com o novo mecanismo. Cada nó armazenaria localmente a quantidade de pedaços recebidos e pedaços enviados de cada nó vizinho. De tempos em tempos, o nó calcularia a taxa de cada vizinho e obteria uma pontuação *P* de 0 a 10 para cada um. O nó enviaria uma mensagem de “atualizar tabela” para uma entidade central com a sua tabela de pontuação. Note que cada nó só enviaria pontuações referentes aos seus vizinhos, ou seja, o nó não precisaria conhecer a pontuação dos demais nós da rede. As equações 4 e 5 representam a fórmula para obter a taxa de cada vizinho para um dado nó.

$$T_v = \frac{p_{Rv}}{(p_{Tv} + p_{Rv})} \quad \text{, para } p_{Rv} > 0 \tag{4}$$

$$T_n = 0 \quad \text{, para } p_{Rv} = 0 \tag{5}$$

onde T_v é igual a taxa do vizinho *V*, P_{RV} é igual ao número de pedaços recebidos de *V*, e P_{TV} é igual ao número de pedaços transmitidos para *V*.

A entidade central receberia a tabela de pontuação de cada nó e ao longo do tempo teria uma visão global da pontuação *P* de todos os nós. A cada mensagem "atualizar tabela" recebida, a entidade atualizaria na tabela global a pontuação de cada nó contido na mensagem. Isso seria feito através de métodos estatísticos ou através de inteligência computacional capazes de consolidar a pontuação de cada nó.

De tempos em tempos, que não precisa ser necessariamente igual a montagem da tabela de pontuações, cada nó avaliaria seus próprios vizinhos. Caso o nó observasse que um vizinho não faz mais parte da rede sobreposta, ou que a pontuação do seu vizinho está abaixo da sua pontuação por um fator pré-estabelecido, o nó removeria o nó vizinho e enviaria uma mensagem ao mesmo avisando de sua decisão.

A seguir o nó contactaria a entidade central, enviaria uma mensagem de "atualizar tabela" com a sua nova tabela de vizinhos e suas respectivas pontuações, e também enviaria uma mensagem de "solicitar vizinho" para obter um novo vizinho que preenchesse determinadas condições pré-estabelecidas. A Figura 29 apresenta um pseudo-código do algoritmo que cada nó utilizaria.

Algoritmo de atualização dos vizinhos de cada nó
Parâmetros de entrada: $P_{Rv}(v)$ → vetor contadores de pedaços recebidos do vizinho, $P_{Tv}(v)$ → vetor contadores de pedaços transmitidos ao vizinho.

Nomenclatura:
A(t) → Conjunto dos nós ativos no sistema
$V_n(t)$ → Conjunto dos nós vizinhos de n.
P_v → Pontuação do vizinho v
V → Quantidade de vizinhos na lista = lenght[$V_n(t)$]

V_{MAX} → Quantidade máxima de vizinhos

#Parte 1 – Calcular taxa e pontuação de cada nó

iniciar
para todo v variando de 0 a V **faça**
se (P_R(v)) > 0 **então**

$$T_v = \frac{p_{Rv}}{(p_{Tv} + p_{Rv})}$$

$$P(v) = inteiro[10 \times T(v)]$$

senão

$$T(v) = .$$

$$P(v) = 0$$

fim do se
fim do para
enviar mensagem de “atualização de tabela” à entidade central
fim

#Parte 2 – Remover os nós vizinhos que saíram ou que renderam mal

iniciar
num_vizinhos = V
para todo v variando de 0 a num_vizinhos **faça**
se ((v ∉ A(t)) || $(\frac{P(v)}{P(n)} < k)$) & P(n) <> 0 **então**
remover (v, V_n(t))
enviar mensagem de “remoção de vizinho” ao vizinho
fim do se
fim do para
se comprimento de V < VMAX **então**
enviar mensagem de “atualização de tabela” à entidade central
enviar mensagem de “solicitar vizinho” à entidade central
fim do se
fim

#Parte 4 – Adicionar um novo vizinho

iniciar
enviar mensagem “adicionar vizinho”
se resposta de v == OK
adicionar (v, V_{node}(t))
enviar mensagem de “atualização de tabela” à entidade central

```
senão
       enviar mensagem “solicitar vizinho” a entidade central
fim do se
fim

#Parte 5 – Responder a um nó solicitante

iniciar
Enviar mensagem “consultar candidato” à entidade central
Receber resposta da entidade (vizinho, P(vizinho))
se (  (P(v)/P(n) > k)  ) então
       adicionar (v, V_node(t))
       enviar mensagem “OK” para V
       enviar mensagem de “atualização de tabela” à entidade central
senão
       enviar mensagem “NOK” para V
fim do se
fim

#Parte 5 – Remover um nó vizinho por solicitação

iniciar
se Receber mensagem “remover vizinho” de um vizinho v faça
       remover (v, V_n(t))
fim do se
fim
```

Figura 29: Pseudo-código do algoritmo para atualização do conjunto de vizinhos de cada nó.

Ao receber uma mensagem de “solicitar vizinho”, a entidade central deveria ser capaz de selecionar um novo nó para parceiro do nó requisitante, uma vez que a mesma teria um *ranking* com a pontuação de todos os nós da rede e a lista de vizinhos de cada um. Para que a nova parceria fosse estabelecida, ambos os nós precisariam preencher os requisitos pré-estabelecidos. A entidade central faria este julgamento antes de enviar a resposta para o nó que fez a solicitação. A Figura 30 apresenta o pseudo-código do algoritmo da entidade central.

Após receber a sugestão de nó vizinho da entidade central, o nó solicitante enviaria uma mensagem de “adicionar vizinho” ao nó sugerido. O nó sugerido enviaria uma mensagem de “consultar candidato” à entidade central, antes de aceitar, apenas para receber a pontuação do nó solicitante a fim de se certificar que o nó atende aos requisitos mínimos, e

para verificar se o nó o encontrou via entidade, para ter certeza que não se trata de um nó mal intencionado.

Após confirmar que o nó solicitante pode ser seu vizinho, o nó candidato o adiciona como vizinho e o envia uma mensagem de aceitação, mensagem “OK”. O nó solicitante finalmente adiciona o nó candidato como vizinho.

```
Algoritmo de atualização dos vizinhos de cada nó

#variáveis
mensagem
candidato-encontrado

Requisitos: A(t) > 0 e n ∈ A(t)

#Parte 1 – Atualizar a tabela de pontuação e a lista de vizinhos

início
Recebe mensagem

Caso mensagem == “atualização de tabela” faça
       Atualizar tabela geral de pontuação
       Atualizar listas de vizinhos de N e V

Caso mensagem == “solicitar vizinho” faça
       para todo n ∈ A(t) && !candidato-encontrado faça
              se (P(vizinho) < P(solicitante))  &&  (P(vizinho)/P(solicitante) > k)  &&
                 (solicitante <> vizinho) && (vizinho ∉ V_solicitante(t)) &&
                 (lenght(V_vizinho(t)) < VMAX) ) então
                     candidato-encontrado = true
                     ENVIAR SUGESTÂO (candidato,P(candidato)) ao nó solicitante
              fim se
       fim para
       se candidato-encontrado = false então
              Enviar mensagem “candidato não encontrado” para nó solicitante
       fim se

Caso mensagem == “consultar candidato” faça
       Buscar pontuação do nó investigado
       Enviar mensagem (vizinho,P(vizinho) ao nó solicitante

fim
```

Figura 30: Pseudo-código do algoritmo da entidade central do mecanismo de atualização proposto.

4.2 No simulador

Em termos de simulação, a modelagem poderia ser simplificada já que o simulador é uma entidade "onipresente", uma entidade que consegue obter facilmente a quantidade real de pacotes recebidos e enviados por cada nó em um determinado intervalo de tempo. Este subterfúgio não é incoerente, visto que métodos estatísticos ou inteligência computacional deveria estimar valores bem próximo ao valor da situação real descentralizada sugerida anteriormente.

Para conseguir criar este *ranking* durante a simulação, seria necessário a criação de contadores no simulador. Contadores de pedaços recebidos (p_R) e transmitidos (p_T) que seriam criados para cada nó *n*. De tempos em tempos, esses valores seriam contabilizados e a taxa de cada nó seria calculada, conforme as equações 6 e 7:

$$T_n = \frac{p_{Tn}}{(p_{Rn} + p_{Tn})} \quad \text{, para } p_{Tn} > 0 \tag{6}$$

$$T_n = 0 \quad \text{, para } p_{Tn} = 0 \tag{7}$$

Veja que T_n é máximo quando p_{Rn} é igual a 0, ou seja, o nó não recebeu nenhum pedaço, o que ocorre com o nó fonte, ou com algum nó que já tenha completado seus pedaços e continua compartilhando. Quanto mais o nó enviar maior será a sua taxa, e quanto menos enviar, menor será sua taxa.

Após contabilizar a taxa de cada nó, seria necessário montar um *ranking* com valores inteiros de 0 a 10. Para obter esses valores, bastaria multiplicar o valor da taxa de cada nó por dez e desconsiderar as casa decimais do número obtido. Assim, cada nó estaria associado a uma pontuação P de 0 a 10, quanto mais próximo a 10 mais cooperativo seria o nó, quanto mais próximo a zero menos cooperativo seria o nó.

No simulador, o algoritmo de atualização dos vizinhos de cada nó seria responsável por realizar toda a tarefa descrita acima e mais as tarefas de remover os nós vizinhos que não esteja atingindo os requisitos necessários e adicionar novos vizinhos a um determinado nó. A Figura 31 apresenta o pseudo-código do algoritmo do mecanismo que sugerimos que seja introduzido no simulador para tornar o sistema mais justo, ou seja, nós agrupados a vizinhos com pontuação próximas a dele.

Na primeira etapa do pseudo-código do algoritmo da Figura 31, a taxa de cada nó é calculada e depois normalizada para que seja obtida a pontuação de cada um. Pode ser observado que se a quantidade de pacotes transmitidos for igual a zero, a pontuação do nó também será igual a zero independente da quantidade de pacotes recebidos. Percebe-se que a pontuação é diretamente proporcional à quantidade de pacotes transmitidos pelo nó.

A segunda etapa do algoritmo trata da remoção da lista de vizinhos de cada nó, os vizinhos que saíram da rede sobreposta ou que colaboraram pouco. Cada nó faz a verificação de cada vizinho, observando se o mesmo não está mais no conjunto dos nós ativos na rede ou se a razão entre a pontuação do nó vizinho e a pontuação do nó é menor que um fator k desejado.

A etapa três é responsável pela adição de novos vizinhos para os nós que ainda não atingiram o número máximo de vizinhos. Um vetor R seria criado para conter todos os nós ativos em ordem decrescente de pontuação, ou seja, o nó que possuísse a melhor pontuação seria o primeiro elemento do vetor.

Sendo assim, o nó mais cooperativo seria o primeiro a verificar se precisaria de um novo vizinho, e caso precisasse, sortearia de maneira aleatória um novo vizinho dentro do conjunto dos nós ativos. Após o sorteio, haveria a etapa de verificação para admitir ou não o novo nó como vizinhos. Para que seja admitido como vizinho, o nó candidato deveria preencher as seguintes condições:

1. A pontuação do nó candidato deve ser menor que a pontuação do nó. Esta condição é necessária para garantir que um nó com pontuação baixa consiga obter novos bons vizinhos;

2. A razão entre pontuação do nó candidato e a pontuação do nó deve ser maior que um fator *k* desejado. Esta condição garante que um nó cooperativo não aceite novamente um nó com pontuação muito baixa;
3. O nó não pode adicionar à sua lista de vizinhos ele próprio;
4. O nó candidato não deve já pertencer ao conjunto de vizinhos do nó;
5. O número de elementos do conjunto de vizinhos do nó candidato deve ser menor que a quantidade máxima de vizinhos. Isto evita que um nó muito cooperativo acabe sobrecarregado.

Algoritmo de atualização dos vizinhos de cada nó
Parâmetros de entrada: $P_R(n)$ → vetor contadores de pedaços recebidos pelo nó, $P_T(n)$ → vetor contadores de pedaços transmitidos pelo nó.

Nomenclatura:
A(t) → Conjunto dos nós ativos no sistema
$V_n(t)$ → Conjunto dos nós vizinhos de n.
P_n → Pontuação do nó n
N → Quantidade de nós na rede = lenght[A(t)]
V_{MAX} → Quantidade máxima de vizinhos

Requisitos: t = atualizar, A(t) > 0 e n ∈ A(t)

#Parte 1 – Calcular taxa e pontuação de cada nó
para todo n variando de 0 a N **faça**
se (PT(n)) > 0 **então**

$$T(n) = \frac{p_{Tn}}{(p_{Rn} + p_{Tn})}$$

$$P(n) = inteiro[10 \times T(n)]$$

senão

$$T(n) = \cdot$$

$$P(n) = 0$$

fim do se
fim do para

#Parte 2 – Remover os nós vizinhos que saíram ou que renderam mal
para todo n variando de 0 a N **faça**
para todo v variando de 0 a lenght[$V_n(t)$] **faça**

```
            se ( (v ∉ A(t)) ||   (P(v)/P(n) < k)  ) & P(n) <> 0 então
                  remover (v, Vn(t))
                  remover (n, Vv(t))
            fim do se
      fim do para
fim do para

#Parte 3 – Adicionar novos vizinhos aos nós.
#Criar um vetor R ordenando os nós pela pontuação R(n).

j = N
para todo n variando de 0 a N faça
      node = R(n)
      contadorvizinhos = length (Vnode(t))
            enquanto (contadorvizinhos < VMAX  & j > 0) faça
                  id = valorrandomico[0,j]
                  vizinho = A(id)
                  se ( (P(vizinho) < P(node))  &&  (P(vizinho)/P(node) > k)  && (node <>
                  vizinho) && (vizinho ∉ Vnode(t)) && (lenght(Vvizinho(t)) < VMAX) )
                  então
                        adicionar (v, Vnode(t))
                        adicionar (n, Vvizinho(t))
                        tmp = A(j – 1)
                        A(j – 1) = A(id)
                        A(id) = tmp
                        j - -
                        contadorvizinhos + +
                  senão
                        tmp = A(j – 1)
                        A(j – 1) = A(id)
                        A(id) = tmp
                  fim do se
            fim do enquanto
fim do para
```

Figura 31: Pseudo-código do algoritmo para o mecanismo de atualização proposto para ser inserido no simulador.

Importante destacar que as condições listadas acima devem ser sequenciais, ou seja, se a primeira condição já não for satisfeita, o teste deve retornar falso de imediato. Isto garante que nunca seja testada a condição *p(vizinho)/p(node)* quando *p(node)* fosse igual a zero, para não ocorrer inconsistência.

Caso todas as condições sejam satisfeitas, os dois nós passam a ser vizinhos um do outro. As outras operações servem apenas para garantir que o nó candidato não seja escolhido novamente na seleção aleatória.

O mesmo nó faria todo esta etapa novamente até atingir o número máximo de vizinhos admitidos ou até que descubra que no momento do teste não há nenhum outro nó no conjunto dos nós ativos com condições de atendê-lo bem, pois já foi visto que em alguns casos, é melhor ter vizinhos a menos do que vizinhos que não contribuam para o bom desempenho do sistema. Após o nó passar por todos esses testes, chegaria a vez do nó seguinte. O novo nó passaria por todas as etapas descritas anteriormente e assim sucessivamente.

4.3 Observações

Acreditamos que o novo mecanismo de atualização promova uma justiça maior na rede com relação a contribuição/qualidade de vídeo de cada membro. O nó que contribuísse mais seria recompensado por isso, e o nó que contribuísse menos acabaria sendo punido ao ser agrupado a nós com o mesmo tipo de comportamento quanto à contribuição.

O mecanismo faria esta tarefa de maneira gradativa e contínua, quanto mais tempo uma rede ficasse distribuindo vídeo, mais ela tenderia a ficar estável e justa para seus membros participantes. Isto aconteceria porque os nós ditos normais, com boa pontuação, iam aos poucos removendo os nós não-cooperativos de sua lista de vizinhos, substituindo-os por vizinhos cooperativos. Isso faria com que a qualidade do vídeo fosse alta no momento da reprodução.

Em contrapartida, um nó não-cooperativo sentiria o efeito contrário. Com a pontuação baixa, deixaria de ter parceiros cooperativos, o que o levaria a ter uma baixa qualidade na reprodução do vídeo. Esta punição pode vir a fazer com que o nó não-cooperativo pense na possibilidade de cooperar mais para obter uma melhor qualidade. De certa forma, o mecanismo serviria para incentivar todos da rede a cooperar, e isso faria com que o desempenho da rede como um todo melhorasse, chegando próximo ao melhor desempenho da rede, que conforme vimos, ocorre quando há 0% de nós não-cooperativos.

Para avaliar este último mecanismo proposto seria necessário inseri-lo no simulador PeerSim. A alteração de várias classes, incluindo as que representam os nós, e o *protocolo push/pull*. Além disso, um novo protocolo para implementar o mecanismo deveria ser integrado ao código. Com isso seria possível realizar uma nova sequência de simulações para comparar os resultados das mesmas com os obtidos anteriormente.

Além disso, seria interessante avaliar os mecanismos propostos em um sistema real para comprovar os benefícios e a consistência dos mesmos.

Estas atividades devem ser realizadas em trabalhos futuros.

CAPÍTULO 5 CONCLUSÕES

A transmissão de fluxos de vídeo baseada em modelos onde os usuários do sistema devem cooperar na retransmissão do fluxo tem sido pesquisada há muito tempo (Kon et al, 1998) (Borko et al, 1999) por exemplo. Nestes modelos o usuário é somente admitido se estiver disposto a cooperar. Desta forma o sistema consegue obter uma certa escalabilidade e garantia de que recursos mínimos serão acrescentados ao sistema a cada nova admissão. Atualmente esta abordagem tem sido investigada e aplicada sobre os modelos de redes P2P onde os nós participantes têm uma certa autonomia para decidir como vão participar e cooperar no sistema. Se por um lado as infra-estruturas e protocolos para redes P2P estão relativamente bem estabelecidos, garantir que recursos mínimos estarão disponíveis para a transmissão de vídeo para todos os nós participantes não é trivial. Isso ocorre justamente pela relativa autonomia que cada nó tem para definir seu grau de cooperação. Determinado nó pode decidir não cooperar com a doação de recursos para a retransmissão de fluxo de vídeo, embora queira receber o fluxo para seu consumo.

Neste contexto, o presente trabalho investigou (i) como nós não cooperativos afetam a qualidade da aplicação de distribuição de fluxos de vídeo da uma rede P2P onde co-existem nós cooperativos e não cooperativos; (ii) se um mecanismo de controle de admissão, onde nós cooperativos pudessem selecionar vizinhos cooperativos e preterir vizinhos não-cooperativos, poderia resultar em uma forma de incentivo à cooperação tornando a rede mais justa para os nós cooperativos. A partir desta investigação um mecanismo dinâmico para a seleção de vizinhos é proposto com o objetivo de permitir aos nós cooperativos manter o serviço de distribuição de vídeo em uma qualidade aceitável com um “equilíbrio” de vizinhos cooperativos e não-cooperativos.

Para o trabalho selecionamos redes P2P que fazem uso de protocolos de difusão de pedaços *push/pull*. Utilizamos o simulador PeerSim e o módulo P4S, fazendo alterações e acréscimos necessários para incluir os aspectos sendo investigados.

Na primeira etapa do trabalho estudou-se o desempenho das redes P2P selecionadas face à presença de nós com comportamento não-cooperativo utilizando-se simulação. Alteramos as classes *peersim.core.GenericNode, peersim.core.Network, p4s.core.Alternate* do

simulador P4S/PeerSim, conforme explicado no Capítulo 3. Estas alterações foram responsáveis pela possibilidade de incluir nós com comportamento anormal na rede sobreposta. Um nó com comportamento não-cooperativo deixa de enviar voluntariamente pedaços durante o estado *PUSH*, ele apenas envia pedaços durante o estado *PULL* ao ser solicitado.

Foram realizadas 300 rodadas de simulação, variando a porcentagem de nós não-cooperativos na rede em 0%, 5%, 10%, 15% e 20%, e mais 480 rodadas variando a porcentagem de nós não-cooperativos da lista de vizinhos. Nas primeiras 300 rodadas, metade das mesmas foi executada com a rede possuindo nós com banda de *upload* fixada em 600 kb/s. A segunda metade foi executada com a banda variando entre os valores 200, 360, 600 e 1000 kb/s.

Para mensurar o desempenho do protocolo de difusão de pedaços *push/pull*, nós utilizamos quatro métricas de qualidade diferentes: tempo máximo, índice de continuidade, qualidade do sistema de vídeo e atraso inicial. As quatro métricas foram definidas na seção 3.2.1.1. Os resultados comprovaram que o desempenho da aplicação *push/pull* diminuía à medida que a porcentagem de nós não-cooperativos aumentava. Em geral todas as métricas tiveram melhor valor quando a porcentagem de nós não-cooperativos era igual a 0%. Observamos, também, que os resultados dos nós não-cooperativos foram melhores do que os resultados dos nós cooperativos, ressaltando um outro lado do problema.

Um outro ponto observado é que em algumas rodadas de simulação ocorreu de nós cooperativos sequer serem ativados. Além disso, vale a pena comentar que para porcentagem de nós não-cooperativos mais alta, como 50% por exemplo, a transmissão de vídeo tornou-se impraticável, pois o desempenho da rede ficou bastante degradado e vários nós sequer foram ativados.

Foi visto também que o modelo simulado (redes P2P utilizando protocolo de difusão *push/pull* e grafo estático) ainda não está preparado para bandas de *upload* variáveis, pelo menos, variando com bandas de valor baixo como 200 e 360 kb/s, o modelo é bem dependente de como o grafo é inicialmente criado, já que o grafo não é alterado ao longo da simulação, salvo quando ocorre a saída ou entrada de um nó, o que não foi feito durante as simulações. Os resultados mostraram que todas as métricas tiveram seus valores prejudicados.

Diante disto, o restante das simulações foi feito com banda fixada em 600 kb/s, para apenas observar o parâmetro quantidade de nós não-cooperativos.

Após as análises da primeira etapa, introduzimos no modelo um mecanismo para controlar a quantidade de vizinhos não-cooperativos na lista de vizinhos de cada nó, fazendo com que um nó cooperativo só admitisse um percentual máximo de vizinhos não-cooperativos na sua lista. Esta nova alteração tinha como propósito melhorar o desempenho da rede P2P com o protocolo *push/pull* como um todo, garantindo que todos os nós cooperativos fossem ativados e que com um número controlado de vizinhos não-cooperativos, estes obtivessem melhor desempenho do que os nós não-cooperativos.

Para incluir esta nova alteração no simulador, foi necessário modificações no método *WirekOutUnd* da classe *peersim.graph. GraphFactory*, responsável pela geração do grafo da rede. Criamos um mecanismo de admissão para controlar a quantidade de nós não-cooperativos na lista de vizinhos de nós cooperativos. A partir de um determinado limiar definido, o nó cooperativo rejeita novos nós não-cooperativos para sua lista.

Os resultados mostraram que o desempenho da aplicação melhorou com a adição do novo mecanismo para valores mais baixos, até 20%, de porcentagem de nós não-cooperativos na lista de vizinhos dos nós cooperativos. Observamos que o controle de admissão de vizinhos tornou a rede mais balanceada em termos de nós não-cooperativos. Além disso, não se observou mais o problema de nós cooperativos não serem ativados. Este fenômeno ocorreu apenas com os nós não-cooperativos, o que funciona como um mecanismo de punição.

Entretanto, o controle de admissão não conseguiu solucionar um problema: o fato de nós não-cooperativos ativados continuarem tendo melhores resultados do que nós cooperativos. Este problema está relacionado ao fato da formação do grafo ser estático, ou seja, uma vez formado o grafo, ele não sofre mais alteração. Desta forma, o bom desempenho dos nós fica sujeito apenas à condição inicial em que o grafo é formado, não há qualquer incentivo à cooperação dos nós após a formação deste.

Assim, numa última etapa do trabalho, foi proposto um mecanismo dinâmico de atualização da lista de vizinhos de cada nó. Este mecanismo não foi simulado e seus efeitos devem ser comprovados. Acreditamos, entretanto, que este mecanismo proporcionará o efeito de justiça desejado, introduzindo uma punição aos nós não-cooperativos que, ao longo da

execução da aplicação, vão estar ligados a menos nós cooperativos, tendo a qualidade do vídeo recebido degradada. Isso serviria de incentivo para o nó não-cooperativo mudar seu comportamento. Detalhes da proposta deste mecanismo encontram-se no capítulo 4.

5.1 Trabalhos futuros

Acreditamos que alguns pontos podem ser explorados em trabalhos futuros. O primeiro ponto seria inserir o mecanismo proposto no simulador PeerSim. A alteração de várias classes, incluindo as que representam os nós, e o *protocolo push/pull*. Além disso, um novo protocolo para implementar o mecanismo deveria ser integrado ao código. Com isso seria possível realizar uma nova sequência de simulações para comparar os resultados das mesmas com os obtidos anteriormente.

Consideramos que seria oportuno também o estudo do desempenho do protocolo *push/ pull* face a presença de outros tipos de nós não-cooperativos, com definições diferentes para os mesmos. Por exemplo “gradações de não-cooperação” (um nó poderia ser muito, pouco ou mediamente não-cooperativo). Poderia também “contaminar” o estado *pull* ou até mesmo “contaminar” os dois estados. Neste último caso, um nó com esta característica seria extremamente nocivo à rede, já que não contribuiria de maneira alguma.

Um outro ponto que teria grande utilidade seria a criação de um mecanismo para tornar o modelo menos sujeito à variação de banda de *upload*. Esse mecanismo provavelmente teria que ser dinâmico, similar ao mecanismo proposto para incentivo, entretanto deveria focar nas bandas disponibilizadas por cada nó.

Fazer um estudo de qual a melhor maneira de se inicializar o vídeo sem obter um grande atraso inicial e em contrapartida obter um bom desempenho na reprodução do vídeo é também uma boa sugestão para trabalhos futuros.

Por fim, um outro ponto interessante que poderia ser estudado é qual o tamanho ideal do pedaço para manter uma boa relação entre as métricas índice de continuidade e atraso inicial, pois pedaços pequenos podem acarretar em atraso inicial pequeno, mas desempenho

pobre com várias pequenas interrupções. Já pedaços grandes podem acarretar em atraso inicial alto, mas um bom desempenho ao longo da reprodução.

Como os resultados foram satisfatórios, seria interessante avaliar os mecanismos propostos em um sistema real para comprovar os benefícios e a consistência dos mesmos.

REFERÊNCIAS

ADAR, E., HUBERMAN, B.. "*Free riding on Gnutella"*. *First Monday*, volume 5, número 10. Outubro, 2000.

BALAKRISHNAN, H., KAASHOEK, M. F., KARGER, D., MORRIS, R., and STOICA, I. 2003. "Looking up data in P2P systems". *Commun. ACM* 46, 43-48 (February. 2003). DOI= http://doi.acm.org/10.1145/606272.606299

BANERJEE, S., BHATTACHARJEE, B., KOMMAREDDY, C.. 2002. "Scalable application layer multicast". In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications* , Pittsburgh, Pennsylvania, USA, August 19 - 23, 2002. DOI= http://doi.acm.org/10.1145/633025.633045

BORKO F., WESTWATER R., JEFFREY I.. 1999. "A New Approach for Radio and Video Broadcasting over the Internet". In *Proceedings of the 1999 IEEE international Conference on Multimedia Computing and Systems - Volume 02*. ICMCS. IEEE Computer Society, Washington, DC, 553, June 07 - 11, 1999. DOI= http://dx.doi.org/10.1109/MMCS.1999.778545

CARLSSON, N., EAGER, D.. 2007. "Peer-assisted on-demand streaming of stored media using BitTorrent-like protocols". In *Proceedings of the 6th international Ifip-Tc6 Conference on Ad Hoc and Sensor Networks, Wireless Networks, Next Generation internet,* Atlanta, GA, USA, May 14 - 18, 2007.

CASTRO, M., DRUSCHEL, P., KERMARREC, A., NANDI, A., ROWSTRON, A., SINGH, A.. 2003. "*SplitStream: High-bandwith multicast in cooperative environments"*. Em *ACM Symposium on Operationg Systems Principle*, SOSP, páginas 298-313, 2003.

CHU, Y., RAO, S., ZANG, H.. 2000. "A case for end system multicast (keynote address)". In *Proceedings of the 2000 ACM SIGMETRICS international Conference on Measurement and Modeling of Computer Systems,* Santa Clara, California, United States, June 18 - 21, 2000. DOI= http://doi.acm.org/10.1145/339331.339337.

CIGNO, R., RUSSO, A., CARRA, D.. "On Some Fundamental Properties of P2P Push/Pull Protocols", *IEEE*, 2008.

COSTA, C., CUNHA, I., BORGES, A.. 2004. “Analyzing client interactivity in streaming media”. In *Proceedings of the 13th international Conference on World Wide Web,* New York, NY, USA, May 17 - 20, 2004. DOI= http://doi.acm.org/10.1145/988672.988744

COULOURIS, G., DOLLIMORE, J., KINDBERG, T.. 2007. “Sistemas Distribuídos – Conceitos e Projeto”. Editora Bookman. Quarta edição. 2007.

DRUSCHEL, P., ROWSTRON, A.. 2001. “PAST: *A large-scale, persistent peer-to-peer storage utility”*. Em *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII). Schloss Elmau*. Alemanha. May, 2001.

FELDMAN, M., LAI, K., STOICA, I., CHUANG, J.. 2004. “Robust incentive techniques for peer-to-peer networks”. In *Proceedings of the 5th ACM Conference on Electronic Commerce,* New York, NY, USA, May 17 - 20, 2004. DOI= http://doi.acm.org/10.1145/988772.988788

GNU 2000. *“The Gnutella protocol specification”* 2000. http://www.content-networking.com /papers/gnutella-protocol-04.pdf. Página acessada em 29/04/2010.

HABIB, A., CHUANG, J., “*Incentive mechanism for peer-to-peer media streaming*”. IWQOS 2004: 12th *International Workshop on Quality of Service*, 171-180, Montreal, Canadá, 2004.

HALES, D. E., PATARIN, S.. “*Computacional sociology for system in the wild: the case of BitTorrent”, IEEE Distributed Systems Online*, 2005

HEFEEDA, M., HABIB, A., BOTEV, B., XU, D., BHARGAVA, B.. 2003. “PROMISE: peer-to-peer media streaming using CollectCast”. In *Proceedings of the Eleventh ACM international Conference on Multimedia,* Berkeley, CA, USA, November 02 - 08, 2003. DOI= http://doi.acm.org/10.1145/ 957013.957022

HOONG, P. K., MATSUO, H.. 2008. “*Push-Pull Incentive-based P2P Live Media Streaming System*”. *WTOC* 7, 33-42, February, 2008.

IYER, S., ROWSTRON, A. e DRUSCHEL, P.. 2002. “Squirrel: a decentralized peer-to-peer web cache”. In *Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing,* Monterey, California, July 21 - 24, 2002. DOI= http://doi.acm.org/10.1145/571825.571861

JIANG, X., DONG, Y., XU, D., BHARGAVA, B.. 2003. “GnuStream: a P2P media streaming system prototype”. In *Proceedings of the 2003 international Conference on Multimedia and Expo - Volume 1,* July 06 - 09, 2003.

KON, F., CAMPBELL, R. H., TAN, S., ZHIGANG, M., ZHIGANG, C., e WONG, J.. 1998 “*A Component-Based Architecture for Scalable Distributed Multimedia”*. In *Proceedings of the 14th International Conference on Advanced Science and Technology (ICAST'98)*, Lucent Technologies, Naperville, USA. April, 1998.

KOSTIC, D., RODRÍGUEZ, A., ALBRECHT, J., VAHDAT, A.. 2003. “Bullet: high bandwidth data dissemination using an overlay mesh”. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles,* Bolton Landing, NY, USA, October 19 - 22, 2003. DOI= http://doi.acm.org/10.1145/945445.945473

KUBIATOWICZ, J., BINDEL, D., CHEN, Y., CZERWINSKI, S., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C. E., ZHAO, B.. 2000. “*OceanStore: an architecture for global-scale persistent storag*e”. ASPLOS, 190-201. November, 2000.

LOCHER, T., MEIER, R., SCHMID, S., WATTENHOFER, R.. 2007. “*Push-to-Pull Peer-to-Peer Live Streaming*”. In *Proc. 21-st DISC*, Sept. 24-26, Lemesou, Cyprus, LNCS 4731, Springer, 2007.

MARTI, S., GIULI, T. J., LAI, K., BAKER, M.. 2000. “Mitigating routing misbehavior in mobile ad hoc networks”. In *Proceedings of the 6th Annual international Conference on Mobile Computing and Networking,* Boston, Massachusetts, United States, August 06 - 11, 2000. DOI= http://doi.acm.org/10.1145/345910.345955

MORAES, I., CAMPISTA, M., BICUDO, M., CARDOSO, K., VASCONCELOS, S., DUARTE, O., de REZENDE, J.. 2003. “Desenvolvimento de um ambiente de testes com suporte à qualidade de serviço para transmissão de vídeo digital”. *IV WorkShop* da RNP, 2003.

MORAES, I., CAMPISTA, M., MOREIRA, M., RUBINSTEIN, M., COSTA, L., DUARTE, O.. 2008. “Distribuição de Vídeo sobre Redes Par-a-Par: Arquiteturas, Mecanismos e Desafios”. Livro texto dos minicursos, capítulo 3, SBRC, 2008.

MORAES, I.. “Distribuição de vídeo sobre redes par-a-par”. 2009. Tese de Doutorado do Programa de Pós-graduação em Engenharia Elétrica, COPPE, UFRJ. Dezembro, 2009.

MOURA, L.. 2009. “Uma política de escambo para gerenciamento de recursos em sistemas distribuídos par-a-par”. Exame de qualificação de Doutorado. COPPE, UFRJ. Julho, 2009.

Página do Napster, http://www.napster.com/. Página acessada em 29/04/2010.

NS-2. “Simulador NS2”, http://www.isi.edu/nsnam/ns/. Página acessada em 29/04/2010.

P4S. “P4S – Peer-to-Peer 4Streaming System”, http://www.dit.unitn.it/networking/P4S-main.html. Página acessada em 29/04/2010.

PAI, V., KUMAR, K., TAMILMANI, K., SAMBAMURTHY, V., MOHR, A.. 2005. “*Chainsaw: Eliminating Trees from Overlay Multicast*”. In *Proc. IPTPS*. February, 2005.

PEERSIM. “*PeerSim: A Peer-to-Peer Simulator”,* http://peersim.sourceforge.net/. Página acessada em 29/04/2010.

PENDARAKIS, D., SHI, S., VERMA, D., WALDVOGEL, M. 2001. “ALMI: an application level multicast infrastructure”. In *Proceedings of the 3rd Conference on USENIX Symposium on internet Technologies and Systems - Volume 3,* San Francisco, California, March 26 - 28, 2001.

PIANESE, F., KELLER, J., BIERSACK, W.. 2006. “*PULSE, a Flexible P2P Live Streaming System*”. Em *Proc. 9th IEEE Global Internet Symposium*. April, 2006.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R. e SCHENKER, S. 2001. “A scalable content-addressable network”. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications,* San Diego, California, United States, 2001. DOI= http://doi.acm.org/10.1145/383059.383072

ROWSTRON A. e DRUSCHEL, P.. 2001. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM international Conference on Distributed Systems Platforms Heidelberg,* November 12 - 16, 2001.

SANGHAVI, B.S., HAJEK, B., e MASSOULIE, L.. 2007 "*Gossiping with multiple messages"*. *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* December, 2007.

SAROIU, S., GUMMADI, K. P., GRIBBLE, S. D.. 2003. "Measuring and analyzing the characteristics of Napster and Gnutella hosts". *Multimedia Syst.* 9, 170-184, August, 2003. DOI= http://dx.doi.org/10.1007/s00530-003-0088-1

SHIRKY, C.. 2000 "*What's P2P and what's not"*. *Internet Publication,* November, 2000.

STOICA, I., MORRIS, R., LIBEN-NOWELL, D., KARGER, D. R., KAASHOEK, M. F., DABEK, F. e BALAKRISHNAN, H.. 2003. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.* 11, 17-32, February, 2003. DOI= http://dx.doi.org/10.1109/TNET.2002.808407

SUNG, Y., BISHOP, M., RAO, S.. 2006. "*Enabling Contribution Awareness in an Overlay Broascasting System*". In Proc. SIGCOMM. September, 2006.

TANENBAUM, A. S.. 2003. "Redes de Computadores". Editora Campus, 4ª edição, 2003.

TANENBAUM, A. S., STEEN, M. V.. 2007. "Sistemas Distribuídos – princípios e paradigmas". Editora Pearson Prentice Hall, 2ª edição, 2007.

TRAN, D. A., HUA, K. A., DO, T. T.. 2003 "*ZIGZAG: An efficient peer-to-peer scheme for media streaming"*. *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies,* 1283-1292, volume 2, 2003.

TRAN, D. A., HUA, K. A., DO, T. T.. 2004 "*A Peer-to-Peer Architecture for Media Streaming*". Em *IEEE JSAC: Special Issue on Advances in Overlay Networks,* 121-133, volume 22, N.1. January, 2004.

VENKATARAMAN, V., FRANCIS, P.. 2006. "*Chunkyspread: Multi-tree Unstructured Peer-to-Peer Multicast*". In *Proc. IPTPS*. February, 2006.

XINYAN, Z., JIANGCHUAN, L., BO, L., TAK-SHING, P.. 2005 "*CoolStreaming/DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming*". INFOCOM 2005. 24th *Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2102-2111, volume 3. March 2005.

WIKI01, "Arquitetura P2P", acessado em 03/04/2010. http://pt.wikipedia.org/wiki/P2P.

ZHAO, B.Y., HUANG, L., STRINBLING, J., RHEA, S.C., JOSEPH, A.D. E KUBIATOWICZ, J.D.. "*Tapestry: A resilient global-scale overlay for service deployment*". IEEE Journal on Selected Areas in Communications, 41-53, volume 22, n. 1. January, 2004.