DESENVOLVIMENTO E AVALIAÇÃO DE UM PROTOCOLO PEER-TO-PEER PARA APLICAÇÕES DA INTERNET ORIENTADAS A INTERESSE

Héberte Fernandes de Moraes

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Claudio Luis de Amorim

Rio de Janeiro
Junho de 2011

DESENVOLVIMENTO E AVALIAÇÃO DE UM PROTOCOLO PEER-TO-PEER PARA APLICAÇÕES DA INTERNET ORIENTADAS A INTERESSE

Héberte Fernandes de Moraes

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Claudio Luis de Amorim, Ph.D.

Prof. Luis Felipe Magalhães de Moraes, Ph.D.

Prof. Jauvane Cavalcante de Oliveira, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2011

Moraes, Héberte Fernandes de

Desenvolvimento e Avaliação de um protocolo Peer-to-Peer para aplicações da Internet orientadas a interesse/Héberte Fernandes de Moraes. – Rio de Janeiro: UFRJ/COPPE, 2011.

XIII, 100 p.: il.; 29, 7cm.

Orientador: Claudio Luis de Amorim

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 82 – 88.

1. Rede Overlay. 2. Rede móvel. 3. Redes Ad hoc. 4. Rede de sensores. 5. Rede de interesse. 6. Rede social. 7. Aplicaćão social. 8. Internet. 9. Network Address Translator. 10. Modelo de comunicaćão. I. Amorim, Claudio Luis de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha mãe pelo dom da vida e amparo ao longo desses anos. Ao meu pai pelo incentivo e apoio durante toda minha vida.

Agradecimentos

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte financeiro.

Agradeço aos meus professores de graduação na UFJF que tanto me ensinaram, principalmente os da área de arquitetura de computadores, sistemas operacionais e redes.

Agradeço ao meu orientador de graduação, Professor Marcelo Lobosco, pelo apoio incondicional a minha candidatura ao mestrado na COPPE.

Agradeço aos professores do Programa de Engenharia de Sistemas e Computação da COPPE pelas aulas e discussões interessantes e divertidas que tive nesses anos.

Agradeço aos Técnicos-Administrativos do Programa de Engenharia de Sistemas, que sempre auxiliaram e estiveram de prontidão para responder quaisquer duvidas, por mais bobas que fossem.

Agradeço aos colegas e colaboradores do Laboratório de Computação Paralela (LCP) que me ajudaram ao longo deste período de estudos e trabalho.

Um agradecimento especial para o Professor Claudio Luis de Amorim pela orientação durante esse período na COPPE. Graças ao seu seu suporte pude enriquecer meus conhecimentos.

Um agradecimento especial ao Renato Dutra, por tantos ensinamentos e discussões que em muito ajudaram a completar minha formação ao longo do mestrado.

Um agradecimento especial ao Rodrigo Granja, pelos ensinamentos passados, estando sempre de prontidão a ajudar os outros, mesmo quando estava em vias de defender seu mestrado.

Um agradecimento especial a meu pai que sempre cultivou em mim o prazer de estudar e aprender, mesmo que isso tenha me deixado com trauma de um dia vir a ser um empacotador de compras no Bramil.

Um agradecimento especial aos professores da banca examinadora, os professores Luis Felipe Magalhães de Moraes e Jauvane Cavalcante de Oliveira por disponibilizarem seu tempo e por contribuírem com este trabalho.

Um agradecimento a todos meus amigos e as pessoas que de alguma forma acabaram por ficar em “terceiro” plano durante o desenvolvimento deste trabalho.

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

DESENVOLVIMENTO E AVALIAÇÃO DE UM PROTOCOLO PEER-TO-PEER PARA APLICAÇÕES DA INTERNET ORIENTADAS A INTERESSE

Héberte Fernandes de Moraes

Junho/2011

Orientador: Claudio Luis de Amorim

Programa: Engenharia de Sistemas e Computação

Utilizamos um modelo de comunicação baseado em conteúdo para desenvolver uma rede puramente *Peer-to-Peer* (P2P) centrada nos interesses dos usuários e avaliar seu potencial como uma rede sobreposta eficiente na Internet para o desenvolvimento de novas aplicações *web*. Na rede proposta, cada nó define, de modo completamente distribuído, um prefixo ativo (PA) composto de campos de características e interesses, permitindo que os nós atuem colaborativamente na entrega de mensagens e formação de grupos de interesses. Para isto, um nó se comunica enviando mensagens cujo cabeçalho é o seu PA, e o protocolo de comunicação utiliza o PA de cada nó como filtro de decisão de encaminhamento e destino de cada mensagem recebida segundo o PA da mensagem. Nesta dissertação, um protocolo de comunicação foi desenvolvido para construir e avaliar por simulação uma tal rede que denominamos Rede Endereçada por Interesses (REPI) para a qual foram medidas as taxas de entrega, latência e sobrecarga de mensagens em REPIs com mais de 10 mil nós. Além disso, utilizamos o protocolo da REPI para mostrar sua eficácia no desenvolvimento de um protótipo de aplicação de rede social onde usuários são diretamente interconectados através de interesses comuns sem terem se registrado previamente em qualquer portal *web*.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DEVELOPMENT AND EVALUATION OF A PROTOCOL TO PEER-TO-PEER INTERNET ORIENTED APPLICATIONS OF INTEREST

Héberte Fernandes de Moraes

June/2011

Advisor: Claudio Luis de Amorim

Department: Systems Engineering and Computer Science

We used a content-based communication model to develop a pure P2P network centered on the users' interests and evaluate its potentials as an efficient overlay network to the internet for the development of new web applications. In the proposed network, each node defines in a completely distributed way, an active prefix (AP) that consists of characteristic and interest fields to allow nodes to work collaboratively. To this end, a node communicates by sending messages using its AP as message's header, and a communication protocol that uses each node's AP as a decision filter to forwarding and delivery each received message according to the message's PA. In this work, a network protocol has been developed to build and evaluate therefor simulation which we called the Active Prefix Networks (APNs) to which we measured the delivery rate, latency, and message overheads in APNs with more than 10 thousand nodes. In addition, we used the APN protocol to show its efficacy to the development of a social-network application prototype for APNs where the users are directly interconnected by common interests without being previously registered on any web site.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A Internet começou a crescer com o surgimento do *World Wide Web* que permitiu a publicação e busca de informação através de páginas *Web*. Com o tempo, novas tecnologias de comunicação e computação surgiram proporcionando a criação de novas aplicações que facilitassem a interconexão das pessoas e organizações, o que a tornou cada vez mais atrativa para a sociedade em geral. Em particular, apareceram as aplicações sociais que exploram a facilidade de estabelecer a comunicação entre os usuários através de um ou vários tipos de relacionamentos [2]. Porém, a arquitetura da Internet não foi preparada para esta nova onda de aplicações, ou seja, o conjunto de protocolos atualmente utilizados (também chamada de pilha de protocolos TCP/IP [3, 4]) não tem o suporte ideal para este novo leque de aplicações cada vez mais presente na Internet.

As aplicações atuais tem sido desenvolvidas sobre redes sobrepostas ou *middleboxes* [5–8] que são formas de abstrair a atual composição da Internet com o intuito de permitir o funcionamento das aplicações no contexto de comunicação em que vivemos.

Nos últimos anos tem sido crescente o uso destas aplicações sociais. De acordo com os relatórios realizados pelo site Alexa [9], os sites mais acessados em 2011 são de alguma maneira, para busca de informação (ex.: Google[10], Yahoo![11]) ou aplicações sociais (ex.: Youtube[12], Facebook[13], LinkedIn[14]). Estas aplicações têm o foco em encontrar ou disponibilizar informações para os usuários, ou seja, o foco atual da Internet tem sido o usuário.

No inicio, quando surgiram as primeiras pesquisas sobre tecnologias de redes, o foco das aplicações era o compartilhamento de recursos [3, 4, 15], onde existiam os supercomputadores com grandes capacidades de processamento (para aquela época) que proviam serviços. O objetivo era permitir aos usuários o acesso, através da rede, a estes serviços. Sendo assim, o modelo de comunicação inicial da Internet (e ainda é nos dias atuais) é resultado da comunicação entre duas máquinas [7, 16]: uma provendo serviços e outra querendo utilizá-los. Ainda hoje o compartilhamento de

recursos é importante em sistemas cuja computação é distribuída.

Assim surgiram os primeiros (e atuais) protocolos da Internet [17], onde esse contexto de clientes em busca de serviços providos por servidores está presente na arquitetura e bem visível na pilha de protocolos TCP/IP, tal como o IP (*Internet Protocol*) que utiliza em seus pacotes o endereço do requisitante e do provedor dos serviços (endereços de origem e destino, respectivamente).

Este modelo de comunicação usa o princípio de *onde* está o serviço, mas as pessoas avaliam a Internet por *qual* conteúdo ela tem [7, 16, 18]. Para isso é necessário mapear a informação *que* o usuário deseja para *onde* na rede ela está. Por isso as aplicações atuais necessitam de adaptações (redes sobrepostas e *middleboxes*) para poder suprir esse desejo dos usuários, já que a rede não é capaz de fazer isto.

Pelas dificuldades impostas pela atual arquitetura da Internet, novas pesquisas [16, 18–20] tem sido realizadas com o objetivo de desenvolver uma rede que dê o suporte necessário para estas aplicações. Essa nova arquitetura deve manter os princípios que a levaram ao sucesso atual, tais como a facilidade de implantação e a adaptabilidade de seus protocolos, mas com conceitos novos que facilitem o desenvolvimento das aplicações.

Neste trabalho, apresentamos um protótipo de aplicação social que faz uso de um novo modelo de comunicação. Tal modelo utiliza o conceito de prefixo ativo formado por características e interesses do usuário como meio de interconectá-los. A troca de mensagens é realizada através dos interesses definidos pelos usuários, utilizando um novo protocolo (REPI) responsável por formar uma Rede Endereçada Por Interesses.

Para este novo contexto da Internet, apresentamos a REPI como uma alternativa de protocolo para realizar a entrega de mensagens, busca de informação, publicação de dados por meio de interesses, mudando o foco da comunicação do dispositivo para o usuário.

Através do uso de prefixos ativos, a REPI faz com que os dispositivos presentes na rede colaborem encaminhando as mensagens entregando-as aos grupos de usuários interessados. O prefixo ativo é composto de campos responsáveis pela decisão de encaminhamento das mensagens e campos com os interesses que definem a mensagem.

1.1 Motivação

Dutra [21, 22] propôs um novo protocolo de comunicação endereçado por interesse nomeado de REPI (Rede Endereçada Por Interesse). Baseadas nesse novo protocolo as aplicações do contexto atual da Internet podem fazer uso do endereçamento por interesse para estabelecer a comunicação entre os usuários em nível de rede. Assim sendo, a rede daria às aplicações o suporte necessário para permitir a interação entre

os usuários e as informações de seus interesses sem a necessidade de adaptações como existe atualmente.

Através do mecanismo de troca de mensagens da REPI, as atuais e novas aplicações podem ser desenvolvidas sem a necessidade da utilização de adaptações sobre a atual arquitetura da Internet. Por meio dos interesses é possível realizar a comunicação entre as aplicações com o suporte da rede sem a necessidade de abstrair questões físicas de sua estrutura.

Para a avaliação desse novo modelo de comunicação foi necessário abstrair as questões físicas atuais da Internet, com isso desenvolvemos uma rede sobreposta com o objetivo de permitir a comunicação direta entre os dispositivos participantes da rede. Baseado no modelo P2P, foi desenvolvido um algoritmo para a formação de uma vizinhança em cada dispositivo, dessa forma as mensagens recebidas por cada nó são encaminhadas de acordo com o protocolo REPI para seus vizinhos contando com a colaboração de todos da rede.

1.2 Contribuições

A seguir são apresentadas as principais contribuições deste trabalho.

Algoritmo REPI-Internet

Elaboração de um algoritmo para formação de uma rede sobreposta P2P, abstraindo as questões físicas atuais da Internet. Tal algoritmo estabelece e mantém as conexões entre os dispositivos presentes na rede. Sobre esta rede formada, o protocolo REPI foi aplicado para realizar a entrega das mensagens baseado nos interesses dos usuários.

Aplicação Mensageira

Desenvolvimento de uma aplicação social que faz o uso do modelo de comunicação estudado e utiliza o algoritmo de formação de uma rede sobreposta P2P. Esta aplicação está disponível para a comunidade em [23]. A aplicação é um primeiro protótipo que utiliza o protocolo REPI na Internet.

Alteração Simulador

Implementação do algoritmo de formação da rede sobreposta P2P e do modelo estudado no *Network Simulator* (NS-3.8) [24]. A implementação da REPI-Internet no simulador facilita o estudo deste novo modelo de comunicação por outros grupos de pesquisa. Nem todos tem uma infraestrutura boa o suficiente para realizar experimentos de larga escala no nível da Internet por isso a implementação do protocolo

sobre a Internet no simulador é um passo inicial para incentivar a pesquisa nesta área.

Avaliação

A avaliação da viabilidade de utilização desse novo modelo de comunicação sobre a Internet, além da proposta de utilização do modelo em aplicações sociais ou em redes P2P. Com os resultados obtidos neste trabalho é possível tomar decisões de aplicar este protocolo em outros contextos na Internet, além de ser possível identificar os pontos falhos e propor formas de melhorar a adaptação do protocolo na rede infra-estruturada.

O restante do trabalho está organizado da seguinte forma: a seguir, o Capítulo 2 apresenta alguns dos trabalhos relacionados referentes a aplicações sociais e a arquitetura P2P. No Capítulo 3, descreve-se o modelo de comunicação orientado ao interesse que é à base do protocolo REPI, ilustrando o seu funcionamento, descrevendo seus conceitos e pontos principais e por fim apresentando a primeira implementação do protocolo em redes sem fio. O Capítulo 4 apresenta a implementação do protótipo da aplicação REPI sobre Internet descrevendo as decisões de projeto, abordando os problemas encontrados e as soluções adotadas. Ainda no Capítulo 4 apresentamos detalhes sobre a implementação da aplicação REPI-Internet no simulador NS-3.8. No Capítulo 5 estão expostos os cenários utilizados para a realização da avaliação e os resultados obtidos durante estas avaliações. Ainda neste capítulo é feita uma discussão sobre estes resultados. No Capítulo 6 concluímos expondo as vantagens de utilização do modelo sobre a Internet e apresentamos as possibilidades de trabalhos futuros.

Capítulo 2

Trabalhos Relacionados

Com o objetivo de entender e levantar os diversos problemas da arquitetura atual da Internet, apresentamos algumas das diversas iniciativas e sistemas que visam o desenvolvimento de um novo modelo de comunicação para a Internet. Esta dissertação apresenta os seguintes aspectos que podem ser comparados a outros trabalhos:

- A apresentação de um novo modelo original de troca de mensagens baseado em interesses;
- O desenvolvimento de uma rede P2P sobreposta para abstrair questões físicas da atual arquitetura da Internet;

Baseados nestes aspectos apresentamos dois conjuntos de trabalhos relacionados. O primeiro é baseado em novos modelos de comunicação focados no conteúdo que representa o desejo do usuário. O segundo conjunto, são aplicações que funcionam na arquitetura atual de rede utilizando uma rede sobreposta baseado no modelo P2P para abstrair as questões físicas da rede.

2.1 Redes baseadas em conteúdo

Nesta seção, apresentaremos um trabalho cujo foco é o desenvolvimento de um novo modelo de comunicação que se adeque ao atual contexto da Internet, onde o conteúdo é central. Este contexto se preocupa apenas em qual conteúdo o usuário quer, ao invés de onde na rede ele está. O trabalho que aborda este novo contexto da Internet em nível de rede é o *Content-Based Network* (CCN) [16] que usa uma abordagem de roteamento baseado em conteúdo.

O CCN tem foco na disseminação do conteúdo ao invés da manutenção das conexões de rede [18], sua essência consiste em mapear nomes aos dados e permitir aos usuários recuperar estes dados através dos nomes que o definem. O roteamento é projetado para descobrir o dado através do casamento de nomes realizado no interior

da rede. Os dados são armazenados nos dispositivos que compõe a rede permitindo que as requisições a esses dados possam ser atendidas pelo roteador mais próximo.

O projeto *Named Data Networking*(NDN) [25] é uma implementação deste modelo de rede baseada em conteúdo. Esta implementação mapeia os dados da rede através de nomes, tais nomes são estruturados hierarquicamente pois é útil para as aplicações existir relações entre dados relacionados.

A NDN funciona seguindo o paradigma Publicador/Assinante [26], pois os roteadores ao receberem pacotes de interesses, armazenam a interface de recebimento e o nome descrito no pacote em uma Tabela de Interesses Pendentes (TIP), ou seja, assina o interesse naquele conteúdo. Dessa forma quando um pacote de dados referente àquele nome for publicado na rede e passar pelo roteador assinante, o pacote será repassado para a interface descrita na TIP.

Este modelo de comunicação se diferencia do modelo utilizado neste trabalho na questão de manipulação dos pacotes na rede, na REPI o encaminhamento é realizado probabilisticamente de acordo com os campos de característica do prefixo ativo, mais detalhes na seção 3. Outra diferença é quanto a utilização do nome, no caso do CCN cada conteúdo deve ser nomeado enquanto que em nosso modelo a utilização dos nomes é pela aplicação ou pelo usuários de acordo com seus interesses. Além do fato de nosso modelo não seguir o paradigma Publicador/Assinante e nem realizar o cacheamento dos dados na rede.

2.2 Aplicações P2P

Um grupo de aplicações para Internet tem o foco no compartilhamento de recursos, onde os usuários podem diretamente compartilhar arquivos e outras informações. Apesar de a maior parte das aplicações P2P conhecidas serem para compartilhamento de arquivos, o modelo P2P tenta prover uma longa lista de funcionalidades [27], algumas delas são: armazenamento redundante, busca eficiente de dados, confiabilidade, anonimato, entre outras, mas nem sempre as aplicações P2P fazem uso de todas essas funcionalidades.

Existem duas classes de redes P2P: estruturadas e não estruturadas. As redes estruturadas são construídas seguindo um processo determinístico, desta forma existe um conhecimento global da topologia da rede. O procedimento de formação da rede mais utilizada neste caso é a organização através de tabelas *hash* distribuídas (DHTs). Nestes sistemas baseados em DHTs cada objeto adicionado na rede recebe uma chave aleatória. Os nós da rede também recebem um identificador e um conjunto de chaves do qual é responsável. Existem várias implementações e protocolos baseados em DHTs, alguns deles são o Chord [28], Pastry [29] e D1HT [30].

As redes não estruturadas normalmente são baseadas em algoritmos aleatórios

para seleção de vizinhos na rede sobreposta. Cada dispositivo mantém uma lista com seus vizinhos descobertos de maneira aleatória. Quando há a necessidade de localizar algum dado específico, a busca é normalmente realizada inundando a rede com mensagens de consulta. Pode ocorrer que as consultas não sejam respondidas devido ao dado buscado estar em dispositivo muito distante do requisitante, pois existem mecanismos limitadores que impedem que uma mensagem se propague indefinidamente consumindo os recursos da rede. Nenhum dispositivo na rede tem a visão global da topologia tendo apenas conhecimento parcial da rede, geralmente se comunicando com uma quantidade limitada de vizinhos. Várias aplicações são P2P não estruturados como exemplo temos o BitTorrent [31] e Gnutella [32].

O BitTorrent [31, 33] é um protocolo para compartilhamento de arquivos utilizando o modelo P2P, mas faz uso de uma unidade centralizada responsável por gerenciar a comunicação entre os usuários. O arquivo distribuído pelo BitTorrent é particionado em pedaços, tais pedaços são tratados como a unidade básica de compartilhamento no sistema. O protocolo tenta ser justo quanto à colaboração dos integrantes da rede; usuários que compartilham os pedaços que já baixaram tem prioridade para baixar os pedaços que ainda faltam. As unidades centralizadoras conhecidas como *trackers* são responsáveis por manter informações sobre os arquivos compartilhados na rede e de conectar os usuários para realizar a troca dos pedaços dos arquivos.

Para que um usuário possa iniciar o compartilhamento de um arquivo é necessário encontrar um *torrent* que contenha as informações sobre os pedaços do arquivo compartilhado. Este *torrent* normalmente é distribuído através de servidores *web* onde é possível realizar a busca de maneira eficiente. Neste arquivo estão informações como: nome, tamanho, quantidade de pedaços, informações de *hashing* e o endereço do *tracker*. Para cada pedaço do arquivo está associado um *hash* que é usado para verificar a integridade dos pedaços copiados de outros usuários.

Já o Gnutella [32] é um protocolo para compartilhamento de arquivos descentralizado. Neste protocolo não existe nenhuma unidade para gerenciar a topologia da rede e nem o processo de busca de arquivos. Não existe nenhuma regra que defina a forma da topologia da rede, quando um usuário se conecta à rede ele precisa descobrir pelo menos outro já presente. Vários métodos podem ser utilizados para se interligar a rede Gnutella, pode ser através de uma lista pré-existente de endereços possíveis ou através de sites. Ao encontrar o primeiro usuário, este lhe envia sua lista de vizinhos aos quais se tentará estabelecer a conexão. Este processo de recebimento da lista de vizinhos ocorre até que seja atingida uma cota de vizinhos padrão ou definida pelo usuário.

Para se localizar um item na rede, o usuário consulta seus vizinhos e esta consulta é então enviada para os vizinhos dos seus vizinhos inundando a rede. Este processo

de inundação é realizado até um determinado raio de busca. Este método não é escalável e pode ser que não encontre o objeto buscado pelo fato da consulta não atingir toda a rede. Quando a consulta encontra um resultado, o arquivo é copiado pelo usuário requisitante diretamente do dispositivo que contém o arquivo.

Nosso trabalho não altera a arquitetura da Internet, apesar de o modelo de troca de mensagens apresentado ser um novo modelo de rede. Em nosso caso foi desenvolvida, sobre a atual arquitetura da Internet, uma rede sobreposta baseado no modelo P2P. Avaliamos o algoritmo de formação desta rede sobreposta e o novo modelo de troca de mensagens abordado.

O foco deste trabalho é avaliar o comportamento da rede formada ao se utilizar características e interesses como meio de encaminhamento e entrega de mensagens. Este modelo original se difere dos demais ao utilizar interesses como meio de troca de mensagens. Nos outros trabalhos apresentados neste capítulo, a nomeação do conteúdo trocado e o armazedamento do mesmo na rede são as suas principais características, enquanto que em nossa abordagem utilizamos características para o encaminhamento e interesses do usuário ou aplicação como identificação das mensagens.

Capítulo 3

Modelo de Comunicação Distribuída Orientada ao Interesse

Atualmente as informações que trafegam na rede têm origem e destino bem definidos [34]. No Modelo de comunicação distribuída Orientada ao Interesse (MOI) um usuário envia uma mensagem com um determinado interesse para um determinado grupo de usuários que compartilham deste mesmo interesse. O MOI não necessita que seja definido endereço único para cada integrante da rede, esse fato reduz bastante à complexidade de formação e manutenção da rede. Não existe o conceito de roteamento, pois os pacotes transmitidos, a princípio, não tem um destino único pré-determinado. As mensagens são encaminhas probabilisticamente utilizando um conjunto de termos, sendo tais termos os interesses definidos pelo usuário (ou pela aplicação que faz uso deste modelo) de forma que os dispositivos representem os usuários na rede. O uso dessa forma de envio reduz a complexidade nas decisões de encaminhamento realizadas pelos dispositivos na rede, pois não é necessário decidir qual vizinho deve receber exclusivamente essa mensagem. A decisão de encaminhamento é feita individualmente por cada nó de acordo com os critérios definidos por uma função de casamento.

Este modelo foi desenvolvido com o foco em rede móvel, pois nesse tipo de rede a entrada e a saída de dispositivos acontecem com muita frequência; um dispositivo ao se movimentar pode entrar ou sair do campo de transmissão de seus vizinhos, além do fato de que sua energia poder acabar ou até mesmo ser desligado pelo usuário. Esta instabilidade na rede não tem muito impacto no MOI, pois o modelo não necessita de conhecimento da estrutura da rede para o envio das mensagens, como ocorre em alguns protocolos de roteamento [35].

O modelo não provê a garantia de entrega das mensagens, o motivo é o fato de que não há controle para o envio das mensagens em relação ao direcionamento destas. Cada dispositivo processa as mensagens recebidas sem o intermédio de seus vizinhos e decide se deve ou não encaminhá-las, decisão realizada localmente. Neste modelo,

os dispositivos considerados destinos de uma mensagem são aqueles cujos interesses são coincidentes de acordo com algum critério definido localmente no dispositivo, como essa informação é local outro membro da rede não tem conhecimento se existe algum destino para suas mensagens.

A colaboração é um fator importante, pois cada dispositivo tem seu papel na disseminação das mensagens. Cada nó transmite suas mensagens a seus vizinhos em seu alcance de radio frequência; e esses vizinhos as retransmitem para seus vizinhos e assim sucessivamente. A retransmissão dessas mensagens segue alguns critérios definidos localmente em cada nó, esses critérios são fatores que impedem que essas mensagens inundem a rede. Se alguns destes nós não retransmitirem as mensagens, vários outros dispositivos poderão ser prejudicados por não as receberem, pois estes dispositivos prejudicados poderiam ter interesse nessas mensagens. O impacto de um nó não colaborador pode ser um fator crítico como também pode ser imperceptível para a rede. Esse é um dos fatores importantes a ser estudado sobre o modelo em questão.

Baseado neste modelo de comunicação, [21, 22] propõe o desenvolvimento da Rede Endereçada Por Interesse (REPI). Na seção seguinte é discutido em detalhes o funcionamento da REPI e seus componentes principais.

3.1 Rede Endereçada Por Interesse

A rede de endereçamento por interesses (REPI) é uma implementação do MOI proposto em [21, 22]. Todas as características descritas para o modelo são aplicadas na REPI, sendo que otimizações foram desenvolvidas com o intuito de melhorar o desempenho quando implementado em um ambiente real.

3.1.1 Composição do Prefixo Ativo

Cada dispositivo contém um Prefixo Ativo (PA) composto por informações definidas pelo usuário (ou aplicação). Um PA, Figura 3.1, é subdivido em dois grupos de campos: o primeiro grupo contém dados que seguem uma determinada distribuição multivariada, em nosso estudo usamos uma distribuição gaussiana, chamado de campos de características (C); o outro grupo de campos contém informações que descrevem os interesses do usuário, sendo o interesse qualquer sequência de caracteres, chamado de campos de interesses (I).

C é composto por campos que seguem uma determinada distribuição probabilística. Nesta implementação utilizamos as características comuns e seguem uma distribuição gaussiana. Esses campos são utilizados para realizar o encaminhamento probabilístico de mensagens, ou seja, a ocorrência de valores iguais implica no en-

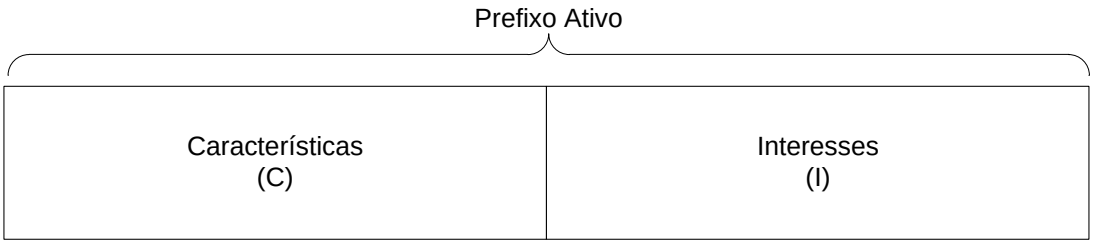


Figura 3.1: Prefixo Ativo (PA) REPI

caminhamento da mensagem na rede. Esses campos são parte da proposta original da REPI descrita em [22].

I é composto pelos interesses do usuário. Esse conjunto é responsável pelo endereçamento das mensagens para grupos composto de usuários de mesmos interesses. I por si só não tem a capacidade de atingir vários dispositivos na rede, pois um interesse muito popular na rede provoca a inundação de mensagens na rede enquanto que um interesse menos popular pode não atingir nenhum interessado. Para resolver este problema utilizamos C para realizar o encaminhamento contando com a colaboração dos dispositivos na rede.

No cabeçalho das mensagens está presente o PA, composto pelos campos de C e I do transmissor. Este PA presente no cabeçalho da mensagem é avaliado por todos os dispositivos que a receber, sendo usado pela função de casamento para comparar com o PA do dispositivo.

A propagação das mensagens na rede ocorre através do encaminhamento realizado pelos dispositivos da rede. A decisão de encaminhar ou aceitar uma mensagem é realizado individualmente através de uma função de casamento. Essa função de casamento avalia o PA de uma mensagem comparando, campo a campo, com o PA do dispositivo. Essa política de encaminhamento de mensagens utiliza todo o PA em suas decisões. Para o encaminhamento utiliza-se os campos de C, pois de acordo com a distribuição utilizada existe a chance de ocorrer o casamento dessas informações, assim sendo vários caminhos entre os dispositivos passam a existir. A aceitação da mensagem, ou seja, o repasse da mensagem para a aplicação é feita de acordo com I, pois os interesses são os responsáveis pela formação de grupos de interesse estabelecendo canais de comunicação entre as aplicações na rede.

3.1.2 Funcionalidades da REPI

Existem três funcionalidades de comunicação na REPI, são elas:

- Encaminhamento: O encaminhamento das mensagens é realizado por todos os nós integrantes da rede. Para que o encaminhamento ocorra, a REPI faz o uso do filtro de casamento. Como descrito anteriormente, o filtro de casamento

compara, campo a campo, o PA da mensagem com o do dispositivo. Esse filtro de casamento realiza a filtragem das mensagens que devem ser aceitas, sendo repassadas para aplicação; e as que devem ser encaminhadas, sendo reenviadas para rede transmitindo-as para seus vizinhos, caso contrário, essas mensagens são descartadas localmente;

- Formação de grupos: há formação de dois tipos de grupos na rede: grupo de encaminhamento, composto pelos dispositivos cujo C contém algum campo em comum que favorece o encaminhamento de mensagens; grupo de interesse, composto pelos dispositivos que compartilham de pelo menos um interesse em comum. Um usuário pertence a um determinado grupo de acordo com a decisão comparativa usada na função de casamento. A rede só funciona quando existem grupos, pois os integrantes desses grupos serão os responsáveis pelo encaminhamento das mensagens, nesse momento a colaboração entre os dispositivos é um fator importante para a rede. Dispositivos em um mesmo grupo de encaminhamento são chamados de colaboradores, pois estes dispositivos são os responsáveis pela disseminação das mensagens pela rede fazendo com que estas alcancem dispositivos pertencentes ao mesmo grupo de interesse;

- Endereçamento: existem duas formas de endereçamento usando este modelo, são eles: grupo e convencional. O endereçamento de grupo é realizado através de I, onde todos os interessados aceitam as mensagens. O endereçamento convencional é realizado utilizando todo o PA como endereço. O PA do nó destinatário é colocado no cabeçalho da mensagem, dessa forma apenas o nó com o PA mapeado no cabeçalho aceitará a mensagem. A proposta original da REPI, as mensagens são compostas de dois PAs: o PA do nó origem utilizado para o encaminhamento das mensagens; e o PA do nó destinatário, sendo que para o caso de endereçamento de grupos, os campos de características são preenchidos com zeros e apenas os interesses são avaliados. Neste trabalho utilizamos apenas um PA no cabeçalho das mensagens, pois avaliamos apenas o endereçamento de grupos.

A decisão de encaminhamento das mensagens é feita pelo filtro de casamento. Esse filtro tem uma versão proposta em [22] que faz uso dos campos do PA para filtrar as mensagens que devem ser propagadas, as que devem ser aceitas ou por fim descartadas. Três tipos de função de casamento foram propostos, são eles:

- Filtro de casamento total: compara campo a campo do PA da mensagem com o do nó avaliador e verificar se **TODOS** esses campos são iguais;

- Filtro de casamento parcial: compara campo a campo do PA da mensagem com o do nó avaliador e verificar se **PELO MENOS UM** desses campos é

igual;

- Sem filtro: não faz verificação, ou seja, sempre encaminha a mensagem.

De acordo com o tipo de filtro usado pelo dispositivo, a mensagem avaliada é encaminhada ou não. O filtro usado em nossos estudos é o parcial, mas nada impede que algum dos outros tipos de filtro sejam utilizados ou mesmo criados. Pode-se utilizar, também uma forma híbrida, onde em determinado conjunto de campos do PA se utilize o filtro total e em outro conjunto de campos utilize o filtro parcial, essa é uma questão a ser avaliada e não está no escopo deste trabalho.

A função de casamento tem um papel importante na rede, pois é usada principalmente para limitar a propagação das mensagens. Outras técnicas podem ser usadas em conjunto com o filtro de casamento. Como exemplo, pode ser adicionado à mensagem um campo que limita a quantidade de saltos ou nós visitados (HTL - *hop to live*). Também se pode usar um campo com o tempo de vida da mensagem (TTL - *time to live*), técnicas usadas nos modelos de comunicação convencionais. Ao se utilizar uma dessas técnicas se reduz o raio de propagação da mensagem e impede que ela trafegue indefinidamente na rede consumindo seus recursos.

Outra técnica é o uso de uma tabela para armazenar as últimas mensagens que já passaram pelo dispositivo. Assim é possível criar uma memória local onde cada dispositivo possa verificar se uma determinada mensagem já foi encaminhada por ele. Com essa memória é possível diminuir a quantidade de mensagens replicadas na rede, pois cada mensagem será encaminhada apenas uma vez. Esta tabela é limitada com valor pré-determinado de entradas (100 mensagens em nossa implementação) e apenas as últimas mensagens recebidas estão contidas nela.

Em nossa avaliação, utilizamos duas destas técnicas: o HTL para limitar o raio de propagação e uma tabela com as últimas mensagens recebidas para funcionar como uma memória evitando que o nó encaminhe uma mensagem já transmitida anteriormente.

Na seção seguinte será apresentado um exemplo ilustrativo de funcionamento da REPI.

3.2 Exemplo de REPI

Nesta seção, será mostrado em detalhes o funcionamento de uma REPI. Supondo uma rede composta por quatro dispositivos dispostos como na Figura 3.2, onde o raio de transmissão de cada dispositivo não é capaz de atingir todos os outros presentes na rede.

A Tabela 3.1 apresenta o PA de cada elemento da rede. Supondo que utilizamos dois campos de características com cinco possibilidades cada. Pode-se observar que

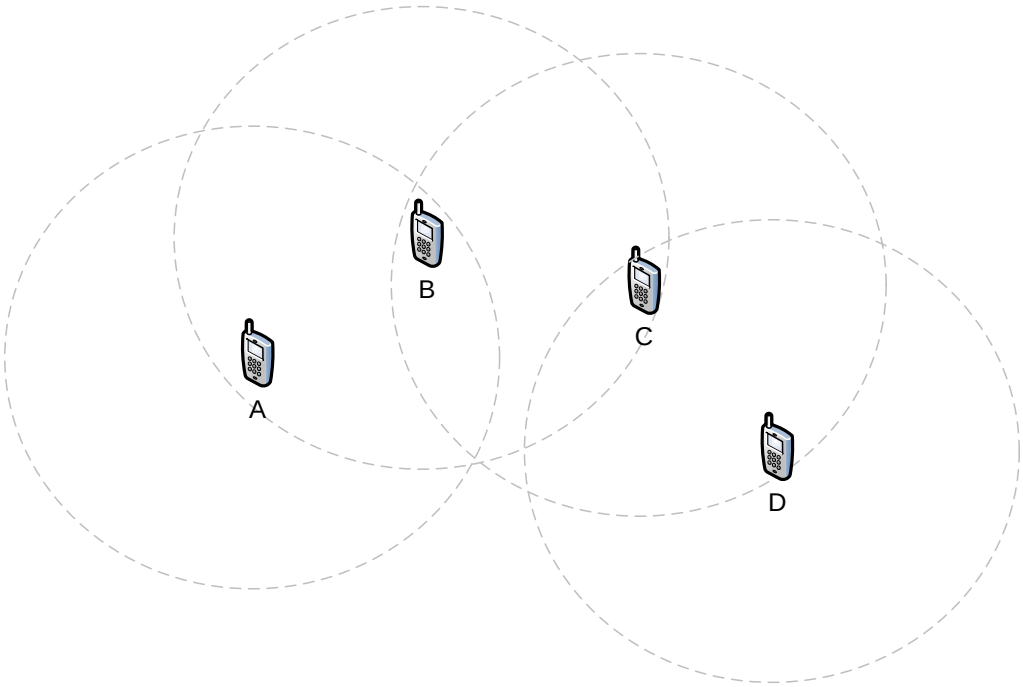


Figura 3.2: Rede formada por quatro dispositivos

o dispositivo D tem o mesmo interesse que o dispositivo A, sendo assim as mensagens enviadas por A serão aceitas por D, mas A não tem capacidade de enviar mensagem diretamente para D, por esse fato há a necessidade de que os outros membros da rede colaborem para que as mensagens oriundas de A cheguem até D.

Tabela 3.1: PAs participantes da REPI

Dispositivo	**PA**		
	Características		**Interesse**
	$\mathbf{C_1}$	$\mathbf{C_2}$	
A	①	5	Futebol
B	3	5	Carona
C	①	2	Almoço
D	3	4	Futebol

Supondo que nesta rede estamos utilizando o filtro de casamento parcial, ou seja, um dispositivo irá encaminhar a mensagem se pelo menos uma das características do dispositivo for igual a algum presente no PA da mensagem. Neste exemplo, o dispositivo A envia uma mensagem contendo em seu cabeçalho o seu PA. A mensagem enviada será recebida pelo dispositivo B por estar no raio de transmissão de A. B ao receber a mensagem a avalia através do filtro de casamento e verifica que seu PA tem a mesma característica "C_2", com isso, B decide encaminha-la enviando-a para C. Ao receber a mensagem, C também avalia seu PA é constata que existe

uma característica em comum, a "C_1", logo a encaminha para o dispositivo D. D verifica em seu PA que ela é de seu interesse, logo aceita a mensagem repassando-a para a aplicação e/ou mostrando-a para o usuário, o filtro de casamento de D também constata que não há características em comum, logo o dispositivo D não a encaminha.

Como é possível observar os dispositivos B e C não tinham o interesse na mensagem, mas a encaminharam colaborando com a rede. Estes dispositivos são chamados de Colaboradores na REPI. Em uma REPI, ter colaboração é importante para a disseminação das mensagens, mas colaboração demais pode ser muito custosa para a rede quando o interesse não é muito popular, logo deve existir um *tradeoff* entre encaminhar as mensagens e a popularidade de um interesse. Esta questão é influenciada pela quantidade de campos e possibilidade existentes nas características. Avaliar qual a melhor configuração de campos e possibilidades no PA foge do escopo deste trabalho.

Uma gama de aplicações pode utilizar a REPI como protocolo de comunicação podendo explorar algumas características do contexto ao qual aquela aplicação está inserida. A aplicação mais simples, cujo uso da REPI é evidente, é o caso de uma aplicação mensageira onde os usuários se comuniquem por interesse sem ter o conhecimento prévio dos outros integrantes da rede. Fazendo uma analogia com o rádio, nessa aplicação cada usuário pode sintonizar os canais de interesse aos quais deseja ouvir, podendo receber mensagens de um ou vários interesses distintos concorrentemente. Diferentemente do rádio, com a REPI há a capacidade de sintonizar mais de um canal simultaneamente.

3.3 REPI-A

Em [36] foi desenvolvido um protótipo da REPI utilizando sensores Tmote [37] que utilizam o protocolo de comunicação 802.15.4 (ZigBee) [38–40]. A REPI-A foi desenvolvida com objetivo de avaliar e estudar o comportamento do modelo de comunicação endereçada por interesse. Para esta avaliação foi desenvolvida uma aplicação mensageira que realiza a troca de mensagens baseado nos interesses do usuário. Nesta implementação descrita em [36], foi necessário o desenvolvimento de um *middleware* responsável pelo controle do *hardware* Tmote realizando a tarefa de troca de mensagens entre a aplicação mensageira e o transceptor de rádio frequência do dispositivo. A pilha de protocolos implementada para a REPI-A está ilustrada na tabela [1] 3.2.

Integrado à aplicação mensageira está presente o protocolo REPI-A, cuja função é avaliar a política de encaminhamento e o controle de duplicação de mensagens

[1] Tabela baseada na figura apresentada por [36]

Tabela 3.2: Pilha de protocolos utilizados para a formação da REPI-A

Aplicação Mensageira	Camada de Aplicação
Protocolo REPI-A	Camada de Rede
Middleware Tmote	Camada de enlace e física
802.15.4 (ZigBee)	

de acordo com as configurações possíveis. De acordo com o autor, essa escolha de projeto foi influenciada pelas várias limitações de recursos disponíveis no dispositivo utilizado. Logo, o *middleware* implementado no dispositivo realiza apenas a tarefa de repassar as mensagens recebidas no dispositivo para a aplicação e vice-versa.

Ao receber uma mensagem, o protocolo REPI-A decide se a encaminha de acordo com a politica utilizada, nesta implementação pode ser escolhido tanto o filtro parcial quanto o total. O filtro de casamento também avalia se a mensagem deve ser mostrada na aplicação mensageira de acordo com o interesse contido nela. Caso o usuário tenha o mesmo interesse, tal mensagem é exibida na interface da aplicação. Existe também uma memória para controlar a propagação das mensagens, essa memória impede que um mesmo dispositivo encaminhe a mesma mensagem mais de uma vez. Todo esse processo é realizado pelo protocolo REPI-A.

A rede formada pelo protocolo REPI-A é distribuída, ou seja, não conta com nenhuma unidade central controlando-a. As mensagens são originadas e encaminhadas pelos dispositivos presentes na rede. Essa rede formada pela REPI-A é semelhante a uma rede *peer to peer*, onde cada membro da rede é autônomo suficiente para decidir quanto ao encaminhamento das mensagens.

As redes sem fio são semelhantes às redes *peer to peer* (P2P), pois ambas são distribuídas onde cada dispositivo tem um mesmo papel colaborando para o funcionamento global da rede. A REPI foi desenvolvida para ser distribuída e necessita da colaboração de todos os dispositivos presentes. Baseado nestas afirmações conclui-se que a REPI pode ser adaptada para funcionar sobre a Internet utilizando o modelo P2P. No próximo capítulo é apresentada uma aplicação mensageira desenvolvida utilizando o modelo P2P que implementa a REPI em nível de aplicação.

Capítulo 4

REPI Internet

Tomando como base a REPI-A desenvolvida para redes sem fio, nesta dissertação iremos implementar a REPI sobre a Internet. Os mesmos conceitos do protocolo REPI são utilizados tanto na REPI-A como na REPI-Internet. Uma aplicação mensageira foi desenvolvida sobre a Internet como um primeiro protótipo de aplicação utilizando o protocolo REPI. Juntamente com a aplicação, avaliamos o comportamento da REPI quando estão presentes milhares de dispositivos utilizando o simulador NS-3.8.

Desenvolvemos a aplicação REPI-Internet utilizando o modelo P2P, pois o modelo é distribuído e se assemelha ao protocolo REPI. O desenvolvimento de uma rede P2P sobre a Internet gera alguns desafios como, por exemplo, realizar a comunicação direta entre dispositivos presentes em rede locais representados na Internet por NAT's (dispositivos tradutores de endereços). Nesta seção, apresentamos os principais desafios e as decisões de implementação tomadas para realizar a adaptação da REPI sobre a Internet.

4.1 *Peer to peer*

O modelo *Peer-to-peer*(P2P) é uma solução distribuída de larga escala para Internet, onde muitos usuários participam concorrentemente fazendo requisições ao sistema [27]. Várias aplicações fazem o uso do modelo P2P para compartilhamento de arquivos (exemplo Limewire [41], Kazaa [42], BitTorrent [31]). Pela propriedade distribuída, esses sistemas são escaláveis suportando centenas de milhares de usuários sem degradação do serviço. Em sistemas P2P, cada integrante que compõe a rede possui capacidades e responsabilidades equivalentes, diferente do modelo cliente/servidor, onde existe a distinção dos membros da rede de acordo com suas capacidades: os servidores são dispositivos com grande capacidade de processamento e são dedicados a proverem serviços, enquanto que os clientes são os usuários comuns que desejam usufruir desses serviços. Geralmente no modelo P2P todos os

indivíduos assumem o mesmo papel provendo e usufruindo dos serviços disponíveis. Normalmente, os integrantes desse tipo de rede são heterogêneos com capacidades de processamento e armazenamento distintos.

Várias aplicações utilizam esse modelo para compartilhamento de recursos computacionais, tais como: capacidade de processamento, espaço de armazenamento, serviços de programas, entre outros [27]. Para o estabelecimento de um sistema P2P, não há necessidade da existência de uma administração centralizada; por esse fato, são considerados sistemas distribuídos. Construir um sistema totalmente P2P é difícil, pois a estrutura atual da Internet dificulta o estabelecimento de conexões diretas entre dois dispositivos; isto implica na construção de sistemas híbridos baseados nas características P2P desejadas.

A principal preocupação com relação a uma rede P2P é como cada indivíduo se integra ao sistema. A solução amplamente utilizada faz uso do modelo cliente/servidor [27, 31, 42]. Uma vez que o dispositivo se integra à rede, o mesmo passa a fazer uso do esquema distribuído para colaboração. Para criar um sistema P2P funcionando na Internet é necessária a criação de uma rede sobreposta (*overlay network*). Uma rede sobreposta é a formação de uma rede lógica onde há a formação de ligações lógicas entre vizinhos lógicos que abstraem várias questões físicas da infraestrutura existente na rede real. A formação de uma rede sobreposta é feita em nível de aplicação, ou seja, os algoritmos de roteamento existentes na camada de rede continuam sendo os mesmos.

4.1.1 *Network Address Translator*

O NAT (*Network Address Translator*) causa dificuldades bem conhecidas na comunicação P2P [43, 44], uma vez que os dispositivos envolvidos não podem ser acessados por qualquer endereço IP válido. Na Internet existem dois tipos de endereços IP, são eles: os endereços públicos, acessíveis por qualquer parte da rede mundial; e os endereços privativos, usados em redes locais e acessíveis apenas pelos membros da mesma rede local. Para que os dispositivos em espaços de endereçamentos distintos possam se comunicar é necessário um conversor para mapear os endereços privativos em endereços públicos. O NAT é um método pelo qual os endereços IP são mapeados de um espaço de endereçamento para outro, provendo roteamento transparente para os dispositivos finais.

Para seu funcionamento, o NAT faz uso de uma tabela onde são mapeadas todas as conexões existentes entre os dispositivos locais com dispositivos externos. Cada entrada na tabela, denominada de sessão, é identificada por uma 5-tupla (Porta utilizada no NAT, IP local, porta local, IP remoto, porta remota). Cada sessão NAT tem um tempo de validade e varia de acordo com o sistema operacional. Só é

possível se comunicar com um dispositivo em uma rede local caso o NAT contenha uma sessão mapeada em sua tabela com o IP/porta do dispositivo remoto, caso esta sessão exista as mensagens são então encaminhadas para o IP/porta local contida na sessão.

O dispositivo NAT é o responsável por fazer uma ponte entre os espaços de endereçamento público e o privativo, ou seja, entre a rede local e a rede mundial. O NAT deve ter no mínimo duas interfaces onde: uma tem um endereço público que o conecta com a Internet; e outra com endereço privativo que o conecta a rede local. Toda a comunicação de saída de uma rede local passa pelo NAT, onde estas mensagens tem seu endereço de origem alterado, sendo substituído pelo endereço público do NAT. Sempre que um dispositivo na rede local inicia uma comunicação com algum dispositivo remoto, uma nova sessão é criada na tabela. As mensagens externas à rede local são aceitas pelo dispositivo NAT caso exista uma sessão em sua tabela correspondente ao endereço do dispositivo de origem da mensagem, caso essa sessão exista o endereço de destino das mensagens é substituído pelo endereço local mapeado na sessão.

A Figura 4.1 ilustra o funcionamento do NAT. Neste exemplo, o *host* A iniciou a comunicação com o Servidor S, neste caso a mensagem enviada por A chegará ao dispositivo NAT com o endereço de origem 10.0.0.10:538 e destino 20.14.23.47:80. O NAT irá adicionar em sua tabela a sessão referente à conexão de A com S e irá mapear uma porta para realizar esta comunicação, no exemplo a porta 230 foi escolhida. A mensagem enviada por A terá seu endereço de origem substituído pelo endereço público do NAT, ou seja, terá o endereço de origem 18.18.0.23:230. As mensagens oriundas de S com destino A, serão endereçadas utilizando o endereço público do NAT (18.18.0.23:230). Para todas as mensagens recebidas, o NAT procura em sua tabela se a sessão existe, verificando a porta e o endereço de origem da mensagem, no caso o endereço de S, caso a sessão exista o NAT substitui o endereço de destino para o endereço de A (10.0.0.10:538) e a encaminha para a rede local.

Os dispositivos NAT são barreiras para as aplicações P2P, isto porque estes dispositivos somente permitem o estabelecimento de conexões originadas de suas redes locais, ou seja, os NATs descartam requisições não solicitadas pela rede local, sendo que tais requisições são necessárias para as aplicações P2P [45]. Existem várias formas de permitir a comunicação entre dispositivos em redes locais distintas [44–47]. Uma alternativa, utilizada neste trabalho, é a técnica chamada de *hole punching* [44] que consiste em criar sessões nas tabelas NAT para permitir que sejam estabelecidas as conexões necessárias para as aplicações P2P.

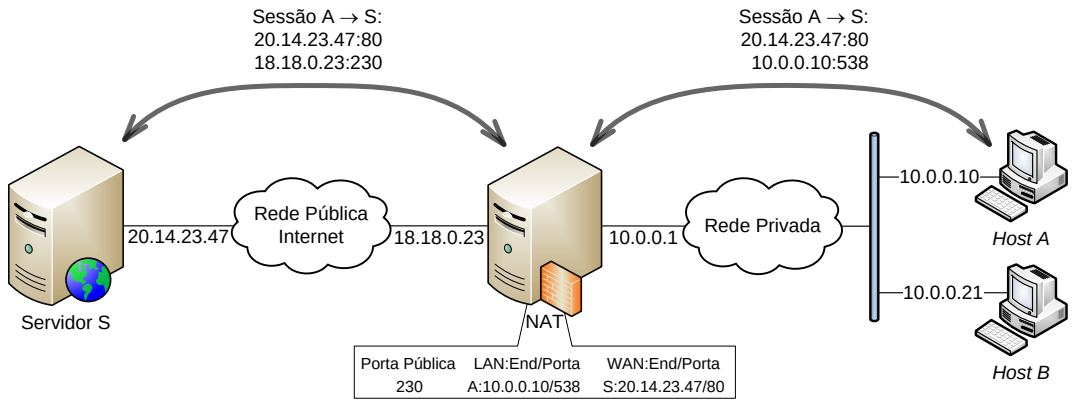


Figura 4.1: Exemplo de funcionamento do NAT

Técnica *Hole Punching*

A técnica *hole punching* permite que dois dispositivos se conectem diretamente com a ajuda de um ponto de encontro globalmente conhecido, mesmo estando em redes locais por trás de um dispositivo NAT. Esta técnica está detalhada em [44]. Um cenário simples composto de três dispositivos será usado para demonstrar o funcionamento da técnica em questão. Neste cenário, dois dispositivos A e B querem se comunicar diretamente, porém os NATs de suas redes locais impedem que a conexão seja estabelecida, logo para resolver este problema um terceiro dispositivo conhecido[1] por A e B é utilizado como ponto de encontro entre estes dispositivos. O *hole punching* assume que os dois dispositivos A e B que querem se comunicar diretamente estão conectados ao ponto de encontro S, como ilustra a Figura 4.2.

A Figura 4.2 mostra o estado das conexões no inicio do processo de estabelecimento de conexão entre A e B. Os dispositivos A e B têm conexões estabelecidas com o ponto de encontro S, logo seus NATs têm mapeado em suas tabelas as sessões que permitem que S se comunique com eles. Pode-se observar na figura que S conhece apenas o endereço do $NAT A$ (18.81.12.36) e $NAT B$(19.14.20.13), sendo os dispositivos NAT os responsáveis por repassarem as mensagens oriundas de S para seus respectivos destinos em suas redes locais.

Supondo que A queira se comunicar diretamente com o dispositivo B, a Figura 4.3 ilustra o processo de estabelecimento desta conexão usado pela técnica de *hole punching*. O processo consiste de três passos, são eles:

- Pedido de estabelecimento de conexão: neste passo o dispositivo A faz uma requisição para S, com o intuito de descobrir o endereço público usado pelo dispositivo B, representado pela mensagem 1 na Figura 4.3. Como A e B já têm conexões com S, S conhece seus endereços públicos, desta forma S pode

[1] Conhecido neste contexto significa que este terceiro dispositivo tem um IP público acessível por qualquer outro dispositivo na Internet.

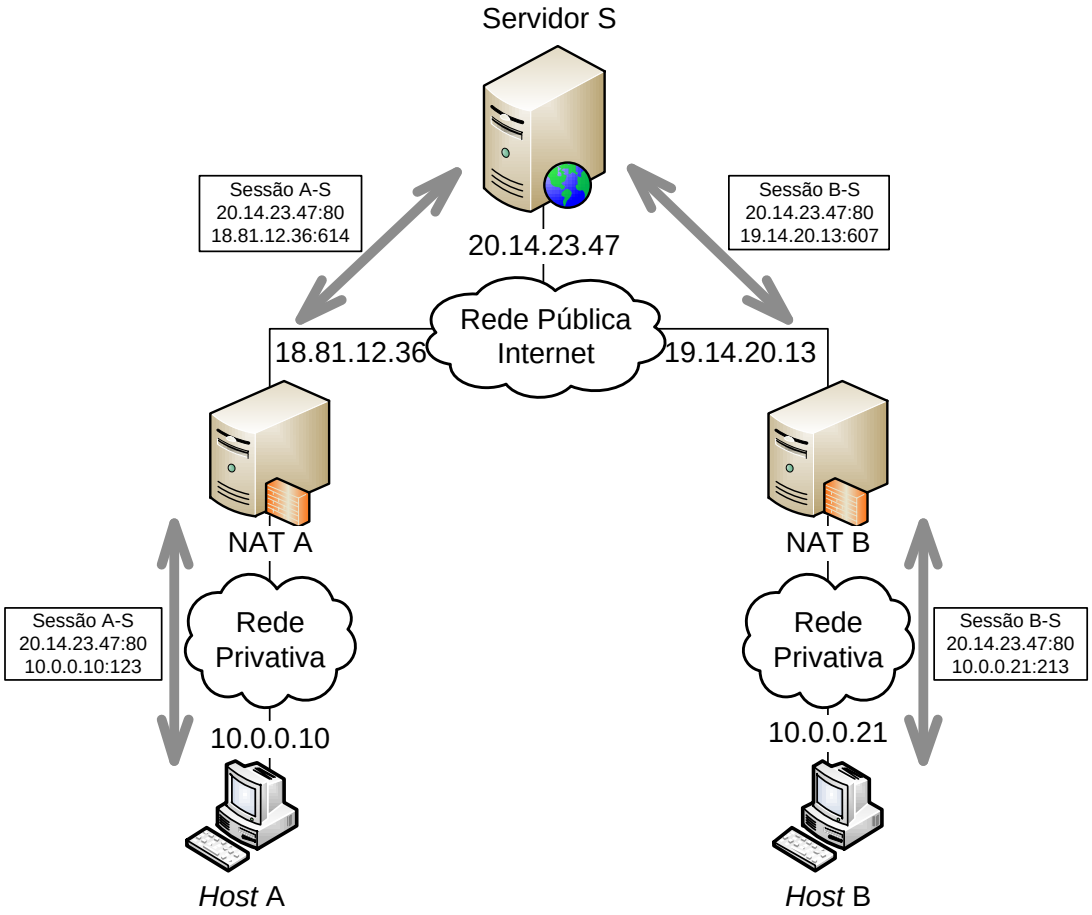


Figura 4.2: Exemplo: *Hole Punching*, antes do processo

dizer a ambos os dispositivos o endereço público utilizado pelo outro;

- Descoberta de endereços: neste passo o dispositivo S responde para A o endereço público utilizado por B, ao passo que avisa a B que A quer se comunicar diretamente com ele, representado pelas mensagens 2 na Figura 4.3. Na mensagem enviada por S para A está o par endereço:porta pública (19.14.20.13:607) utilizados por B, desta forma A poderá enviar uma mensagem para B por conhecer seu endereço público. O mesmo ocorre com a mensagem enviada de S para B, pois contém o par endereço:porta pública (18.81.12.36:614) utilizados por A, assim B poderá enviar mensagens diretamente para A. Uma otimização neste processo é enviar nas mensagens oriundas de S os endereços privativos de A e B, desta forma no processo de conexão, caso A e B estejam em uma mesma rede local não seja necessário o estabelecimento de sessões NAT para a comunicação direta entre eles;

- Estabelecimento de conexão: neste passo, o dispositivo A já conhece o endereço público utilizado por B, sendo assim ele envia uma mensagem utilizando este endereço com o intuito de criar uma sessão em seu dispositivo NAT para per-

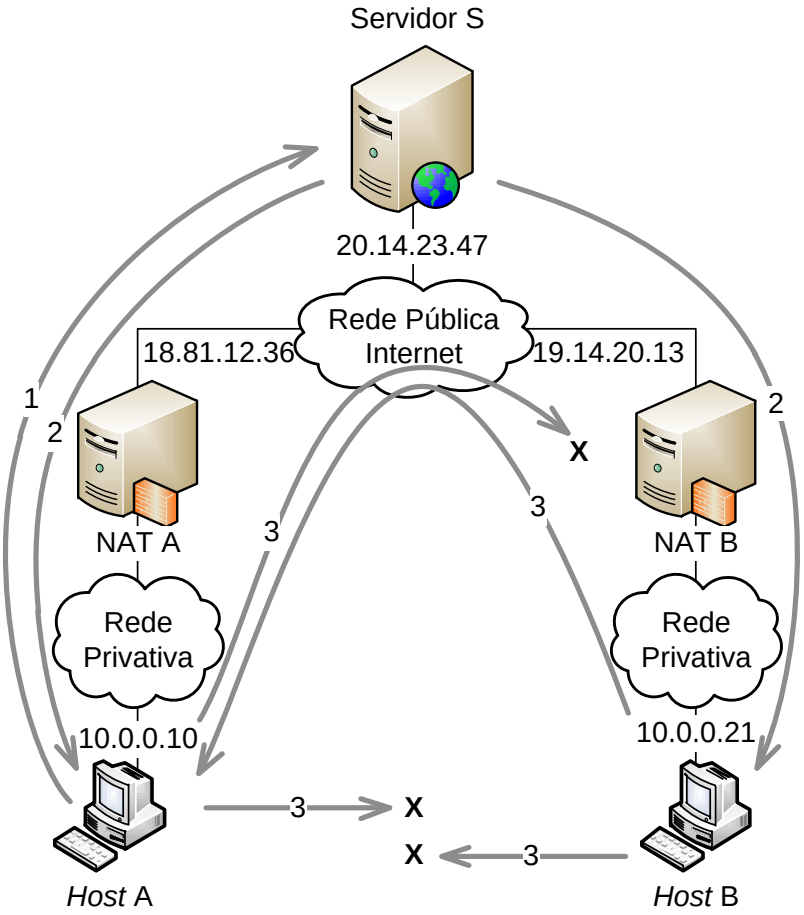


Figura 4.3: Exemplo: *Hole Punching*, durante o processo

mitir que mensagens oriundas de B sejam aceitas pelo $NATA$. O dispositivo B realiza o mesmo processo, enviando uma mensagem para o endereço público de A para criar a sessão em seu NAT e que possa se comunicar diretamente com A. Pode ocorrer da mensagem de A chegar ao $NATB$ antes de B ter enviado a mensagem para A, logo esta mensagem será descartada (representada pela mensagem 3 na Figura 4.3). Porém quando a mensagem de B for enviada, a sessão será criada no $NATB$ e o $NATA$ irá aceitar a mensagem pois a sessão foi criada previamente. Caso esteja utilizando o endereço local dos dispositivos para fins de otimização, A e B também irão enviar uma mensagem utilizando estes endereços, caso A e B estejam em uma mesma rede local, estas mensagens chegarão e assim a conexão será estabelecida localmente. No caso ilustrado na figura, A e B estão em redes distintas e assim estas mensagens são descartadas.

Após o processo do *hole punching* as sessões estão estabelecidas então A e B podem se comunicar diretamente sem a intervenção de S, como ilustra a Figura 4.4. Esta técnica pode ser utilizada tanto com os protocolos UDP [48] quanto TCP

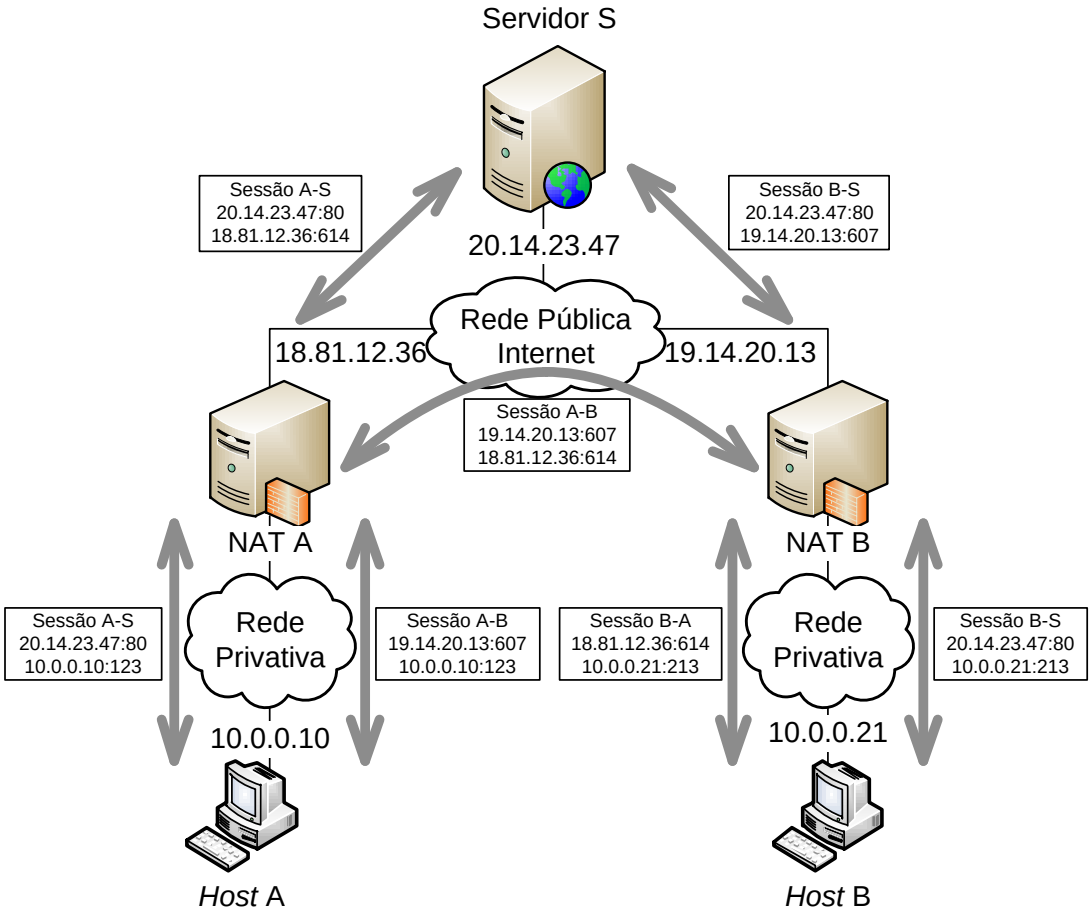


Figura 4.4: Exemplo: *Hole Punching*, após o processo

[49] porém, alguns dispositivos NATs ainda não suportam totalmente o caso em que se estabelecem as conexões utilizando TCP [44]. O complicador no caso do TCP são os pacotes de *handshake*, pois ao tentar iniciar a conexão, em alguns casos, os dispositivos NAT ao invés de descartar a requisição respondem com uma mensagem de erro anulando o processo de estabelecimento da conexão.

4.2 Formação da Rede REPI

Para adaptar a REPI à Internet é necessária a formação de uma rede sobreposta com o objetivo de abstrair as questões físicas da rede IP existentes na Internet. Esta rede formada permite que os dispositivos se comuniquem diretamente independente de sua localização (rede pública ou privativa). Neste trabalho desenvolvemos um algoritmo para formar uma rede P2P onde cada dispositivo tem uma lista de vizinhos aos quais consegue trocar mensagens diretamente. Para a formação desta rede usou-se como base o algoritmo do Gnutella [32, 50].

Pelo problema discutido anteriormente na seção 4.1.1, um dispositivo ao entrar na rede deve se comunicar inicialmente com um dispositivo conhecido globalmente,

nomeamos este dispositivo de nó Origem. Todos os dispositivos ao entrarem na rede se comunicam com o nó Origem para iniciar o processo de descoberta de vizinhos. Estes vizinhos são dispositivos que já estão presentes na rede. Vale ressaltar que esta comunicação com o nó Origem só ocorre no momento em que o dispositivo entra na rede, pois é necessário que exista um ponto de partida para a formação da rede.

O único diferencial do nó Origem em relação aos demais dispositivos é que seu endereço público deve ser conhecido por todos os outros que desejam participar da mesma rede, logo o nó Origem tem a mesma função do dispositivo S descrito na seção 4.1.1. Todos os dispositivos também tem a funcionalidade de poder estabelecer conexões entre outros dois dispositivos, pois o algoritmo executado em todos os nós, incluindo no nó Origem, é o mesmo. O nó Origem apenas auxilia no processo de estabelecimento de conexão do primeiro vizinho de cada dispositivo que entra na rede, sendo assim a sobrecarga sobre ele é reduzida.

Vale ressaltar que vários nós Origem podem ser utilizados para garantir um nível mínimo de robustez. Existindo mais nós Origem, caso algum deles esteja indisponível, os outros continuarão realizando suas tarefas estabelecendo conexões com os novos integrantes da rede. Outro fato importante é que podem existir diversas redes REPI, onde cada uma delas será estabelecida utilizando seu próprio nó Origem, estas redes poderão ser interconectadas caso pelo menos um dispositivo esteja presente em ambas as redes.

A formação da rede consiste em cada dispositivo estabelecer uma vizinhança composta por outros dispositivos. O processo se inicia quando um dispositivo entra em contato com o nó Origem, este por sua vez é o responsável por escolher o primeiro vizinho em sua lista, de acordo com algum critério de vizinhança. O dispositivo recém-chegado ao estabelecer a conexão com o primeiro nó descoberto, requisita a ele um novo vizinho. Este novo nó descoberto será o responsável por mediar o estabelecimento de conexão com outro dispositivo. Este processo de busca de vizinhos se repete até que seja atingindo uma determinada quantidade mínima de vizinhos ou que receba o endereço de um dispositivo já presente em sua lista de vizinhos.

O processo de busca de vizinhos é limitado com o intuito de reduzir o custo com a formação da rede e de limitar a quantidade de mensagens que serão encaminhadas por cada dispositivo, pois as mensagens REPI recebidas por um dispositivo são encaminhadas para todos os membros de sua lista de vizinhos. Ter um número muito grande de vizinhos pode sobrecarregar a rede na região próxima a este dispositivo causando perda destes pacotes. Além do fato de o tempo de entrega das mensagens ser maior, pois como as mensagens são encaminhadas individualmente para cada vizinho, o último nó a receber a mensagem encaminhada será penalizado com um tempo de entrega superior ao dos outros vizinhos.

Várias aplicações do tipo P2P também limitam a quantidade de conexões de acordo com a capacidade de rede de cada dispositivo, como BitTorrent [31], Emule [51], Kaaza [42], entre outros. Neste trabalho avaliamos o impacto do tamanho da lista de vizinhos na rede, mais detalhes no capítulo 5.

No processo de busca de vizinhos, cada dispositivo quando recebe uma requisição de estabelecimento de conexão, seleciona em sua lista de vizinhos o endereço de um de seus vizinhos de acordo com um critério de vizinhança. Este critério de vizinhança tem o objetivo de tentar otimizar a rede durante a sua formação, como cada dispositivo têm informações referentes a seus vizinhos, pode ser interessante fazer uso destas informações para estabelecer as conexões da melhor maneira possível. Neste trabalho foram desenvolvidos dois tipos de critério de vizinhança, são eles:

- Aleatória: este critério de seleção de vizinhos é a forma mais simples de escolha, pois seleciona aleatoriamente um dispositivo na lista de vizinhos e envia para o requisitante;

- Por PA: este critério faz uso de informações da REPI para tentar conectar da melhor forma os dispositivos na rede. Para isso, cada dispositivo deve conhecer o PA de cada um de seus vizinhos, tal informação é adquirida através das mensagens de controle utilizadas para a formação da rede (na seção 4.2.1 encontram-se mais detalhes sobre essas mensagens de controle). Neste caso, conhecendo o PA dos vizinhos e do requisitante, o nó escolhe o vizinho com maior quantidade de campos semelhantes no PA e estabelece a conexão entre eles.

Outros critérios de vizinhança podem ser facilmente adicionados no procedimento de busca de vizinhos. Para adicionar um novo critério, as informações necessárias para a tomada de decisão devem ser enviadas através da mensagem de controle de requisição de vizinhos, dessa forma quando um dispositivo receber esta informação poderá escolher o melhor vizinho dentre os presentes em sua lista de acordo com o novo critério. Com estes critérios de vizinhança é possível reduzir o tempo de entrega de mensagens, número de saltos ou qualquer outra variável desejada. No Capítulo 5 avaliamos o impacto de cada critério de vizinhança desenvolvido.

4.2.1 Algoritmo REPI-Internet

A lista de vizinhos presente em cada dispositivo tem o objetivo de armazenar informações necessárias para conexão e na decisão do critério de vizinhança utilizado. Essas informações são atualizadas sempre que uma nova mensagem de controle chega. Todas as mensagens são compostas pelos campos básicos: TipoMensagem, HTL, endereço IP e porta local do transmissor. Esses campos são utilizados pelos

dispositivos para realizar a identificação da mensagem, controle de encaminhamento, o controle local de formação e manutenção da rede, respectivamente. O restante da mensagem varia de acordo com o seu tipo. Existem sete tipos de mensagens, são elas:

- *Hello*: uma mensagem de controle utilizada no primeiro contato entre dois dispositivos. Essa mensagem é composta pelos campos básicos descritos anteriormente mais informações necessárias para a decisão do critério de vizinhança do transmissor da mensagem. Ao receber essa mensagem o receptor adiciona o transmissor como novo vizinho ou atualiza seus dados caso já esteja presente na lista de vizinhos do receptor e responde com uma mensagem *HelloAck*;

- *HelloAck*: uma mensagem de controle utilizada para responder as mensagens *Hello* recebidas. Esta mensagem é semelhante a *Hello*, contendo os mesmo campos, sendo esta apenas uma resposta com intuito de atualizar as informações referentes ao transmissor, no destinatário. Ao receber esta mensagem, o receptor adiciona o transmissor na lista de vizinhos (caso este ainda não esteja presente) ou atualiza as informações recebidas. Se o processo de descoberta de vizinhos ainda não tiver conseguido encontrar um determinado número mínimo de dispositivos, então o receptor faz uma requisição de um novo vizinho enviando uma mensagem *RequestPeer*;

- *RequestPeer*: uma mensagem de controle utilizada para fazer o pedido de um novo vizinho. Com as informações recebidas na mensagem, o receptor busca em sua lista de vizinhos um membro que melhor atenda ao critério de vizinhança escolhido, caso não encontre seleciona um nó aleatoriamente. Depois de selecionado um vizinho, uma mensagem *SendPeer* é enviada tanto para o nó requisitante quanto para o vizinho escolhido. Assim os dois nós (o requisitante e o selecionado) poderão estabelecer uma conexão e se tornarem vizinhos;

- *SendPeer*: uma mensagem de controle utilizada para enviar as informações básicas sobre os dispositivos na rede, como os endereços público e privativo de um nó. Ao receber essa mensagem o receptor verifica se o dispositivo recebido na mensagem já esta presente em sua lista de vizinhos, caso não esteja envia uma mensagem *Hello* para este nó em questão;

- *KeepAlive*: uma mensagem de controle utilizada para verificar se os atuais vizinhos de um dispositivo ainda estão conectados, além de renovar as sessões NAT com estes dispositivos. Essa mensagem é enviada de tempos em tempos para cada vizinho, esse tempo deve ser menor que tempo de sessão NAT que,

de acordo com [52], gira em torno de 2 minutos. Esta mensagem é composta pelos campos básicos mais as informações utilizadas pelo critério de vizinhança com o intuito de manter a informações sobre os vizinhos sempre atualizadas;

- *StillAlive*: uma mensagem de controle utilizada como resposta para a mensagem *KeepAlive* com o objetivo de informar ao destinatário que o dispositivo transmissor ainda está vivo, além de renovar as sessões NATs, caso exista. Semelhante ao *RequestPeer*, esta mensagem é composta pelos campos básicos mais as informações utilizadas pelo critério de vizinhança;

- *REPI*: essa mensagem é composta pelos campos básicos mais a mensagem REPI encapsulada. Ao receber essa mensagem ela é tratada pela função de casamento sendo encaminhada de acordo com os campos de C; e sendo ou não aceita de acordo com os campos de I do receptor.

Essas mensagens se dividem em três grupos:

- Formação da Rede: as mensagens pertencentes a esse grupo têm como objetivo criar um canal de comunicação entre dois dispositivos da rede aumentando o número de vizinhos de um nó. As mensagens que compõe esse grupo são: *Hello*, *HelloAck*, *RequestPeer*, *SendPeer*;

- Manutenção da Rede: as mensagens pertencentes a esse grupo têm como objetivo manter os canais de comunicação entre os dispositivos considerados vizinhos na rede. As mensagens que compõe esse grupo são: *KeepAlive*, *StillAlive*;

- Mensagem REPI: são as mensagens de interesses, criadas pela aplicação.

Nesta implementação, a aplicação contém duas listas de vizinhos: uma para armazenar os nós pertencentes à mesma rede privativa e outra para armazenar os nós de outras redes. Essa distinção é feita pelo fato de que os dispositivos em uma mesma rede privativa podem se comunicar diretamente sem o intermédio do NAT da rede local ao qual pertencem. Como o endereço utilizado é local e não são válidos na Internet, não podem ser enviados para outros membros fora da rede local. Com isso a aplicação consegue gerenciar tanto seus vizinhos locais quanto seus vizinhos na Internet. Isso também implica que, mesmo sem acesso à Internet, a aplicação funciona montando a rede com os membros presentes localmente.

O algoritmo para formação da rede REPI, denominado algoritmo REPI, é uma variação da técnica de *hole punching* que estabelece conexões entre os dispositivos, mas de forma distribuída. O algoritmo REPI estabelece a conexão e a mantém sempre ativa, assim como as sessões nas tabelas NAT caso existam, além de gerenciar

a tabela de vizinhos de cada dispositivo mantendo apenas os vizinhos conectados. A seguir a descrição detalhada do algoritmo:

Legenda:

- T: Nó transmissor da mensagem;
- R: Nó receptor da mensagem;
- $V(X)$: Nó selecionado na lista de vizinhos do nó X;
- N_i: Dispositivo i qualquer da rede;
- NV: Quantidade de vizinhos descobertos;
- NMV: Quantidade mínima de vizinhos que o algoritmo de descoberta deve encontrar;
- T_{NAT}: Tempo de renovação de sessão do NAT. Esse tempo deve ser menor que o tempo de sessão do NAT;

```
1 Ao receber uma mensagem Hello faça:
2   Adiciona/Atualiza dados de T na lista de vizinhos de R
3   Envia para T uma mensagem HelloAck
```

Passo 1: Mensagem *Hello*

As mensagens de *Hello* são utilizadas no primeiro contato entre dois dispositivos. O Passo 1 descreve o tratamento ao receber uma mensagem de *Hello*. A mensagem de *Hello* é utilizada por um dispositivo para iniciar o processo de estabelecimento de conexão com outros integrantes da rede. Quando um dispositivo recebe esta mensagem, as informações referentes ao transmissor são adicionadas ou atualizadas (caso já exista) na lista de vizinhos. Como resposta, uma mensagem de *HelloAck* é enviada para o dispositivo de origem do *Hello*.

```
4 Ao receber uma mensagem HelloAck faça:
5   Adiciona/Atualiza dados de T na lista de vizinhos de R
6   Se (NV < NMV/2) OU (Random(0,1) < (1 − NV/NMV)) então
7     Envia para T uma mensagem RequestPeer
8   Fim-se
```

Passo 2: Mensagem *HelloAck*

Para toda mensagem *Hello* recebida, o dispositivo receptor responde com uma mensagem *HelloAck*. Quando um dispositivo recebe uma mensagem *HelloAck* significa que a mensagem *Hello* enviada anteriormente foi recebida, ou seja, neste passo do algoritmo temos a garantia de que as sessões nas tabelas NAT dos envolvidos

no processo foram criadas e desta forma podem se comunicar diretamente sem o intermédio de outro dispositivo.

Ao receber uma mensagem de *HelloAck*, o receptor atualiza/adiciona os dados do transmissor em sua lista de vizinhos, assim como descrito no Passo 2. Nesta etapa, o processo de busca de novos vizinhos entra em ação. Este processo consiste em requisitar uma nova conexão ao transmissor do *HelloAck*, porém este processo deve ser executado apenas para descobrir uma quantidade mínima de dispositivos (NMV) pré-determinada com o intuito de reduzir a quantidade de mensagens de controle no processo de busca de novos vizinhos.

Esta limitação no processo de busca foi implementada através de uma heurística que avalia a quantidade de vizinhos descobertos e decide se deve continuar o processo de busca. A busca por novos vizinhos se inicia enviando uma mensagem *RequestPeer*. Antes de enviar esta mensagem, verifica-se a quantidade atual de vizinhos descobertos (NV) comparando com a quantidade mínima de vizinhos (NMV) a serem encontrados. A mensagem *RequestPeer* será enviada caso não tenha descoberto 50% do mínimo de vizinhos determinado, ou baseado em uma probabilidade inversamente proporcional à quantidade de nós atualmente presentes na lista de vizinhos em relação ao máximo permitido. Ou seja, quanto mais dispositivos estão presentes na lista de vizinhos menor é a probabilidade de se requisitar um novo vizinho.

```
9  Ao receber uma mensagem RequestPeer faça:
10   Seleciona V(R) baseado em um critério de vizinhança
11   Se nenhum nó foi escolhido então
12     Seleciona V(R) aleatoriamente
13   Fim-se
14   Se V(R) ≠ T então
15     Envia uma mensagem SendPeer com T para V(R)
16     Envia uma mensagem SendPeer com V(R) para T
17   Fim-se
```

Passo 3: Tratamento da mensagem de *RequestPeer*

Qualquer dispositivo na rede tem a capacidade de estabelecer conexões entre outros dois dispositivos através das mensagens de *RequestPeer* e *SendPeer*. Ao enviar uma mensagem do tipo *RequestPeer*, o dispositivo está requisitando ao destinatário que estabeleça uma nova conexão com algum de seus vizinhos, ou seja, o processo de *hole punching* como descrito na seção 4.1.1.

Quando algum dispositivo recebe uma mensagem de *RequestPeer*, ele escolhe algum vizinho em sua lista seguindo algum critério de vizinhança. Para estabelecer a conexão entre os dois dispositivos, o requisitante e o selecionado, o dispositivo envia para cada um deles o endereço do outro utilizando uma mensagem do tipo *SendPeer*. O Passo 3 descreve este processo de estabelecimento de conexão.

Desta forma, cada dispositivo ao receber a mensagem *SendPeer* passará a conhecer o endereço público utilizado pelo outro, podendo assim iniciar a comunicação.

```
18 Ao receber uma mensagem SendPeer faça:
19   Se o nó recebido na mensagem não está na lista de vizinhos então
20     Envia uma mensagem Hello para esse nó recebido
21   Fim-se
```

Passo 4: Tratamento da mensagem de *SendPeer*

Quando um dispositivo recebe uma mensagem *SendPeer*, ele verifica se o dispositivo recebido já está presente na lista de vizinhos, caso não esteja envia uma mensagem *Hello* para o endereço recebido, o Passo 4 descreve este processo. Neste passo do algoritmo o dispositivo recebido não é adicionado na lista de vizinhos, pois não há garantia de que a conexão será estabelecida. Só há garantia de conexão quando a mensagem de *Hello* ou *HelloAck* for recebida, pois essas mensagens são originadas pelo nó ao qual se quer estabelecer a conexão.

```
22 Para cada V(N_i):
23   Se N_i não enviou/recebeu mensagem qualquer de V(N_i) no intervalo de
     tempo T_NAT então
24     Envia mensagem de KeepAlive para V(N_i)
25   Fim-se
```

Passo 5: Manutenção da lista de vizinhos

As mensagens de *KeepAlive* e *StillAlive* são utilizadas para manutenção das sessões criadas nas tabelas NAT dos dispositivos presentes na lista de vizinhos. O tempo de sessão nas tabelas NAT é pequeno para o protocolo UDP, portanto se o NAT não receber nenhuma mensagem referente a uma determinada sessão durante este intervalo de tempo a sessão deixa de existir. Para evitar que as sessões deixem de existir impedindo um dispositivo de se comunicar com seus vizinhos, de tempos em tempos é enviada uma mensagem de controle *KeepAlive*. Como descreve o Passo 5, uma mensagem *KeepAlive* é enviada caso nenhuma outra mensagem tenha sido enviada ou recebida durante o intervalo de tempo menor que o tempo de sessão NAT.

```
26 Ao receber uma mensagem KeepAlive faça:
27   Atualiza os dados de T na lista de vizinhos de R
28   Envia para T uma mensagem StillAlive
```

Passo 6: Tratamento da mensagem de *KeepAlive*

Além da função de renovar as sessões nas tabelas NAT de cada dispositivo, a mensagem *KeepAlive* tem a função de avaliar se um vizinho ainda está presente na

rede com o intuito de manter a lista sempre atualizada. Quando um dispositivo recebe a mensagem *KeepAlive* ele responde com uma mensagem *StillAlive*, dando o sinal de que ainda está presente na rede. O Passo 6 descreve o tratamento da mensagem *KeepAlive*. As mensagens *KeepAlive* e *StillAlive* contém informações utilizadas pelo critério de vizinha, logo ao recebê-las essas informações são atualizadas na lista de vizinhos.

```
29 Ao receber uma mensagem StillAlive faça:
30    Atualiza os dados de T na lista de vizinhos de R
```

Passo 7: Tratamento da mensagem de *StillAlive*

A mensagem *StillAlive* é apenas uma confirmação de que um dispositivo ainda está presente na rede e de que a sessão NAT ainda existe, ela é usada como resposta à mensagem *KeepAlive*, como mostra o Passo 7. Caso não seja recebida nenhuma mensagem *StillAlive* de um dos vizinhos após ter enviado duas mensagens *KeepAlive*, este dispositivo é eliminado da lista de vizinhos. A mensagem *StillAlive* contém informações recentes utilizadas pelo critério de vizinhança. O dispositivo ao receber a mensagem *StillAlive* atualiza estas informações sobre o dispositivo em sua lista.

```
31 Ao receber uma mensagem REPI faça:
32    Avalia a mensagem usando a função de casamento
33    Se houver casamento nas características então
34       Envia essa mensagem para todos os vizinhos de T
35    Fim-se
36    Se houver casamento no interesse então
37       Aceita a mensagem repassando para a aplicação
38    Fim-se
```

Passo 8: Tratamento da mensagem *REPI*

As mensagens REPI contêm os dados que o usuário deseja difundir na rede. Esses dados vão trafegar pela rede passando por vários dispositivos colaborativamente. Quando um dispositivo recebe esta mensagem, ele avalia seu PA encaminhando-a para todos seus vizinhos, caso ocorra casamento nas características; e aceitando-a caso ocorra casamento no interesse da mensagem. O filtro de casamento pode ser qualquer, mas o utilizado neste trabalho é o filtro parcial que determina a ocorrência do casamente caso pelo menos um dos campos seja semelhante ao comparar, campo a campo, o PA da mensagem com o do dispositivo. O tratamento da mensagem *REPI* está descrito no Passo 8.

Como abordado anteriormente, redes P2P em geral necessitam de um ponto de encontro para iniciar a formação da rede. Este ponto de encontro é um dispositivo onde todos devem se comunicar inicialmente para poder participar da rede. Para

39 **Quando N_i inicia o sistema (comunicação com o nó Origem):**
40 Envia uma mensagem *Hello* para todos os nós Origem
41 Envia uma mensagem *Hello* em *broadcast* na rede local

Passo 9: Entrada de um dispositivo na rede

entrar na rede, um dispositivo envia uma mensagem do tipo *Hello* para todos os nós Origem predefinidos e assim se inicia o processo de descoberta de vizinhos. A rede também é formada com dispositivos presentes na rede local onde não há conexão com a Internet. Pois no inicio do sistema a aplicação envia uma mensagem em *broadcast* em sua rede local com o intuito de descobrir se existem dispositivos que também estejam conectados ao sistema. O Passo 9 descreve este processo.

Todas as mensagens de controle (*Hello*, *HelloAck*, *RequestPeer*, *SendPeer*, *KeepAlive*, *StillAlive*) têm em torno de 85 bytes, contando com os 20 bytes de cabeçalhos do datagrama. Apenas a mensagem do tipo REPI é maior de acordo com o tamanho do corpo da mensagem que o usuário quer enviar.

4.3 Sobrecarga de Mensagens da REPI-Internet

Durante o processo de descoberta de vizinhos, descrito pelo algoritmo apresentado na sessão anterior, a quantidade de mensagens trocadas na rede para descobrir um vizinho tende a um limite definido. A Figura 4.5 ilustra o processo de descoberta de vizinhos quando um nó N entra na rede. As linhas tracejadas mais escuras representam as conexões previamente estabelecidas, pode-se observar que a lista de todos os dispositivos (representado na figura pelos círculos) tem em sua lista o dispositivo ao qual está conectado, enquanto que as linhas tracejadas mais claras representam as novas conexões estabelecidas após a execução do algoritmo de busca de vizinhos. As linhas restantes representam a troca de mensagens entre os dispositivos e a numeração indica a ordem de geração destas mensagens.

De acordo com o algoritmo, a mensagem de *Hello* é utilizada para iniciar o processo de descoberta de vizinhos. Observando a Figura 4.5, temos que um nó N está entrando na rede, os nós A e B já estão presentes na rede há algum tempo, logo já existe alguns dispositivos presentes em suas listas de vizinhos. Quando o dispositivo N inicia o procedimento de descoberta de vizinhos, envia uma mensagem de *Hello* para o nó *Origem* (representado pela linha 1). No algoritmo, quando um nó recebe esta mensagem, adiciona o nó na lista de vizinhos e responde com um *HelloAck*, na figura o *HelloAck* enviado pelo nó *Origem* é representado pela linha 2. O dispositivo N ao receber a mensagem enviada pelo *Origem*, verifica que sua lista esta vazia e envia um *RequestPeer*, representado na figura pela linha 3. Quando o nó *Origem* recebe este *RequestPeer*, escolhe em sua lista um nó de acordo com o

critério de vizinhança adotado e envia para N através da mensagem *SendPeer* com o endereço do nó selecionado (mensagem representada pela linha 4). Neste caso o nó B foi selecionado, logo para estabelecer a conexão entre N e B, o nó *Origem* também envia para B um *SendPeer* (linha 5 na figura) com o endereço de N para que ambos os dispositivos possam enviar mensagens um para o outro.

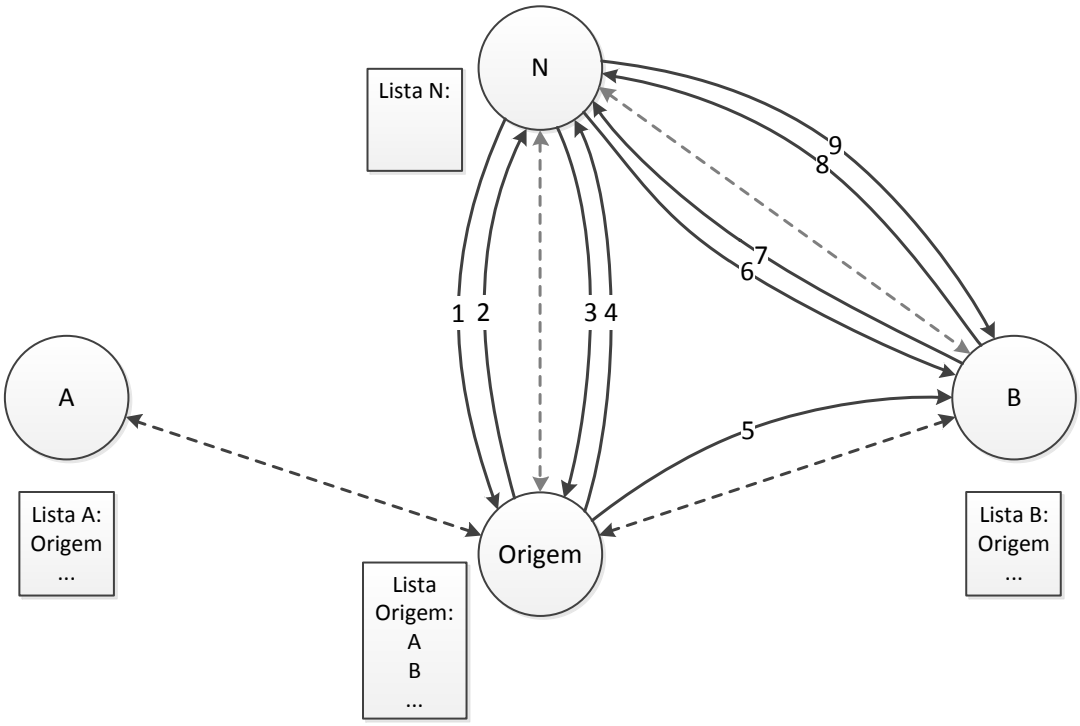


Figura 4.5: Mensagens durante o processo de descoberta de um vizinho

Após N e B receberem a mensagem oriunda do nó *Origem*, ambos enviam uma mensagem *Hello* um para o outro (na figura estas mensagens são representadas pelas linhas 6 e 8 respectivamente). Quando estas mensagens chegarem a seus destinos, o receptor irá adicionar em sua lista o transmissor da mensagem criando assim um canal de comunicação. Estas últimas mensagens *Hello* enviadas são responsáveis por iniciar um novo ciclo para descoberta de novos vizinhos, assim N que acaba de descobrir B irá repetir o processo requisitando um novo vizinho ao nó B.

Neste processo foram necessárias 7 mensagens para N descobrir B, esse número chega ao máximo de 9 contando com as duas últimas mensagens *HelloAck* enviadas no final de um ciclo. Estas duas últimas mensagens são enviadas mesmo se um nó não necessitar descobrir novos vizinhos, isso ocorre pois no algoritmo quando se recebe uma mensagem *HelloAck* é o momento em que o nó decide se deve requisitar um novo vizinho. Baseado neste valor conseguimos avaliar a quantidade máxima de mensagens de controle transmitidas para realizar e manter a rede formada. As equações a seguir descrevem as quantidades de mensagens de controle na rede, em

seus vários níveis:

$$M_{PV} = 7 \tag{4.1}$$

$$M_{MPV} = 2 * \left(\frac{T}{T_{NAT}} \right) \tag{4.2}$$

$$M_{PN} = NV * (M_{PV} + M_{MPV}) \tag{4.3}$$

$$M_{Total} = M_{PN} * N \tag{4.4}$$

$$M_{Origem} = NV * (M_{PV} + M_{MPV}) + 5 * (N - 1) \tag{4.5}$$

M_{PV} (equação 4.1) é a quantidade de mensagens de controle necessárias para descobrir um vizinho, como descrito anteriormente este processo de descoberta necessita de 7 mensagens. Cada nó tem uma lista com determinado número de vizinhos, sendo NV esta quantidade. O número de mensagens necessárias para encontrar os NV vizinhos é o produto entre esta quantidade de vizinhos e a quantidade de mensagens para se descobrir um vizinho ($M_{PV} * NV$).

Todas as conexões estabelecidas por um dispositivo são avaliadas de tempos em tempos T_{NAT}, com o intuito de verificar se cada nó ainda está presente na rede além de renovar as sessões NAT criadas ao estabelecer esta conexão, caso exista. Utiliza-se a mensagem *KeepAlive* para a verificação e renovação das conexões, como resposta cada vizinho envia uma mensagem *StillAlive*. Estas duas mensagens são necessárias para manter a conexão com cada vizinho em um intervalo de tempo T_{NAT}, logo a quantidade de mensagens de manutenção por vizinho M_{MPV} (equação 4.2) é igual à quantidade de vezes em que as mensagens de manutenção são enviadas, multiplicada por dois (dois tipos de mensagens *KeepAlive* e *StillAlive*). Um nó tendo em sua lista NV vizinhos, a quantidade de mensagens de manutenção será igual ao produto entre esta quantidade de vizinhos e a quantidade de mensagens de manutenção por vizinho ($M_{MPV} * NV$).

A quantidade de mensagens de controle utilizadas por um nó, M_{PN}, é a quantidade utilizada para descobrir os vizinhos presentes em sua lista além das mensagens para manter a conexão com estes dispositivos descobertos. A equação 4.3 descreve essa quantidade de mensagens sendo a soma entre a quantidade de mensagens necessárias para descobrir um vizinho M_{PV} e a quantidade de mensagens M_{MPV} necessárias para manter esta nova conexão ativa em um intervalo de tempo T avaliado, multiplicado pela quantidade de vizinhos descobertos NV.

A quantidade total de mensagens de controle M_{Total} na rede para o estabelecimento da vizinhança e a manutenção das conexões criadas, descrita na equação 4.4, é o produto da quantidade de nós na rede N pela quantidade máxima de mensagens de controle por nó M_{PN}. Esta quantidade de mensagens leva em consideração a

quantidade gasta individualmente em cada nó, sendo a soma de todas as mensagens que por ventura serão trocadas na rede durante o tempo avaliado.

O nó *Origem* troca mais mensagens que todos os outros, pois cada nó ao entrar na rede entra em contato primeiramente com ele para descobrir seu primeiro vizinho. O nó *Origem* segue o mesmo algoritmo presente em todos os outros dispositivos na rede, logo será gasto $NV * M_{MPV}$ mensagens para manter sua lista de vizinhos. No processo de descoberta, como ilustra a figura 4.5, a quantidade de mensagens para um nó estabelecer a conexão entre dois dispositivos, caso representado pelas linhas de 1 a 5, são de cinco mensagens. Como todo dispositivo requisita seu primeiro vizinho ao nó *Origem*, são utilizadas cinco mensagens para o estabelecimento desta primeira vizinhança para todo novo nó que entra na rede, com isso tem-se que o nó *Origem* gasta, além das mensagens de manutenção, $5 * (N - 1)$ mensagens para o estabelecimento desta primeira conexão. Como o nó *Origem* tem o mesmo comportamento do restante dos nós na rede, ele gasta M_{PN} para a formação de sua vizinhança. Logo, como ilustra a equação 4.5 simplificada, o nó *Origem* gasta $NV * (M_{PV} + M_{MPV}) + 5 * (N - 1)$ mensagens.

4.4 Aplicação Mensageira na REPI

Com a rede formada, o usuário tem a capacidade de se comunicar com outros membros presentes na rede de acordo com o interesse desejado. A aplicação desenvolvida tem um conjunto de interesses iniciais onde os usuários podem se comunicar. O usuário tem o poder de criar novos interesses de acordo com sua vontade, além de criar senhas para cada novo canal de comunicação criado por ele. Estes canais com senha levam em consideração que os usuários tenham combinado previamente entre eles, para assim conseguir certo nível de privacidade, pois a priori a rede é totalmente pública e anônima.

Quando um usuário envia uma mensagem através de um determinado interesse, essa mensagem é encaminhada para todos os dispositivos presentes em sua lista de vizinhos. Cada vizinho ao receber esta mensagem, toma a decisão de encaminhá-la para seus vizinhos, caso ocorra casamento dos campos de C contido na mensagem com o do receptor. Se um determinado vizinho tiver interesse nesta mensagem recebida, tal mensagem é aceita e mostrada para o usuário pela aplicação.

A aplicação impede que uma mesma mensagem seja encaminhada mais de uma vez pelo dispositivo hospedeiro, essa memória é feita armazenando as últimas mensagens recebidas. Desta forma sempre que uma nova mensagem chega, ela é buscada nesta memória e descartada caso esteja presente. Essa memória funciona em conjunto com o filtro de casamento, como mais um meio de inibir a propagação das mensagens, além do encaminhamento probabilístico.

A aplicação mensageira tem implementado os três tipos de filtro de casamento descritos na seção 3.1.2, mas o utilizado por padrão é o filtro parcial. Neste tipo de filtro, como já descrito, as mensagens são encaminhadas para os vizinhos, caso pelo menos um dos campos C do PA da mensagem seja igual ao campo correspondente em C do PA do dispositivo receptor.

Foi utilizado o protocolo UDP [48] para estabelecer a conexão entre os dispositivos na rede. A escolha deste protocolo foi pelo fato de haver maior compatibilidade com a técnica *hole punching*, descrito anteriormente na seção 4.1.1. Apesar de o UDP ter a facilidade no estabelecimento de conexões passando por NATs, o tempo de sessão UDP em um NAT é pequeno, em torno de 2 minutos [52], o que força o envio periódico de mensagens para manter as sessões sempre ativas. Em nossa implementação optamos por utilizar um intervalo de tempo para envio de mensagens de renovação de 60 segundos. No caso do TCP [49] o tempo de sessão é muito grande e existe um processo de finalização da conexão, porém estabelecer conexões através de um NAT não é uma tarefa fácil por causa da complexidade do processo de estabelecimento de conexão utilizado por este protocolo. Mais detalhes sobre a aplicação estão disponíveis no apêndice A.

4.5 Simulação da REPI-Internet

Para realizar a avaliação da REPI-Internet foi utilizado o simulador Network Simulator (NS-3.8) [24] onde desenvolvemos o algoritmo REPI responsável por formar a rede sobreposta e as trocas de mensagens. O NS-3.8 é um simulador de eventos discretos e como descrito em [53] tem baixo consumo de memória e os menores tempos de computação em comparação com outros simuladores conhecidos pela comunidade. Foi utilizada a versão 3.8 disponível em [54].

O NS-3.8 é implementado em módulos, cada módulo agrupa um conjunto de funcionalidades. A tabela 4.1 ilustra em alto nível os agrupamentos de funcionalidades, cada grupo é composto por um ou mais módulos de acordo com o tipo de rede ao qual se quer simular. Como exemplo, no grupo de roteamento existe um módulo para cada protocolo implementando, como o AODV [55], OLSR [56], roteamento estático, entre outros.

O simulador cria algumas abstrações para facilitar o entendimento e a manipulação dos objetos nele contidos. Alguns jargões que são utilizados na Internet, como por exemplo o termo *host* ou sistema final para definir um dispositivo computacional conectado a rede não são adotados pelo NS-3.8, pois ele é um simulador de rede, não especificamente um simulador de Internet. Usa-se o termo nó para nomear um dispositivo computacional, um termo genérico adotado por outros simuladores. Um nó é composto por dispositivos de rede que são as interfaces de comunicação

Tabela 4.1: Organização do *software* NS-3.8

<table>
<tr><td colspan="4">test</td></tr>
<tr><td colspan="4">helper</td></tr>
<tr><td>routing</td><td>internet-stack</td><td>devices</td><td>applications</td></tr>
<tr><td colspan="2">node</td><td colspan="2">mobility</td></tr>
<tr><td colspan="2">common</td><td colspan="2">simulator</td></tr>
<tr><td colspan="4">core</td></tr>
</table>

que interconectam os nós da rede. Pode-se haver uma ou várias interfaces em cada nó. Em um mesmo nó pode haver uma ou mais aplicações que geram as atividades a serem simuladas.

O algoritmo descrito na sessão 4.2.1 foi implementado no NS-3.8 no nível de aplicação. Esta aplicação é a geradora das mensagens que são utilizadas para a formação da vizinhança em cada nó simulado. Vale ressaltar que nem todos os nós da rede tem a aplicação, estes outros nós são os roteadores que direcionam as mensagens para seus destinos como ocorre na Internet.

A aplicação gerencia a lista de vizinhos presente em cada nó e realiza a troca de mensagens usando o mecanismo da REPI, enviando para todos os presentes na lista de vizinhos, caso ocorra casamento nos campos de C da mensagem. Para facilitar a avaliação do algoritmo desenvolvido no NS-3.8 foi necessária a criação de alguns parâmetros, são eles:

- **TipoFiltro:** Tipo de filtro de casamento a ser usado (0: Total, 1: Parcial, 2: Sem filtro);
- **TipoAlgoritmo:** Tipo de algoritmo a ser usado (0: Com memória, 1: Sem memória);
- **TipoSelecaoNo:** Critérios de vizinhança para a seleção de vizinhos (0: Aleatório, 1: Maior quantidade de casamento no PA);
- **LimBuscaVizinhos:** Limite para o algoritmo de busca de vizinhos;
- **OrigemEVizinho:** Adiciona o nó origem como um vizinho. Caso o nó origem não seja vizinho, ele não recebe mensagens do tipo REPI;
- **QtdCamposC:** Quantidade de campos usados em C;
- **HtlMensagem:** Quantidade máxima de saltos para encaminhamento de uma mensagem REPI;
- **QtdPossibilidades:** Quantidade de possibilidades em cada campo C do PA;

- **QtdREPINos:** Quantidade de nós com aplicação REPI-Internet na simulação (N).

As variáveis computadas pelo sistema para realizar a avaliação são as seguintes:

- Quantidade de mensagens enviadas e recebidas de cada tipo: *Hello*, *HelloAck*, *RequestPeer*, *SendPeer*, *KeepAlive* e *StillAlive* somados compões as mensagens de controle MC, e REPI a mensagem de interesse MI;
- Quantidade de mensagens aceitas, quando ocorre casamento de I (MA);
- Quantidade de mensagens descartadas pelo não casamento parte C, limite de saltos atingido, perda de pacotes na rede (MD);
- Quantidade de mensagens encaminhadas, totais (MET) e colaborativas (MEC). Utilizadas para calcular a quantidade de nós colaboradores (NC);
- Quantidade de vizinhos (NV);
- Quantidade média de saltos de uma mensagem REPI (HTL);
- Quantidade de nós colaboradores (NC);
- Tempo para a entrega das mensagens REPI (T).

Para simular um ambiente no NS-3.8 é necessária a elaboração de cenários. Um cenário é uma representação da topologia ao qual se deseja simular, onde são dispostos os nós da rede e suas respectivas conexões. É necessário especificar os nós, quais protocolos e aplicações estão presentes em cada nó e os nós que serão fontes de pacotes. Para avaliação da REPI-Internet foram utilizados dois cenários, tais cenários são descrito em mais detalhes no decorrer desta seção.

4.5.1 Cenário Preliminar

O cenário preliminar foi utilizado para realizar uma avaliação preliminar com o intuito de validar o funcionamento da implementação REPI-Internet no simulador. Nesta avaliação foi observada a formação da vizinhança, além do custo em mensagens inerente a esta formação. A topologia deste cenário é ilustrada na Figura 4.6. Nesse cenário a parte central da rede é composta por um anel com cinco roteadores e os nós com a aplicação REPI são divididos igualmente a cada roteador formando sub-redes. O nó Origem foi associado a um dos roteadores do anel como ilustra a figura. Este cenário foi nomeado de cenário preliminar pela baixa complexidade de conexões dos dispositivos e pelo fato deste cenário ser utilizado apenas para validar

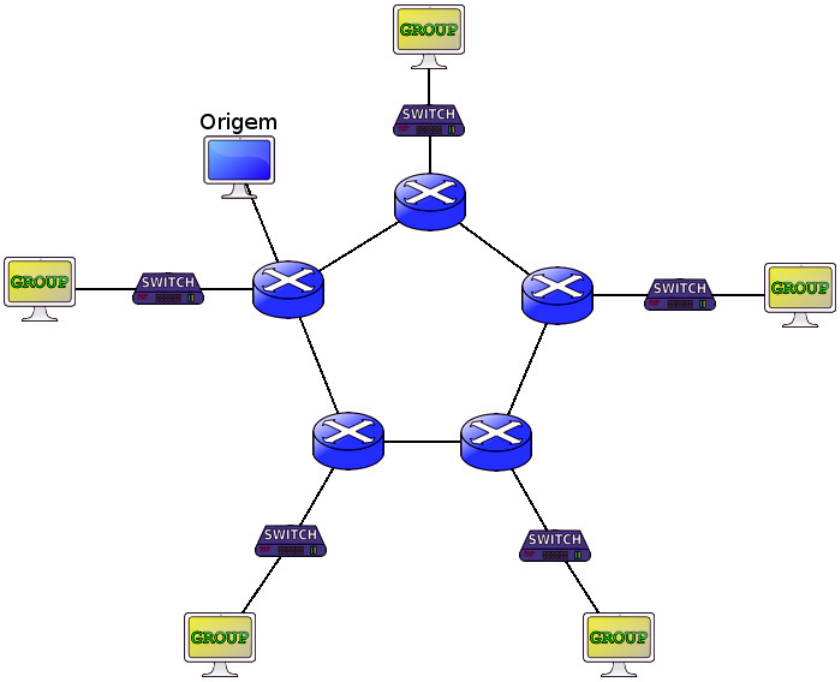


Figura 4.6: Topologia do Cenário Preliminar

o funcionamento do algoritmo demonstrando a formação da vizinhança em cada dispositivo.

A simulação utilizando este cenário foi uma forma inicial de demonstrar que a quantidade de mensagens de controle utilizadas para formação e manutenção da rede está diretamente ligada à quantidade de vizinhos de cada dispositivo e estão limitadas de acordo com as equações descritas na seção 4.3.

4.5.2 Cenário Rede Rio

Depois de validado a implementação no NS-3.8, partimos para uma simulação em uma topologia mais realista. Usamos como base a topologia da Rede Rio [1] (outubro de 2010). A Figura 4.7 ilustra as interconexões do *backbone* da Rede Rio. Utilizamos este cenário para realizar simulações com muitos dispositivos para analisar a sobrecarga de mensagens quando se tem uma grande quantidade de nós na rede.

O objetivo deste cenário é avaliar o comportamento do sistema em um ambiente realista tentando se aproximar ao máximo do custo da Internet. Um cenário realista deve se aproximar com algo que já exista no mundo real, por esse fato escolheu-se a formação de um *backbone* semelhante à Rede Rio. Nesse cenário temos cinco tipos de *links* com capacidades diferentes que variam desde 500 Kbps à 1 Gbps.

Os nós com a aplicação simulada foram distribuídos entres estes *links* presentes no cenário tentando balancear consumo da rede da melhor maneira. Utilizou-se a seguinte equação para o balanceamento dos *links*:

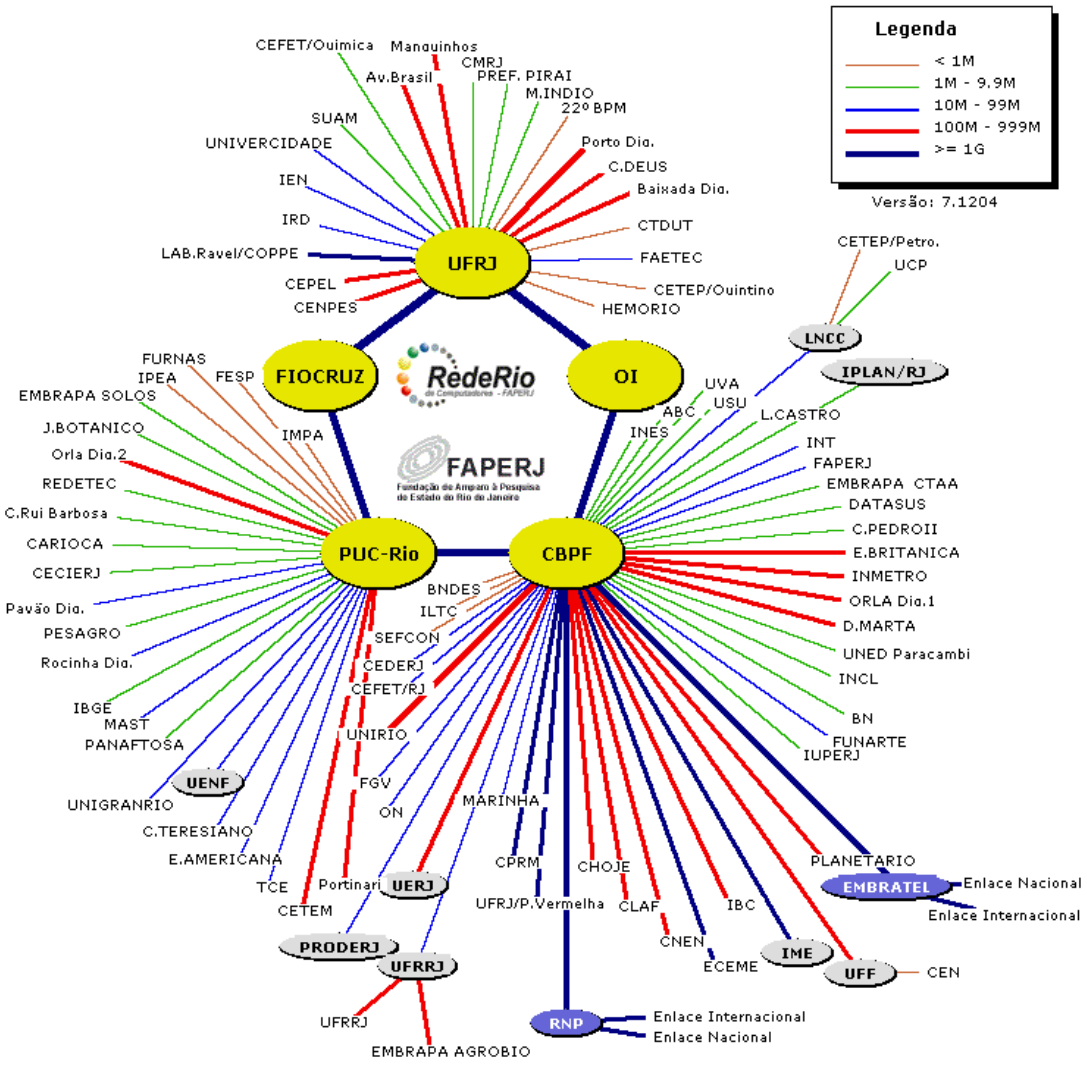


Figura 4.7: Topologia do Cenário da Rede Rio [1]

$$P(i) = \frac{N_i * i}{\sum_{j \in C} (N_j * j)} \tag{4.6}$$

Onde:

- i: é a capacidade do *link*
- N_i: é a quantidade de *links* com capacidade i
- $P(i)$: é o percentual de nós alocados nos *links* com capacidade i
- C: é o conjunto de capacidades existentes [500 Kbps,1 Mbps,10 Mbps, 100 Mbps,1 Gbps]

A equação 4.6 divide proporcionalmente os nós entre cada classe de *links* existentes mantendo um nível de balanceamento. Conhecendo a quantidade de nós a serem alocados em cada *link* é possível criar sub-redes com certa quantidade de nós. As sub-redes em cada classe têm a quantidade de nós baseado na capacidade do *link*.

Em *links* de 1Gbps são criadas sub-redes com no máximo 16 nós. Nos *links* de 100Mbps são criadas sub-redes com nó máximo 8 nós. No caso dos *links* de 10Mbps são criadas sub-redes com no máximo 4 nós. Em *links* com 1Mbps são criadas sub-redes com 2 nós cada, assim como em links com 500Kbps são criadas sub-redes com apenas um nó.

4.5.3 Simulação distribuída

O NS-3.8 permite realizar simulações distribuídas utilizando o padrão MPI para troca de mensagens [57]. Ao criar um cenário no NS-3.8, os nós devem ser associados aos processos em execução. As aplicações geradoras de tráfego devem ser alocadas apenas nos processos que serão responsáveis por tratar seus eventos. Sabendo em qual processo cada nó está associado, o NS-3.8 realiza a troca de mensagens sem a necessidade de nenhuma nova implementação.

Utilizamos a simulação distribuída no cenário da Rede Rio, pelo fato de haver uma grande quantidade de nós geradores de eventos. Os nós são divididos uniformemente entre os processos gerados, para tentar balancear a carga de processamento entre os processadores. Mais detalhes sobre simulação distribuída no NS-3.8 estão disponíveis em [58].

Capítulo 5

Metodologia de Avaliação e Resultados para o Algoritmo REPI

A avaliação realizada neste trabalho utilizou o simulador de redes NS-3.8 [24]. Apesar de haver um protótipo da aplicação REPI, as simulações nos permitem avaliar o comportamento do algoritmo REPI em um ambiente controlado e de maior escala onde podemos avaliar os efeitos das variáveis no desenvolvimento do sistema. Nosso objetivo principal é verificar o desempenho do algoritmo desenvolvido para formar uma vizinhança e através desta realizar a entrega das mensagens de interesse seguindo o protocolo REPI.

A seguir descrevemos a metodologia de avaliação da REPI-Internet, onde detalhamos as métricas utilizadas e os objetivos de cada conjunto de simulações. Serão detalhados e analisados os resultados obtidos nas simulações de cada cenário apresentado.

5.1 Metodologia de Avaliação

Realizamos um estudo preliminar para avaliar a construção da rede, através da verificação da formação da vizinhança de cada dispositivo. Nesta primeira avaliação, analisamos a quantidade de mensagens de controle totais na rede. Utilizamos as mensagens recebidas como principal métrica, pois não houve perdas assim sendo todas as mensagens enviadas na rede foram recebidas pelos seus destinatários.

5.1.1 Notação

Apresentamos abaixo a notação utilizada nas métricas de avaliação do comportamento do sistema. As variáveis do sistema representam as médias dos valores obtidos

a partir de todas as repetições realizadas em cada simulação:

N: Total de dispositivos utilizando a aplicação REPI-Internet durante a avaliação. Nas simulações o número de dispositivos simulados é superior aos da aplicação REPI, pois alguns destes dispositivos simulados são apenas roteadores como ocorre na Internet. Este parâmetro contabiliza somente os dispositivos com aplicação REPI-Internet;

NC: Quantidade de nós que não tem interesse na mensagem, mas a encaminharam (nós colaboradores);

NI: Total de nós pertencentes ao grupo de interesse. É a quantidade de nós destinatários da mensagem enviada. Com esta variável é possível avaliar a proporção de nós interessados que receberam a mensagem, ou seja, ela é utilizada para calcular a taxa de entrega;

NR: Total de nós pertencentes ao grupo de interesse que receberam a mensagem. Ela é utilizada para calcular a taxa de entrega;

NV: Quantidade média de nós presentes na lista de vizinhos;

NVO: Quantidade de nós presentes na lista de vizinhos do nó Origem;

HTL: Esta variável mede a quantidade média de saltos de todas as mensagens de interesses entregues aos NI nós interessados;

MA: Quantidade de mensagens de interesse aceitas pelos NI nós interessados;

MC: Quantidade total de mensagens de controle recebidas por todos N nós. Como mencionado, utilizamos as mensagens recebidas pois não houve perdas. Esta variável é soma de todos os tipos de mensagens de controles presentes no sistema (*Hello*, *HelloAck*, *RequestPeer*, *SendPeer*, *KeepAlive*, *StillAlive*);

MCO: Quantidade de mensagens de controle recebidas apenas pelo nó Origem no processo de estabelecimento de vizinhança e manutenção das conexões;

MI: Quantidade total de mensagens de interesse REPI recebidas pelos N nós na rede;

MS: Quantidade de mensagens de interesse REPI injetadas na rede;

T: O tempo médio gasto para entregar todas as mensagens de interesse aos NI nós interessados.

5.1.2 Métricas

A seguir apresentamos as métricas utilizadas na avaliação do sistema. São elas:

C_{APN}: Custo em mensagens de controle de formação da rede por nó. Esta métrica avalia a formação da rede através do custo médio em mensagens de controle por nó. O objetivo é avaliar o custo em mensagens de controle em cada nó para realizar a formação e a manutenção da vizinhança. A equação 5.1 define como esta métrica é calculada.

$$C_{APN} = \frac{MC}{N} \tag{5.1}$$

C_{APV}: Custo em mensagens de controle de formação da rede por vizinho. Esta métrica avalia a formação da rede através do custo médio em mensagens de controle para descobrir cada vizinho em um nó. O objetivo desta métrica é mostrar o custo em mensagens para a formação da vizinhança, este custo é quantidade de mensagens média gasta para descobrir cada nó presente na lista de vizinhos. A equação 5.2 define o cálculo desta métrica.

$$C_{APV} = \frac{C_{APN}}{NV} \tag{5.2}$$

C_{APVO}: Custo em mensagens de controle de formação da rede por vizinho no nó Origem. Esta métrica avalia o custo médio em mensagens de controle por vizinhos sobre o nó Origem. O objetivo é avaliar o custo em mensagens por vizinho sobre o nó Origem para estabelecer a conexão entre os nós da rede. A equação 5.3 define como esta métrica é calculada.

$$C_{APVO} = \frac{MCO}{NVO} \tag{5.3}$$

C_{PNI}: Custo em mensagens REPI por nó interessado. Esta métrica avalia a troca de mensagens REPI através do custo médio em mensagens de interesse geradas na rede para que cada nó interessado possa recebê-las. O objetivo é avaliar o impacto do interesse na rede verificando o impacto de um interesse popular e não popular. A equação 5.4 define como esta métrica é calculada.

$$C_{PNI} = \frac{MI}{NR} \tag{5.4}$$

C_{RPN}: Custo em mensagens REPI geradas na rede por nó. Esta métrica avalia o custo médio em mensagens de interesse por nó. O objetivo é avaliar o custo em mensagens de interesse que foram geradas na rede para alcançar os nós

destinatários (nós com interesse). A equação 5.5 define como esta métrica é calculada.

$$C_{RPN} = \frac{MI}{N} \tag{5.5}$$

C_{RPV}: Custo em mensagens REPI geradas na rede por vizinho. Esta métrica avalia o custo médio em mensagens de interesse por vizinhos presente em cada nó. O objetivo é avaliar o custo em mensagens de interesse geradas em cada nó ao encaminhar uma mensagem. A equação 5.6 define como esta métrica é calculada.

$$C_{RPV} = \frac{C_{RPN}}{NV} \tag{5.6}$$

T_C: Taxa de colaboração. Esta métrica avalia taxa de colaboração dos nós na rede, ou seja, a fração de nós N da rede que encaminharam a mensagem colaborativamente. A equação 5.7 define como esta métrica é calculada.

$$T_C = \frac{NC}{N} \tag{5.7}$$

T_E: Taxa de entrega. Esta métrica avalia a taxa de entrega de mensagens aos nós interessados, ou seja, a fração de nós NI que receberam as mensagens de interesse. Devido a existência da memória, cada dispositivo interessado aceita as mensagens (MA) destinadas a ele apenas uma vez. No total, devem ser aceitas $MS * NI$, desde que uma mensagem (MS) enviada deve ser entregue a todos os nós interessados (NI). Assim sendo, equação 5.8 define como esta métrica é calculada.

$$T_E = \frac{MA}{MS * NI} \tag{5.8}$$

Utilizamos os cenários apresentados na seção anterior para realizar nossa avaliação. Primeiramente vemos a necessidade de validar a implementação desenvolvida no simulador NS-3.8, para isso foi utilizado o cenário preliminar 4.5.1. Este cenário foi utilizado para verificar a formação da vizinhança em cada dispositivo, além de examinar a influência de alguns dos parâmetros do algoritmo, como por exemplo o critério de vizinhança.

Depois de validado o funcionamento do algoritmo na formação da vizinhança, avaliamos a REPI enviando mensagens de interesse para grupos de nós na rede. Esta avaliação consistiu em verificar o potencial da REPI no contexto de difusão de informações na rede. Para simular esse contexto, utilizamos o cenário da Rede Rio

onde grupos de dispositivos (variando entre 20%, 15%, 10% e 5% dos dispositivos) têm o mesmo interesse que um determinado nó A. Este nó A faz o envio de uma mensagem para este grupo de dispositivos e verificamos quantos nós no grupo recebem esta mensagem e avaliamos o custo em mensagens para realizar esta entrega. O tempo de entrega dessas mensagens também é outra variável importante que foi verificada.

Apresentamos e analisamos os resultados destas simulações a seguir além de discutir os pontos importantes baseado nos resultados obtidos.

5.2 Resultados

Para cada cenário realizamos um conjunto distinto de simulações, pois cada cenário teve um propósito de avaliação diferente. Com o cenário preliminar verificamos o funcionamento da formação da lista de vizinhos em cada nó simulado. Para isso foi utilizado um número de nós variando de 50 a 100. Com poucos nós simulados a validação e visualização dos resultados são mais simples. Depois de avaliado o algoritmo REPI, distribuímos os nós simulados sobre uma rede baseada no *backbone* da Rede Rio de forma a equilibrar o consumo de cada um dos *links* existentes na rede.

A avaliação preliminar foi realizada em 4 máquinas Intel® Core™ 2 Quad Q8200 $2.33GHz$ com 4 núcleos de processamento e $4Gb$ de memória RAM. Esta avaliação foi executada sequencialmente pois não demandou tanto tempo de processamento quanto a avaliação REPI. Pela simplicidade do cenário utilizado neste caso, a análise do comportamento da formação da rede é mais simples e o curto tempo de simulação facilitou o processo de validação.

Para realizar as simulações da REPI e obter resultados em tempo hábil foi necessária a implementação deste cenário em um ambiente paralelo. Estas simulações foram realizadas em um *cluster* composto de 4 máquinas Intel® Xeon® CPU E5410 $2.33GHz$ com 8 núcleos de processamento e $8Gb$ de memória RAM, além de uma máquina Intel® Xeon® CPU E5620 $2.40GHz$ com 8 núcleos físicos e tecnologia *HyperThreading* e $16Gb$ de memória RAM. Contamos, também, com o *cluster* SGI Altix ICE 8200 do NACAD [59] composto por 64 CPUs Quad Core Intel® Xeon® E5355 (*Clovertown*), $2.66GHz$ e $16Gb$ de memória RAM contabilizando um total de 256 núcleos.

Nos resultados apresentados a seguir, avaliamos a influência do critério de vizinhança e a quantidade mínima para o algoritmo de busca de vizinhos. Estes parâmetros são apresentados nos gráficos a seguir da seguinte maneira:

- Eixo X: corresponde a quantidade de dispositivos presentes na rede na ava-

liação preliminar e a porcentagem de dispositivos nos grupos avaliados durante na avaliação REPI;

- Eixo Y: resultado obtido em cada parâmetro avaliado;
- Legenda: caracteriza cada curva do gráfico no modelo "NC (LB)". Onde NC é o nome do critério de vizinhança avaliado e LB a quantidade mínima de nós a serem descobertos pelo algoritmo de busca.

Em nossa avaliação utilizamos 8 campos de características com 8 possibilidades cada. Os nós simulados com a aplicação REPI criam seu PA no inicio da simulação, no instante em que entram na rede. Para cada campo de características é escolhido um valor dos possíveis dada uma probabilidade. Para definir o grupo interessado na mensagem, no inicio da simulação cada nó é escolhido aleatoriamente para participar do grupo de interesse por meio de uma distribuição uniforme.

Vale ressaltar que os resultados apresentados são valores médio de todas as simulações realizadas em cada cenário, este fato pode ter mascarado casos onde possa ter ocorrido resultados ruins.

5.2.1 Avaliação Preliminar

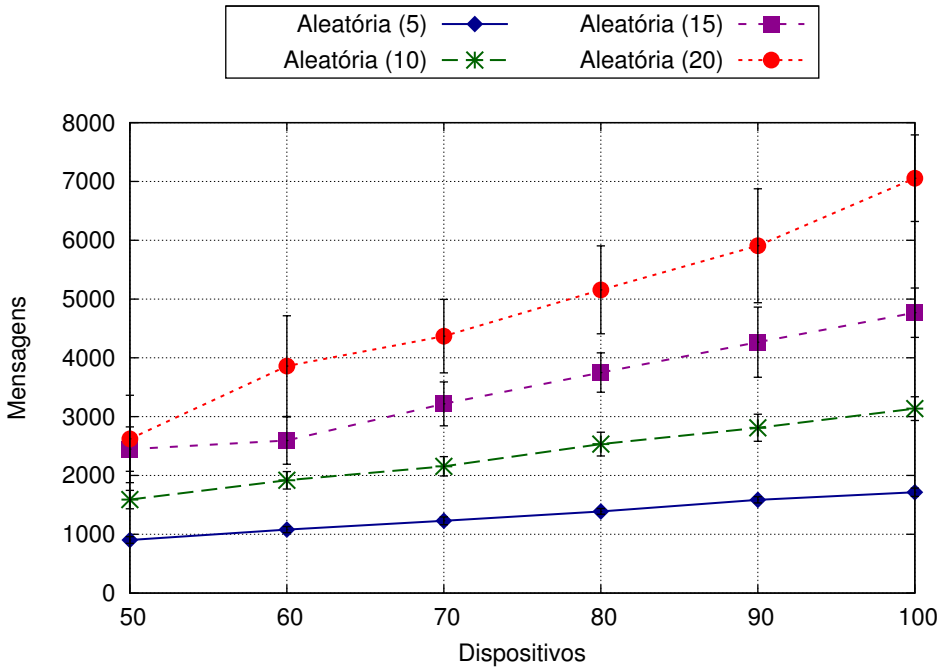
A avaliação preliminar contou com simulações composta de 20 execuções com duração de 100 segundos. O tempo de inicio de entrada de um nó na rede é escolhido aleatoriamente seguindo uma distribuição uniforme entre os valores de 0 a 20 segundos. O tempo de renovação das sessões NAT foi de 60 segundos, ou seja, com 80 segundos de simulação todos os dispositivos já entraram na rede e enviaram ao menos uma mensagem de renovação de sessão NAT.

A simulação foi realizada com 50 a 100 nós e variamos a quantidade mínima de vizinhos em que o processo de busca deve ser executado em cada nó entre 5, 10, 15 e 20 para verificar o custo de estabelecimento de conexão com cada vizinho descoberto. Avaliamos o impacto do critério de vizinhança (Aleatória ou por PA) na formação da rede. Os resultados apresentados são valores médios com nível de confiança de 95% e com erro menor que 5%.

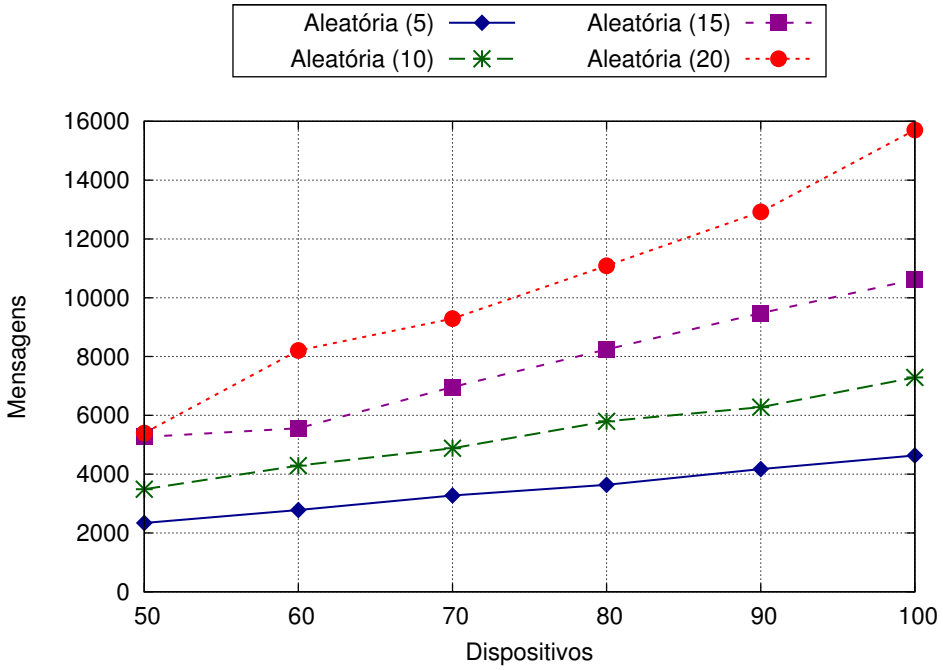
Vizinhança Aleatória

A Figura 5.1(a) mostra a quantidade de mensagens de controle totais recebidas na rede, para o critério de vizinhança aleatória. A taxa de crescimento da quantidade de mensagens de controle é pequena em comparação com o aumento da rede. Este crescimento é linear, porém com uma taxa de crescimento pequena a qual se deve a implementação da heurística mencionada anteriormente. Esta heurística impede que

mensagens sejam trocadas desnecessariamente, já que ter poucos vizinhos provoca um custo menor de mensagens na rede.



(a) Resultado Simulação



(b) Avaliação Analítica

Figura 5.1: Aleatória: Quantidade média de mensagens de controle totais recebidas (MC). (a) Simulado (b) Analítico

A avaliação analítica descrita na seção 4.3 parametrizado para os resultados obtidos na simulação utilizando o critério de vizinhança aleatória é mostrada na Figura 5.1(b). Os resultados obtidos na simulação em relação à quantidade de

mensagens de controle são metade dos obtidos na avaliação analítica. Isso se deve ao fato de que no caso da simulação foram contabilizadas apenas as mensagens recebidas enquanto que na avaliação analítica além das mensagens recebidas contabilizou-se as enviadas também. Logo, levando este fato em consideração, os resultados obtidos nas simulações são exatamente os esperados para o estabelecimento e manutenção da rede.

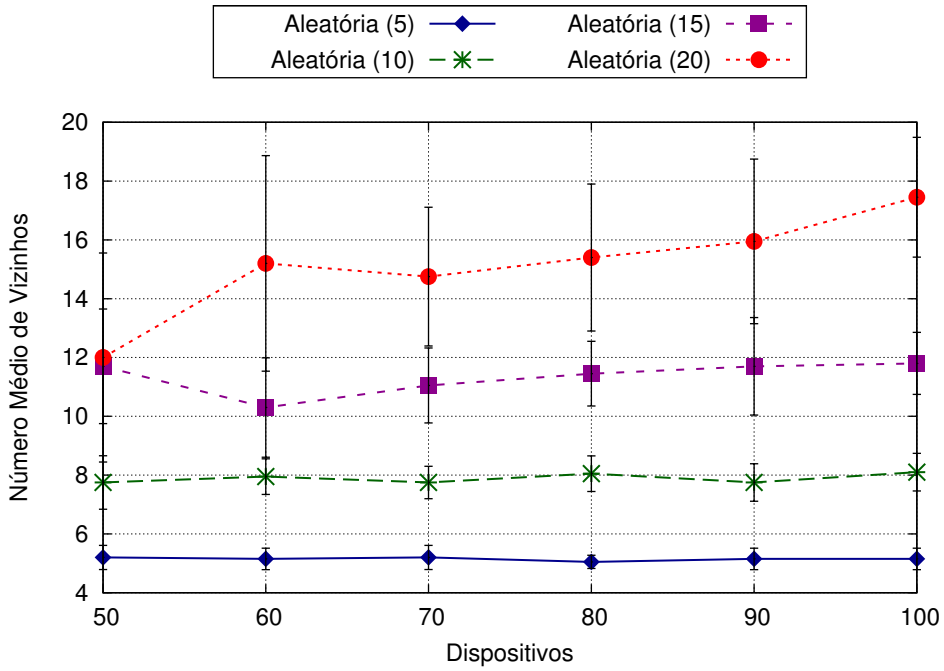


Figura 5.2: Aleatória: Número médio de vizinhos (NV)

A vizinhança foi formada atingindo quase o limite mínimo determinado para cada caso, como pode ser observado na Figura 5.2. É possível perceber a estabilidade do caso *Aleatória*(5), pois neste caso a heurística não é tão ativa quanto aos outros casos pelo fato de se descobrir uma quantidade pequena de vizinhos. No caso *Aleatória*(20), o número de vizinhos descobertos tende a ser constante independente da quantidade de nós na rede, mas existe uma instabilidade nos resultados (alta variância) devido a maior quantidade de nós a serem descobertos na rede o que demonstra a heurística em ação.

Para a formação da vizinhança obtivemos um custo por nó em mensagens constante em cada caso, sendo independente da quantidade de dispositivos na rede, como mostra a Figura 5.3. Tal custo é constante devido ao fato de esta depender apenas da quantidade de vizinhos descobertos na rede. Neste caso (5.3) o custo fica entre 60 e 70 mensagens por nó no pior caso e menos de 20 no melhor caso.

O custo para descobrir cada vizinho também é constante com a quantidade de dispositivos na rede, como esperado. Na avaliação analítica previámos que esta seria em torno de 7 mensagens, porém os resultados mostram que o custo em men-

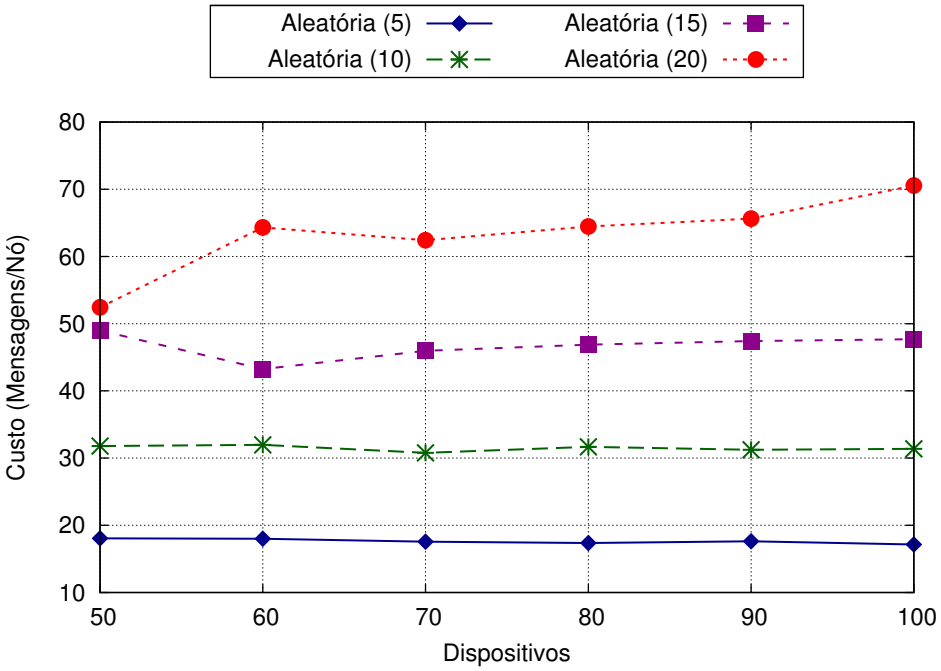


Figura 5.3: Aleatória: Custo em mensagens recebidas por nó (C_{APN})

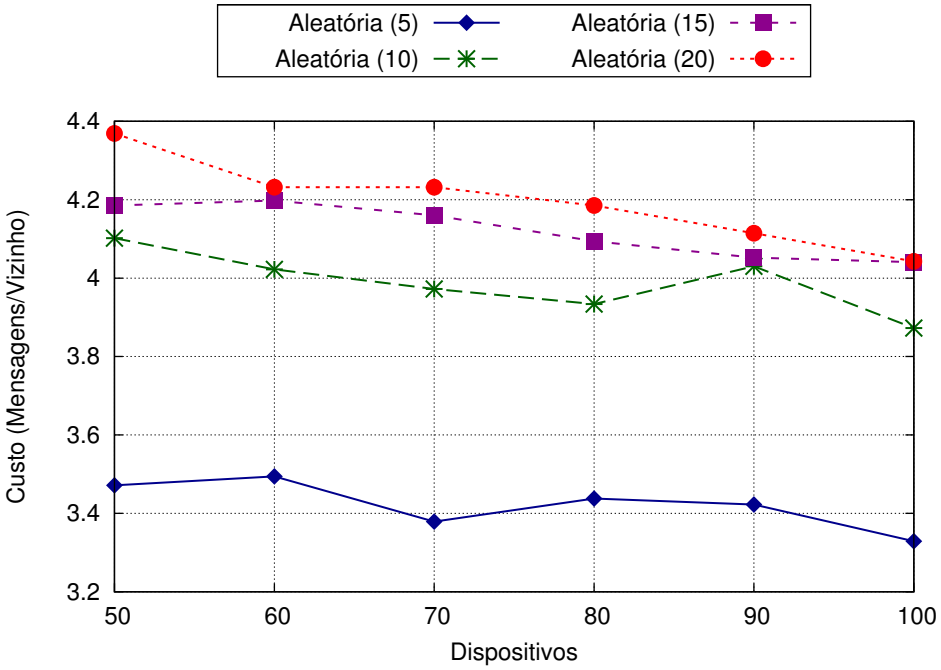


Figura 5.4: Aleatória: Custo em mensagens recebidas para descobrir um vizinho (C_{APV})

sagens recebidas é menor 5 mensagens, como mostra a Figura 5.4. Mas na avaliação analítica contabilizamos tanto as mensagens enviadas quanto recebidas enquanto que nas simulações contabilizamos apenas as mensagens recebidas, ou seja, a avaliação analítica tem o dobro de mensagens que o que deveria ser recebido nas simulações. Ou seja, nas simulações foi atingida uma quantidade inferior a 10 mensagens (envi-

adas e recebidas), valor um pouco superior ao custo para descobrir um nó, porém este custo por vizinho apresentado nas simulações contabiliza as mensagens de renovação de sessão NAT, com isso concluímos que este custo está dentro dos valores esperados.

Vizinhança por PA

Os resultados apresentados anteriormente são para o caso cujo critério de vizinhança é aleatória. A seguir apresentamos os resultados para o critério de vizinhança em que utilizamos o PA para selecionar os vizinhos.

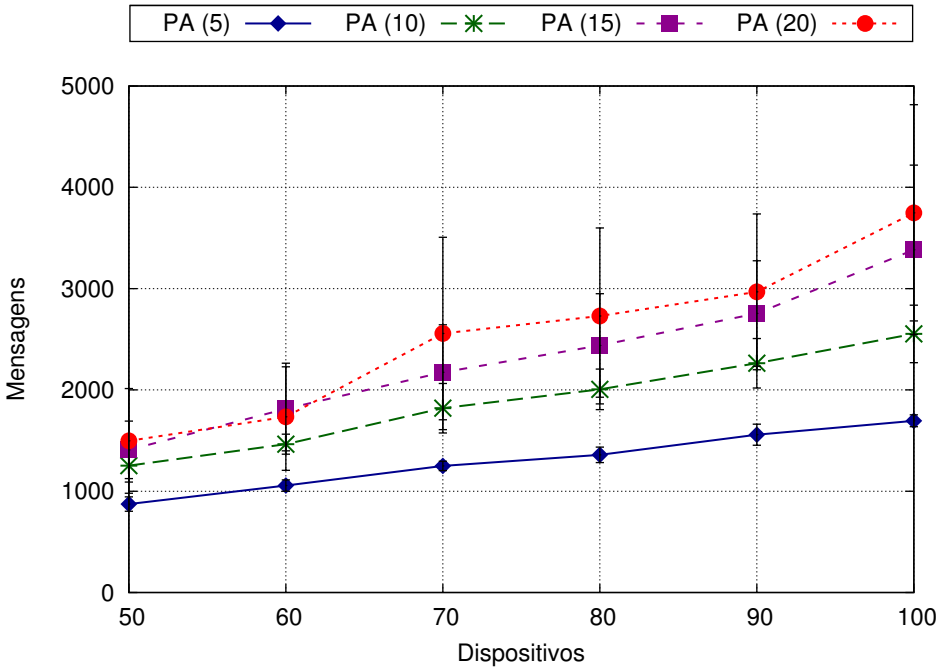
A quantidade de mensagens e número de vizinhos (5.5(a), 5.6) têm valores menores em comparação ao critério de vizinhança aleatória de seleção de vizinhos (5.1(a), 5.2). Ao utilizar o PA como critério de vizinhança, um dispositivo que já tem sua lista de vizinhos preenchida sempre retornará o mesmo vizinho quando outro o requisitar, pois o critério é a quantidade de campos iguais no PA e como a lista do nó requisitado permanece inalterada toda requisição a este nó requisitado retornará sempre o mesmo nó.

Receber um nó já presente na lista de vizinhos faz com que o algoritmo pare a busca de novos vizinhos, reduzindo o número de mensagens e a quantidade de vizinhos descobertos. O mesmo comportamento de instabilidade na descoberta de vizinhos ocorre no caso $PA(20)$, como descrito anteriormente isso ocorre devido à heurística que tenta reduzir a quantidade de mensagens na rede evitando que o nó continue indefinidamente tentando encontrar novos vizinhos.

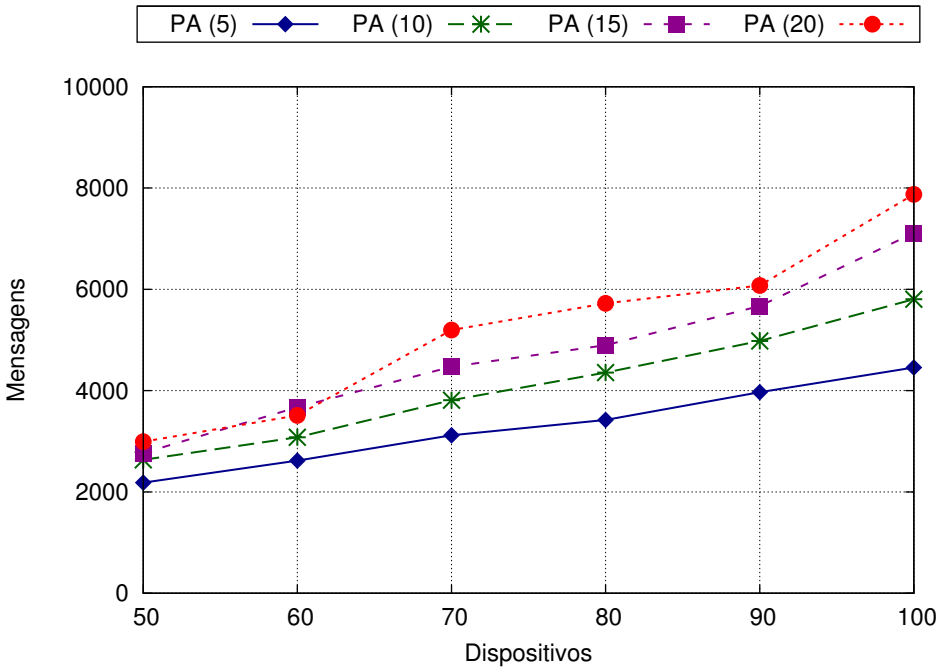
Em relação à avaliação analítica 5.5(b), os valores obtidos na simulação foram ligeiramente conservadores. Da mesma forma que no caso anterior (*Aleatória*), a quantidade de mensagens na avaliação analítica deveria ser o dobro que os resultados obtidos na simulação, porém a avaliação analítica foi ligeiramente superior ao dobro dos resultados obtidos na simulação.

Por haver menos mensagens na rede, o custo por nó tende a ser menor no caso do PA, ao comparar os critérios de vizinhança (5.3, 5.7) apesar de o custo para descobrir um vizinho ser ligeiramente maior (5.4, 5.8). O custo por vizinho é maior pelo fato já descrito anteriormente, pois requisitar um nó a um vizinho mais de uma vez é um desperdício de mensagem já que este vizinho irá retornar o mesmo nó que retornou anteriormente, caso sua lista não tenha sido alterada neste intervalo entre as requisições.

Semelhante ao caso anterior, o custo por vizinho descoberto ficou menor que 7 mensagens, como previámos na análise analítica, estando abaixo de 5 mensagens da mesma forma que caso anterior (*Aleatória*). Levando em consideração que o custo por nó obtido nas simulações leva em consideração uma fase de renovação das sessões nas NAT, ou seja, o custo em mensagens



(a) Avaliação Analítica



(b) Avaliação Analítica

Figura 5.5: PA: Quantidade média de mensagens de controle totais recebidas (MC). (a) Simulado (b) Analítico

Nó Origem

Realizamos uma análise em especial sobre o nó origem. Este tipo de nó tem o objetivo de iniciar a formação da vizinhança dos novos integrantes da rede, dessa forma todos os nós ao entrarem na rede irão se comunicar inicialmente com ele.

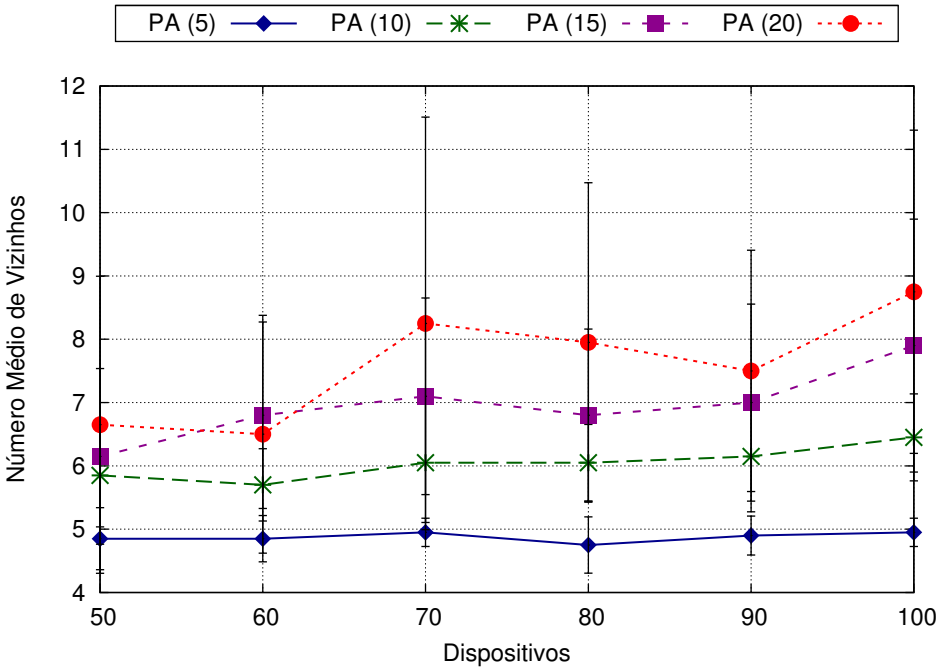


Figura 5.6: PA: Número médio de vizinhos (N_V)

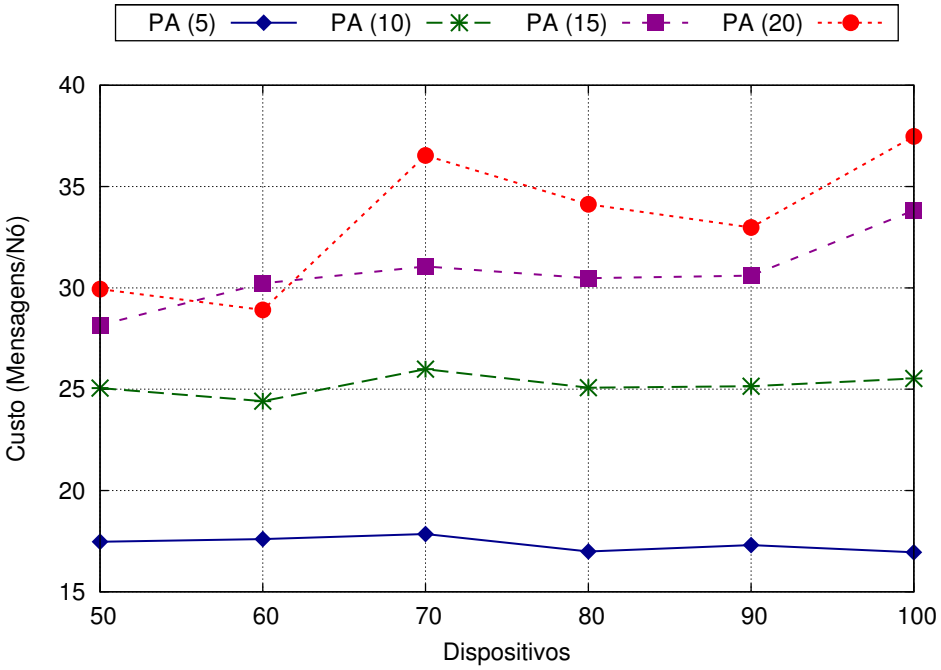


Figura 5.7: PA: Custo em mensagens recebidas por nó (C_{APN})

Todos os nós que entram na rede realizam seu primeiro contato com um nó Origem, ou seja, a quantidade de mensagens e o custo tende a ser superior sobre este nó. As Figuras 5.9(a), 5.9(b), 5.10(a) e 5.10(b) mostram os resultados obtidos para o nó Origem em relação à quantidade de mensagens recebidas, usando critérios de vizinhança Aleatória e por PA, e o custo por cada nó na lista de vizinhos para os casos de vizinhança Aleatória e por PA, respectivamente.

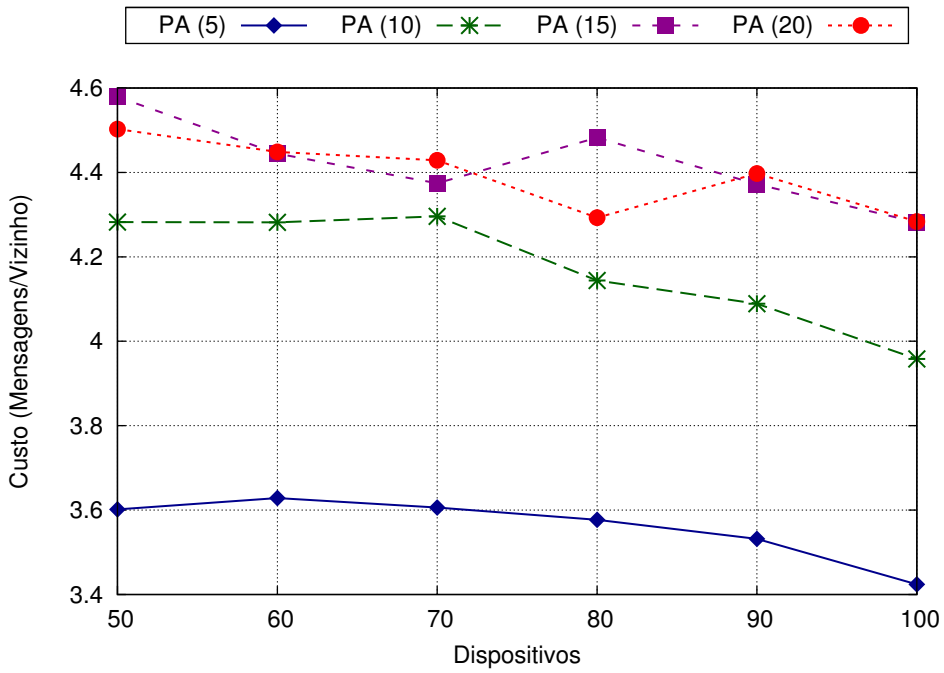


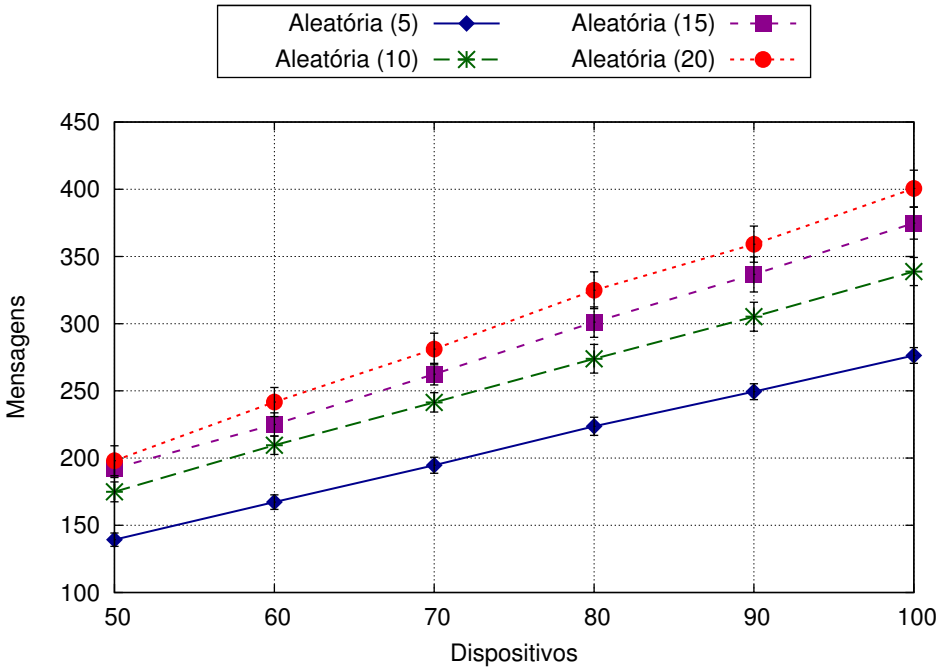
Figura 5.8: PA: Custo em mensagens recebidas para descobrir um novo vizinho (C_{APV})

Em todos os casos, a quantidade de mensagens cresce linearmente com o número de dispositivos na rede, sendo que o caso em que o algoritmo de busca deve encontrar 5 vizinhos obteve a menor quantidade de mensagens em relação aos demais casos. Nos demais casos, as requisições de novos nós aos vizinhos pode retornar dispositivos já descobertos e conectados ao requisitante, logo para que um dispositivo consiga encontrar o número mínimo de vizinhos é necessário um número maior de mensagens.

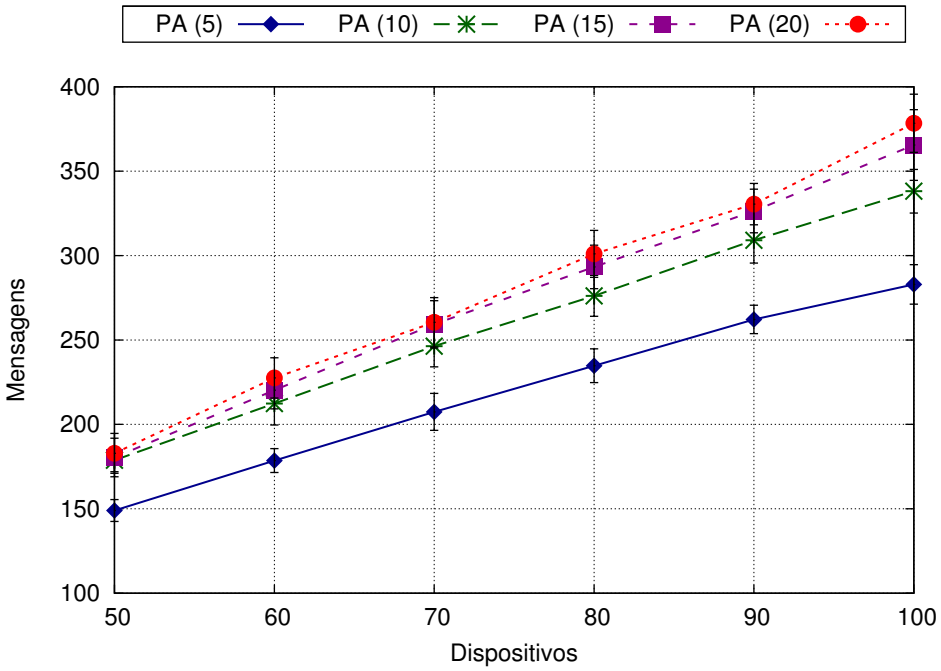
No geral, ao comparar os critérios de vizinhança, o nó Origem troca menos mensagens quando se usa o PA como critério. Pois quando um dispositivo recebe um vizinho já presente em sua lista, o algoritmo de busca conclui a busca e reinicia no próximo processo de manutenção da rede. Como o algoritmo para, a troca de mensagens também para, enquanto que no caso da vizinhança aleatória essa troca possivelmente continua. Dessa forma, o custo para estabelecer a vizinhança entre os nós é maior para o caso de seleção de vizinhos aleatória do que o PA.

O custo por vizinho no nó Origem é inferior nos demais nós, pois como o nó Origem é acessível por qualquer outro na rede, não existe troca de mensagens para o estabelecimento de conexão para com ele. Com o nó Origem, somente são enviadas mensagens para o inicio da comunicação e requisição de novos vizinhos, dessa forma o custo em mensagens por vizinho tende a ser menor.

Ao utilizar o PA como critério de vizinhança tem-se menos mensagens trocadas na rede e como isso se tem um custo menor em mensagens como pode ser observado ao comparar as figuras 5.10(a) e 5.10(b), apesar de o caso *Aleatória*(5) ter o menor



(a) Vizinhança Aleatória

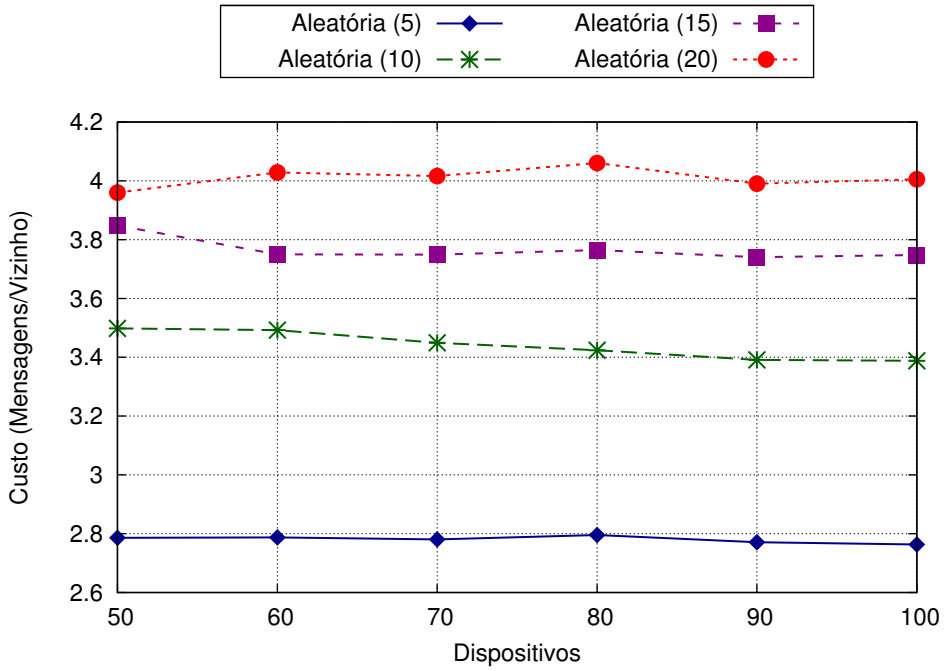


(b) Vizinhança PA

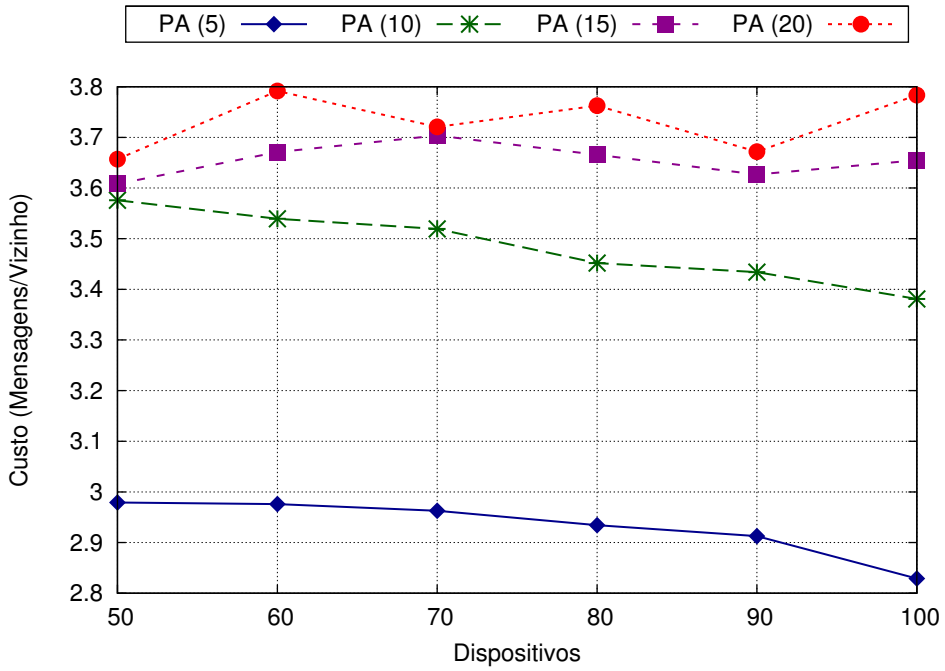
Figura 5.9: Origem: Quantidade média de mensagens de controle totais recebidas (MC). (a) Aleatória (b) PA

custo. Isso ocorre pelo fato já comentado, onde utilizar o critério por PA faz com que um nó envie mais mensagens para descobrir novos vizinhos pois a requisição de um novo nó a um vizinho já requisitado pode retornar um nó descoberto anteriormente.

A avaliação analítica para o nó Origem obteve resultados iguais para ambos os critérios de vizinhança, desde que a quantidade de vizinhos para o nó Origem, em



(a) Vizinhança Aleatória



(b) Vizinhança PA

Figura 5.10: Origem: Custo em mensagens recebidas para estabelecer a vizinhança (C_{APVO}). (a) Aleatória (b) PA

ambos os casos, é igual. Resultados da avaliação analítica na tabela 5.1.

Sabendo que a avaliação analítica contabiliza tanto mensagens enviadas quanto recebidas, vemos que os resultados apresentados na tabela 5.1 é o dobro do limite a ser obtido nas simulações. Ao comparar as figuras 5.9(a) e 5.9(b) com os valores na tabela 5.1, vemos que os resultados obtidos na simulação são bem conservadores,

Tabela 5.1: Avaliação Analítica: Nó Origem

Nós	**Valores**
50	695
60	835
70	975
80	1115
90	1255
100	1395

mesmo sendo a metade do contabilizado na avaliação analítica.

Análise

Baseado nestes resultados preliminares concluímos que o algoritmo consegue estabelecer a vizinhança entre os dispositivos na rede. Verificamos que o custo para descoberta de cada vizinho é baixa, na média não ultrapassou a 5 mensagens recebidas por vizinho no pior caso.

Essa avaliação preliminar contou com 100 segundos de simulação, verificamos que este tempo foi o suficiente para realizar a formação da vizinhança nos dispositivos. Como descrito anteriormente, neste caso cada dispositivo executa o processo de descoberta de vizinhos por duas vezes: primeiro ao entrar na rede se comunicando com o nó Origem; e depois no primeiro processo de renovação das sessões NAT, onde é feita a requisição de novos vizinhos para os atuais nós presentes na lista, caso seja necessário. Neste caso, a saída de algum dispositivo não causa impacto sobre a rede, pois com o tempo os nós conseguem estabelecer novas conexões com outros dispositivos.

Nesta avaliação concluímos que a quantidade de mensagens sobre o nó Origem cresce linearmente com o número de dispositivos na rede. Isso já é uma indicação da escalabilidade do sistema, pois os nós ao entrarem na rede se comunicam primeiramente com nó Origem para descobrir um vizinho e enviam mensagens de tempos em tempos (T_{NAT}) para manter as sessões NAT, ou seja, mesmo com a troca de mensagens realizada com todos os nós ao entrarem na rede ou as mensagens de renovação de sessão NAT o custo tende a ter uma baixa taxa de crescimento ao se aumentar a quantidade de nós.

Os resultados obtidos na simulação foram conservadores em relação à avaliação analítica apresentada anteriormente. As equações apresentadas na avaliação analítica são um limite superior para os valores reais de troca de mensagens.

5.2.2 Avaliação REPI-Internet

Nesta avaliação utilizamos o cenário Rede Rio definido na seção 4.5.2. A REPI foi simulada 20 vezes com 1024, 4086 e 10240 (máximo de dispositivos simulados em tempo hábil) e avaliamos o impacto do critério de vizinhança na entrega das mensagens REPI.

Apresentamos resultados da simulação com duração de 300 segundos onde as mensagens REPI são enviadas aos 150 segundos. Este tempo foi escolhido sabendo que com 100 segundos a rede já está formada, como foi mostrado nos resultados preliminares, logo aos 150 segundos enviamos uma mensagem REPI para um grupo de interesse. Os 150 segundos restantes é o tempo que aguardamos para que todas as mensagens na rede sejam trocadas.

Os resultados são valores médios calculados a partir de 20 rodadas de cada simulação e obtivemos valores com nível de confiança de 95% e erro menor que 5%. Para o caso de 10240 nós contamos apenas com 10 rodadas de simulação devido a grande quantidade de tempo necessário para sua execução (aproximadamente 30 horas para cada rodada). Neste último caso, nas simulações obtivemos valores com nível de confiança de 95% e erro inferior a 8%.

Os resultados são apenas apresentadas as mensagens de interesse. O custo de formação da rede sobreposta não está sendo contabilizado nestes casos, pois já o avaliamos nos resultados anteriores apresentados.

REPI com 1024 **nós**

Neste conjunto de experimentos variamos o tamanho do grupo de interesse em aproximadamente 5% (50), 10% (100), 15% (150) e 20% (190) dos nós simulados. Avaliamos o impacto do critério de vizinhança, variando os dois casos implementados (*Aleatória* e PA). O algoritmo de busca de vizinhos foi parametrizado para buscar no mínimo 5 ou 10 vizinhos.

A Figura 5.11 apresenta a taxa de entrega superior a 99% em todos os casos avaliados. Independente da quantidade de nós interessados na mensagem, praticamente todos eles receberam a mensagem tanto utilizando os critérios *Aleatória* quanto PA.

Essa alta taxa de entrega só é possível graças à alta colaboração da rede, pois os nós não interessados contribuem com a rede encaminhando as mensagens, fato que gera vários caminhos na rede, permitindo que as mensagens cheguem a seus destinos.

A Figura 5.12 apresenta o nível de colaboração da rede. Em todos os casos a rede contou com mais de 50% dos nós colaborando com a rede. A colaboração tende a diminuir conforme a quantidade de nós interessados aumenta, isto ocorre pelo fato

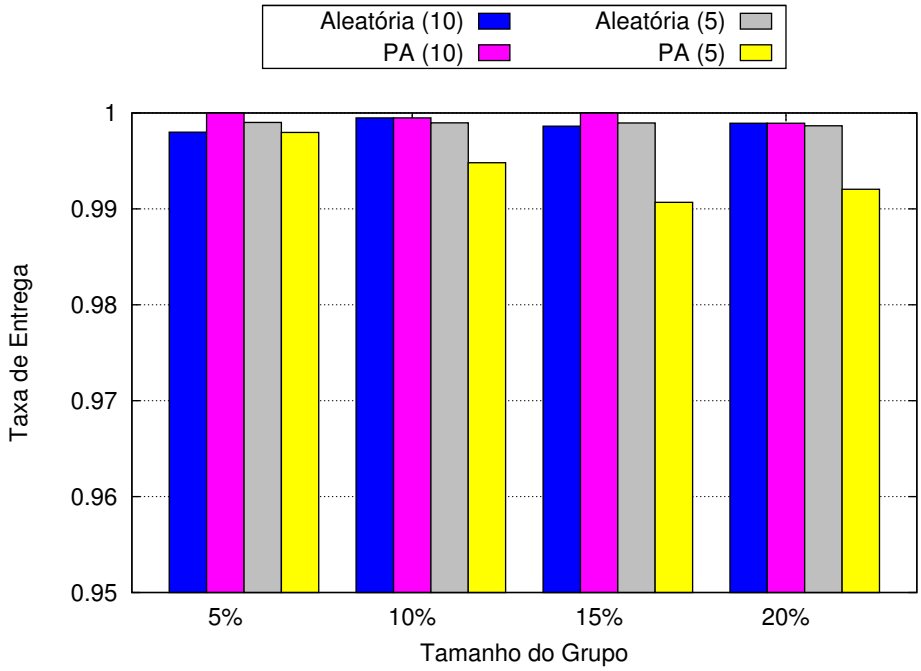


Figura 5.11: REPI-1024: Taxa de entrega (T_E) x Percentual dos 1024 nós da rede (Grupo de interesse)

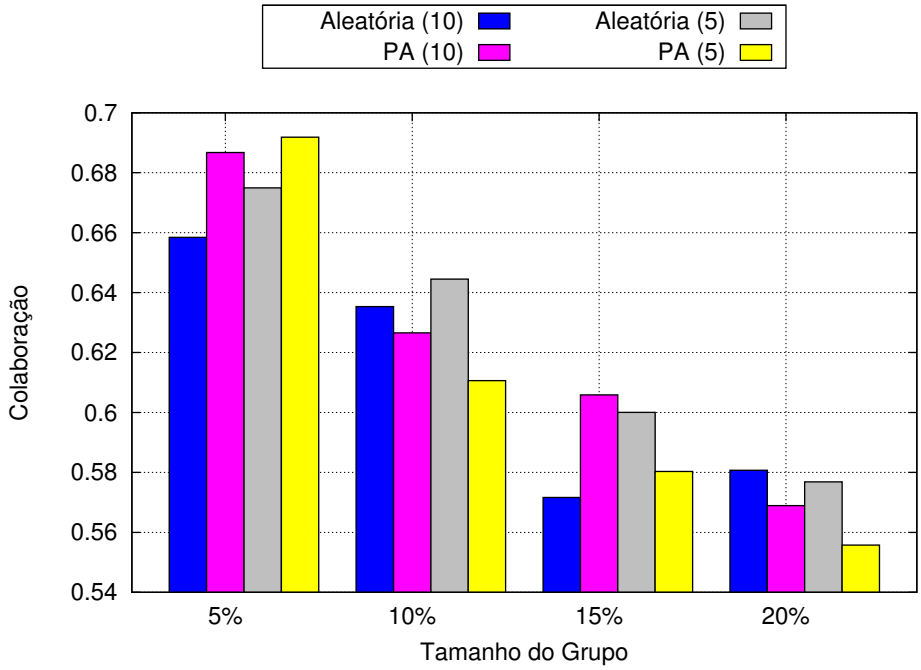


Figura 5.12: REPI-1024: Taxa de colaboração da rede (T_C) x Percentual dos 1024 nós da rede (Grupo de interesse)

de reduzir a quantidade de possíveis nós colaboradores[1].

Essa colaboração está diretamente ligada à quantidade de mensagens que trafe-

[1]Nós colaboradores são encaminhadores de mensagens que não lhes interessam. Outros nós interessados na mensagem podem encaminhá-la, mas não são considerados colaboradores pois estão usufruindo da rede recebendo as mensagens de seu interesse

gam na rede, quanto maior a colaboração maior é o número de mensagens na rede. A Figura 5.13 apresenta os resultados obtidos em relação à quantidade de mensagens na rede.

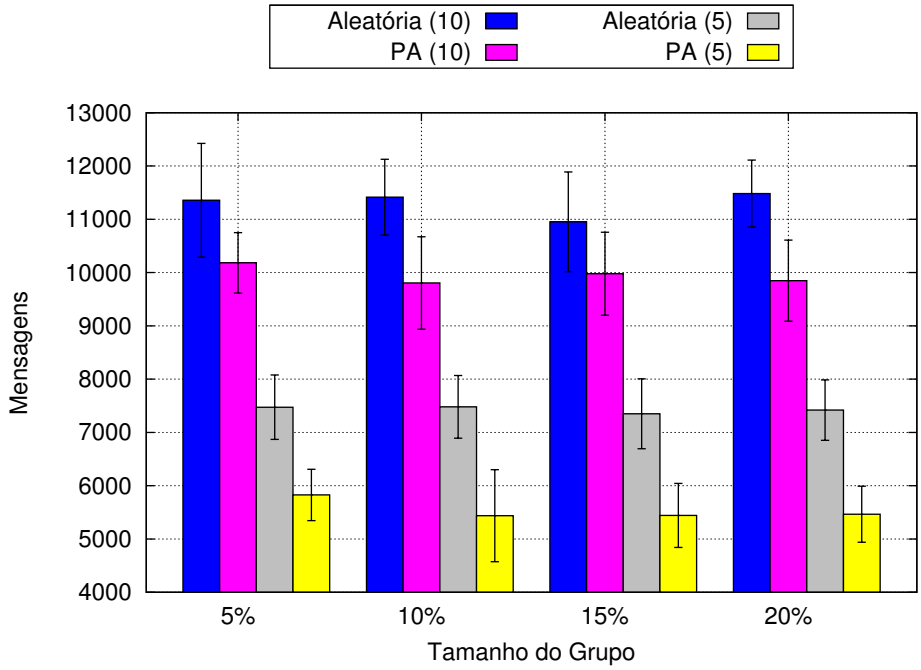


Figura 5.13: REPI-1024: Total de mensagens de interesse (MI) x Percentual dos 1024 nós da rede (Grupo de interesse)

A quantidade de mensagens foi semelhante em todos os casos de tamanhos de grupos de interesse. Isso ocorre pelo fato dessas mensagens dependerem apenas do encaminhamento na rede e não está relacionada com os interesses dos nós. A colaboração é um dos fatores que contribuem para o aumento da quantidade de mensagens, mas vale lembrar que o encaminhamento pode ocorrer em nós com interesse (ou não colaboradores).

Comparando a quantidade de mensagens entre os casos avaliados, é visível que ao utilizar critério de vizinhança PA, obtém-se uma quantidade menor de mensagens na rede, este fato está relacionado com a quantidade de vizinhos em cada nó, pois com este critério de vizinhança consegue-se descobrir menos vizinhos que o critério *Aleatória* como mostra a Figura 5.14.

Ao comparar os critérios de vizinhança, vemos uma diferença de 7 vizinhos entre o critério *Aleatória* e PA em todos os casos. Isso ocorre pois ao se escolher um vizinho aleatoriamente a chance de selecionar um nó escolhido previamente é bem menor do que se utilizar o PA. Pois no algoritmo para seleção de vizinhos baseado no PA, um nó escolhe em sua lista um vizinho cuja quantidade de campos do PA seja maior. Caso a lista de um nó não seja modificada em um intervalo de tempo, o nó selecionado baseado no PA neste intervalo sempre será o mesmo.

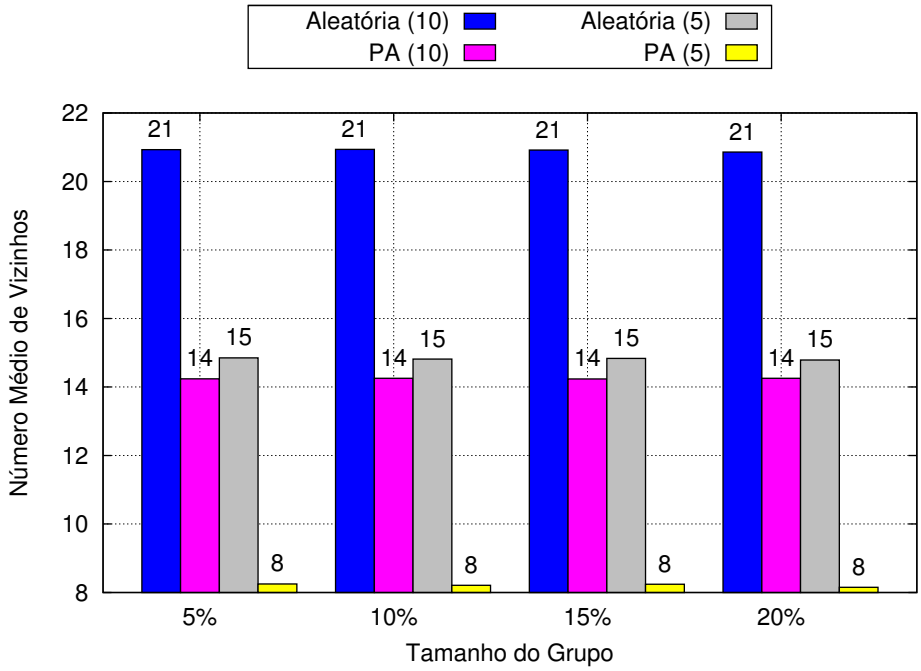


Figura 5.14: REPI-1024: Quantidade média de vizinhos (NV) x Percentual dos 1024 nós da rede (Grupo de interesse)

A quantidade de vizinhos em cada nó é outro fator principal na quantidade de mensagens na rede, pois quando um nó encaminha uma mensagem ele envia uma mensagem para cada vizinho, assim sendo quanto mais vizinhos, mais mensagens são enviadas na rede.

Baseado na quantidade de mensagens, calculamos o custo em mensagens por nó interessado (C_{PNI}). A Figura 5.15 apresenta estes resultados. Com a quantidade de mensagens não depende da quantidade de nós interessados, o C_{PNI} diminui conforme aumenta a quantidade de interessados. Isto mostra, neste caso, que mensagens com interesses populares são menos custosas para a rede. No caso, quando o grupo de interesse é composto por 20% dos nós, o C_{PNI} fica em menos de 60 mensagens chegando a 25 no melhor caso.

A Figura 5.16 apresenta o custo em mensagens por nó (C_{RPN}). Este resultado não depende do tamanho dos grupos de interesse, por isso os resultados foram semelhantes em todos os casos de tamanhos de grupos. Esta métrica mostra que foi necessário que cada nó encaminhasse em média 11 mensagens no pior caso e menos de 6 no melhor caso, para entregar mais de 99% aos nós interessados.

As mensagens de interesses foram entregues, no pior caso ($Aleatória(10)$), em média menos de $140ms$ a todos os dispositivos, como mostra a Figura 5.17. O caso $PA(5)$ foi o melhor, onde as mensagens foram entregues em torno de $60ms$.

O tempo de entrega das mensagens tende a ser maior quando se tem uma quantidade de vizinhos maior, pois como as mensagens são encaminhas individualmente

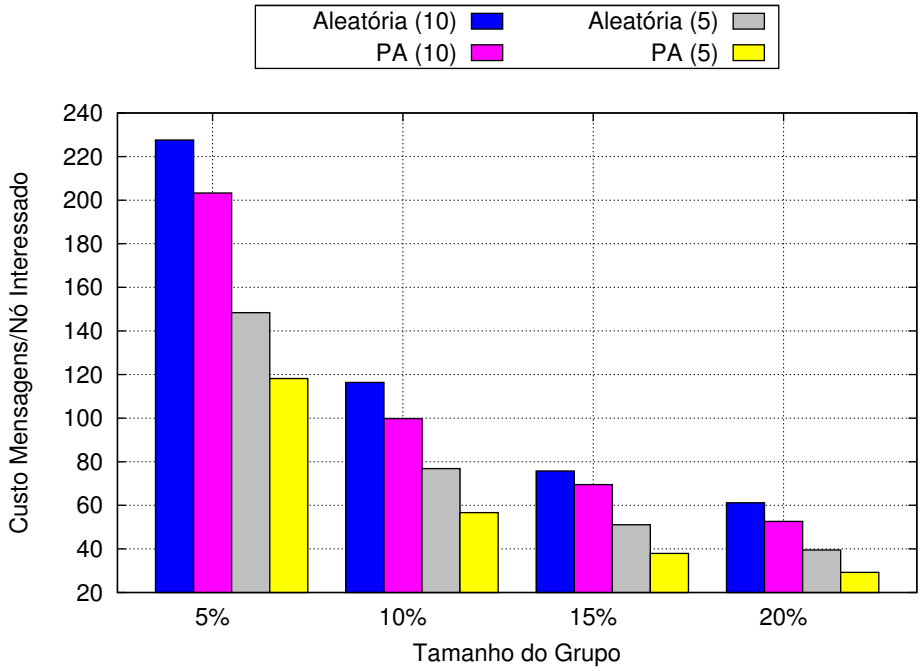


Figura 5.15: REPI-1024: Custo em mensagens por nó interessado (C_{PNI}) x Percentual dos 1024 nós da rede (Grupo de interesse)

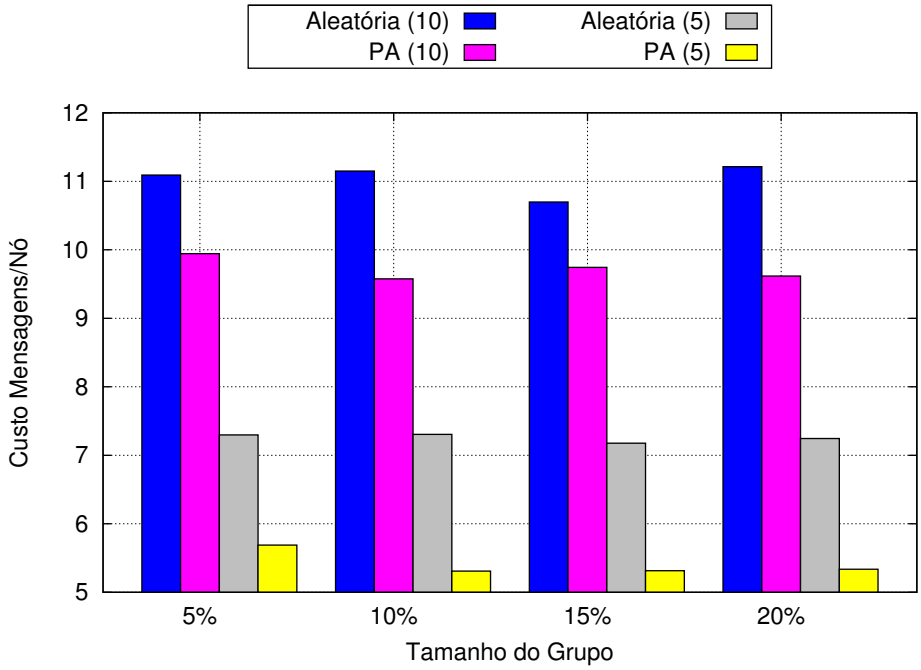


Figura 5.16: REPI-1024: Custo em mensagens por nó (C_{RPN}) x Percentual dos 1024 nós da rede (Grupo de interesse)

para cada vizinho, as mensagens enviadas aos últimos vizinhos serão penalizadas. No caso $PA(5)$ a quantidade de vizinhos foi inferior aos demais e com isso obteve-se uma latência inferior.

Os tempos de entrega encontrados são valores médios de entrega de uma mensa-

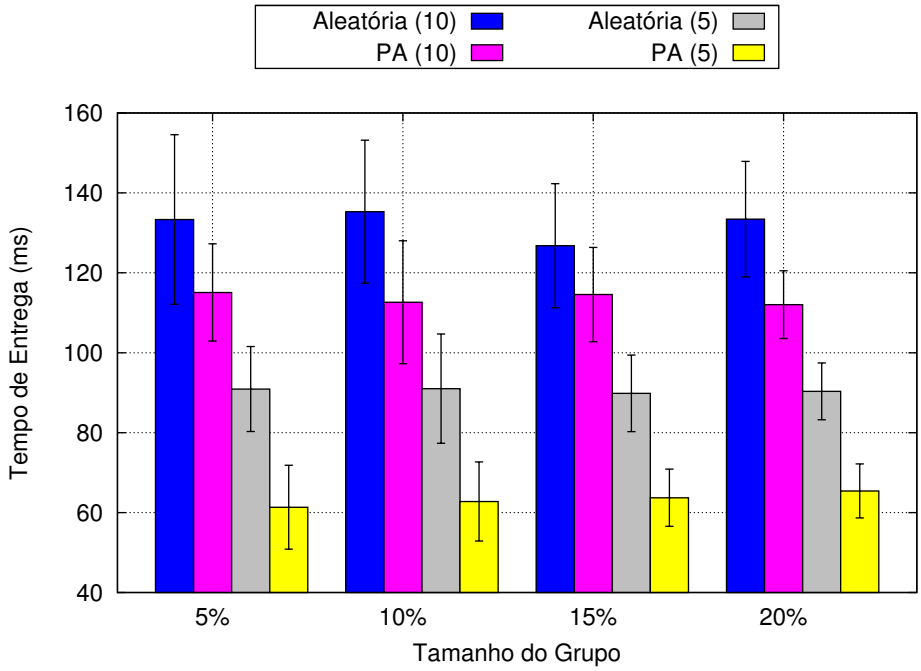


Figura 5.17: REPI-1024: Tempo médio para entrega das mensagens (T) x Percentual dos 1024 nós da rede (Grupo de interesse)

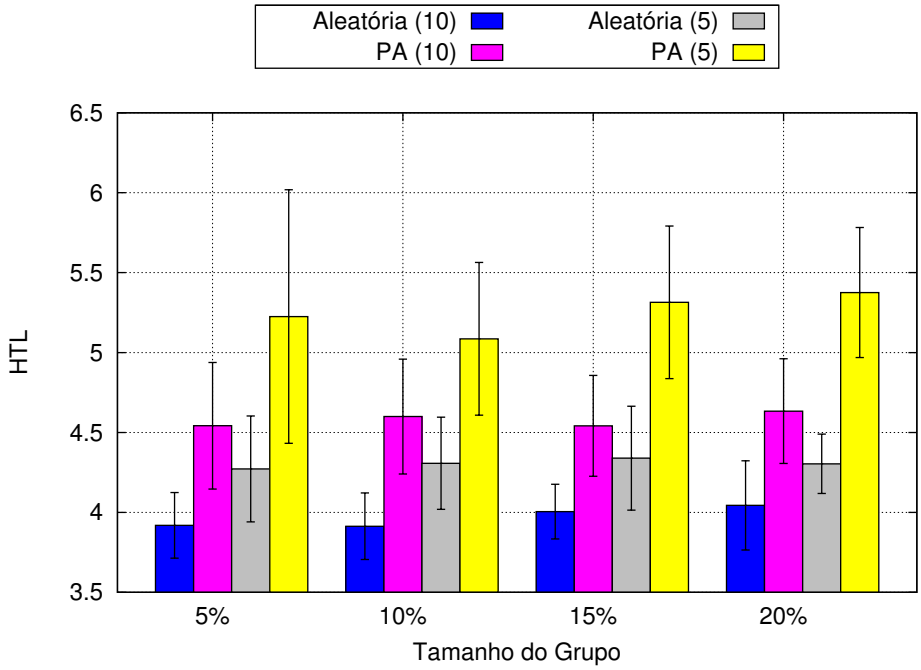


Figura 5.18: REPI-1024: Número médio de saltos (HTL) x Percentual dos 1024 nós da rede (Grupo de interesse)

gem de interesse a todos os membros do grupo, desde que estes resultados são para entrega superior a 99% dos destinatários.

A Figura 5.18 apresenta a quantidade média de saltos para entregar as mensagens de interesse. Pode-se observar que casos onde a quantidade de vizinhos é maior

(*Aleatória*(10)), a quantidade de saltos foi inferior enquanto que houve mais saltos para os casos com menos vizinhos ($PA(5)$). Era esperado que isto ocorresse, pois para atingir uma quantidade maior de nós é necessário haver mais encaminhamento de mensagens o que aumenta a quantidade de saltos.

No caso $PA(5)$ a quantidade de saltos é ligeiramente maior, mas o tempo de entrega é menor que os casos. Isso mostra claramente que ter muitos vizinhos provoca um atraso muito grande na entrega das mensagens, como discutido anteriormente.

REPI com 4096 nós

Neste conjunto de experimentos variamos o tamanho do grupo de interesse em aproximadamente 5% (204), 10% (409), 15% (600) e 20% (760) dos nós simulados. Avaliamos o impacto do critério de vizinhança, variando os dois casos implementados (*Aleatória* e PA). O algoritmo de busca de vizinhos foi parametrizado para buscar no mínimo 5, 10 e 20 vizinhos. Diferente do caso anterior, avaliamos o impacto de se buscar 20 no intuito de verificar se o custo seria muito superior aos outros casos, neste ambiente com um número superior de nós.

O primeiro resultado apresentado é a taxa de entrega que o protocolo REPI consegue atingir nas condições avaliadas. A Figura 5.19 mostra que em todos os casos a entrega foi superior a 99% dos nós, ou seja, praticamente todos os dispositivos dos grupos receberam as mensagens destinadas a eles.

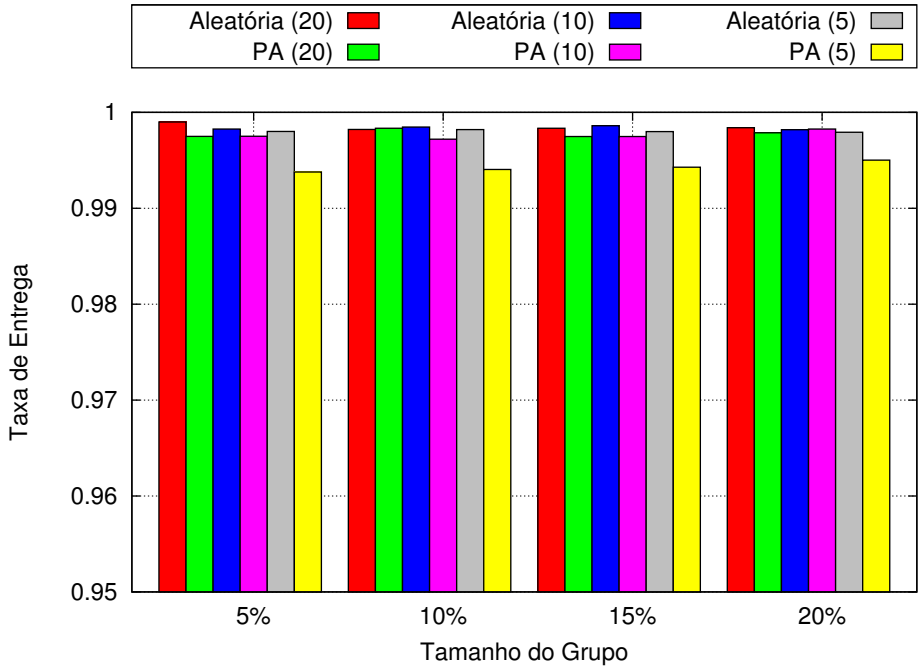


Figura 5.19: REPI-4096: Taxa de entrega (T_E) x Percentual dos 4096 nós da rede (Grupo de interesse)

Para conseguir esta entrega de mais de 99% foi necessário a colaboração dos

dispositivos na rede. Tais dispositivos colaboradores encaminham as mensagens sem terem interesses nelas. Como mostra a Figura 5.20, a colaboração reduz com o aumento do tamanho dos grupos de interesse, isso ocorre pois existe uma quantidade maior de nós interessados nas mensagens, diminuindo a quantidade de possíveis colaboradores.

Quanto maior a colaboração maior é a quantidade de mensagens trafegando na rede, pois mais nós estão a encaminhá-la. Os resultados referentes à colaboração (Figura 5.20) mostram que mais da metade dos dispositivos na rede colaboraram para a entrega das mensagens na rede e devido a essa colaboração mais de 99% dos dispositivos receberam as mensagens de grupo.

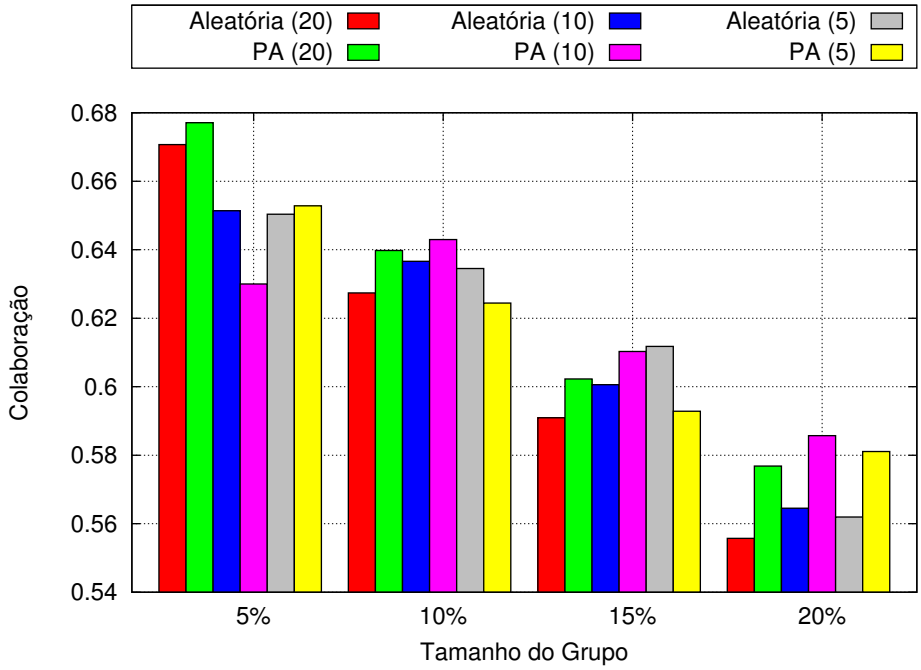


Figura 5.20: REPI-4096: Taxa de colaboração da rede (T_C) x Percentual dos 4096 nós da rede (Grupo de interesse)

O mecanismo probabilístico utilizado pela REPI para disseminar as mensagens faz com que uma mensagem enviada se multiplique na rede percorrendo diversos caminhos distintos com o intuito de atingir o máximo de dispositivos interessados. A Figura 5.21 mostra a quantidade total de mensagens recebidas na rede. Podemos observar que esta quantidade independe do tamanho dos grupos interessados, pois a quantidade de mensagens na rede não depende de quantos estão interessados nela, mas quantos a encaminharam.

A colaboração é um dos fatores principais na quantidade de mensagens na rede, pois quanto maior a colaboração mais mensagens são encaminhadas na rede, como mostra os resultados apresentados em 5.21 e 5.22. A colaboração no caso *Aleatória*(20) foi ligeiramente inferior ao $PA(20)$ e este comportamento também

ocorre na quantidade de mensagens na rede.

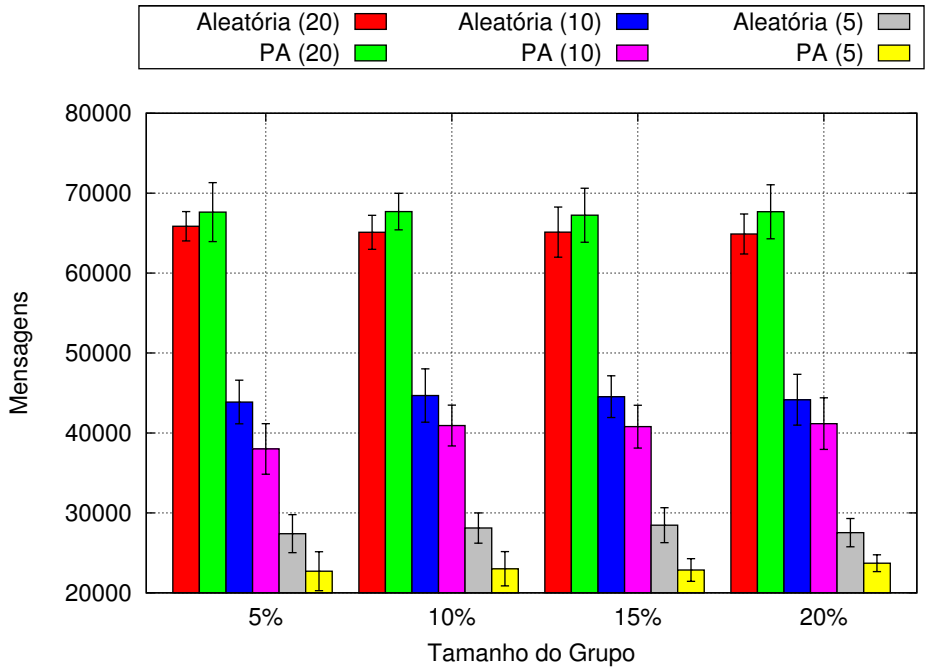


Figura 5.21: REPI-4096: Total de mensagens de interesse (MI) x Percentual dos 4096 nós da rede (Grupo de interesse)

Nos casos $Aleatória(10)$ e $PA(10)$, a colaboração juntamente com a quantidade média de vizinhos em cada nó faz com que a quantidade de mensagens seja superior como pode ser observado ao se comparar as figuras 5.21 e 5.22. Pois cada nó, ao encaminhar uma mensagem, envia para todos os vizinhos em sua lista, assim sendo quanto maior for a quantidade de vizinhos maior será a quantidade de mensagens a serem enviadas quando o encaminhamento ocorre.

Os casos com menor quantidade de mensagens enviadas dentre os avaliados são $Aleatória(5)$ e $PA(5)$, pois se consegue uma alta taxa de entrega com mensagens em torno de 25000 mensagens. Neste caso a quantidade de mensagens foi menor pois a quantidade de vizinhos descobertos foi menor que os outros casos. Assim sendo uma mensagem ao ser encaminhada será enviada para menos nós que nos outros casos.

Avaliamos o custo em mensagens por nó interessado (C_{PNI}), a Figura 5.23 apresenta este resultado. Este custo é a quantidade de mensagens geradas na rede para poder entregar aos nós interessados a mensagem desejada. O resultado mostra que quanto maior a quantidade de interessados na mensagem menor o custo, ou seja, interesses muito populares custa muito pouco para rede enquanto que interesses menos populares tem um custo maior para realizar a entregar das mensagens. Como mostra os resultados, o custo chega a mais de 300 mensagens nos casos $Aleatória(20)$ e $PA(20)$ para um grupo de interesse com 5% dos nós, enquanto que para os mesmos casos mas com grupos de 20% dos nós este custo cai para menos de 100 mensagens.

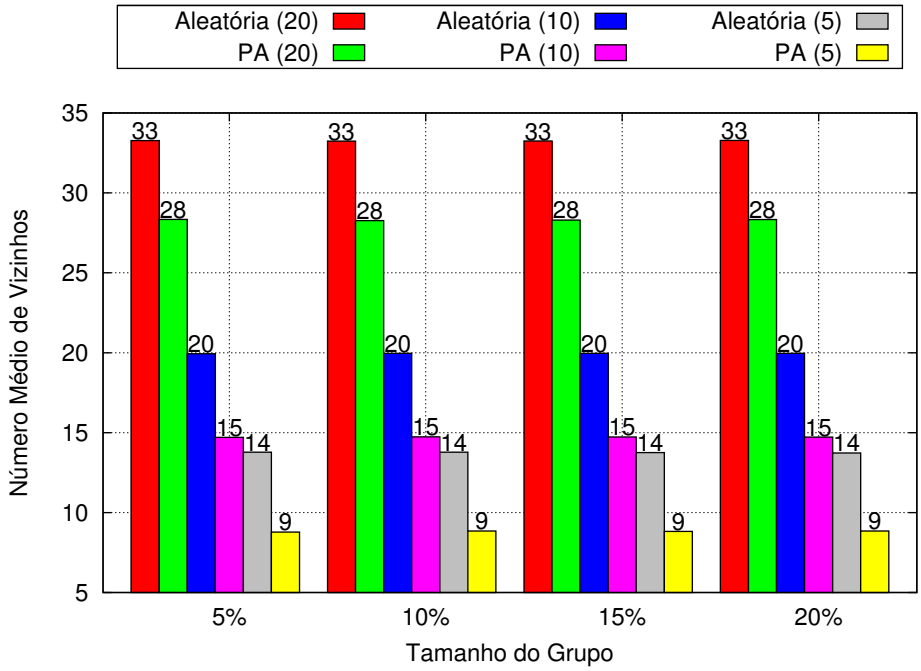


Figura 5.22: REPI-4096: Quantidade média de vizinhos (NV) x Percentual dos 4096 nós da rede (Grupo de interesse)

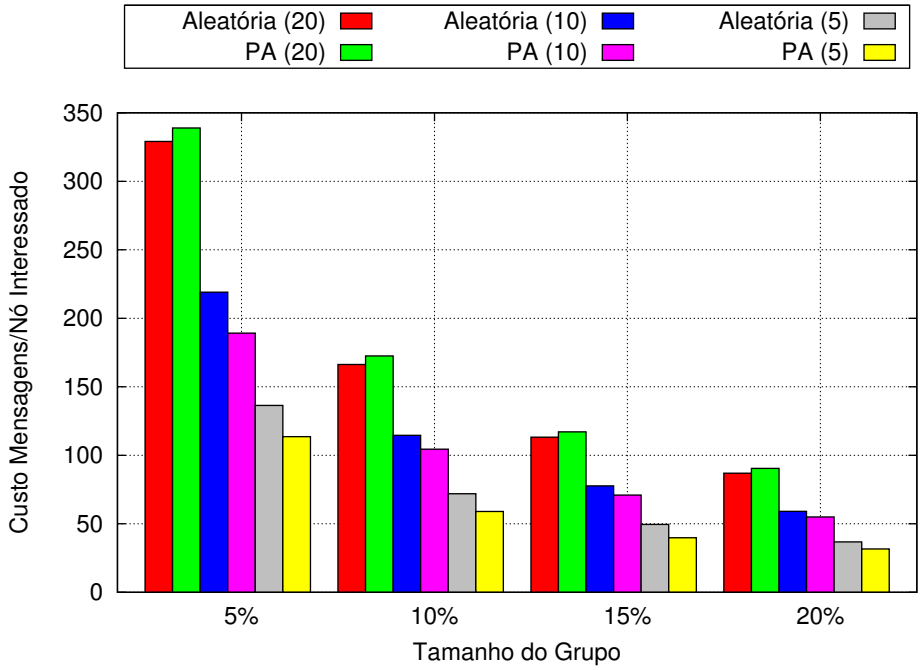


Figura 5.23: REPI-4096: Custo em mensagens por nó interessado (C_{PNI}) x Percentual dos 4096 nós da rede (Grupo de interesse)

Nos casos $Aleatória(5)$ e $PA(5)$ o custo foi bem menor chegando a ser 50% menor em relação aos casos $Aleatória(20)$ e $PA(20)$ independente do tamanho dos grupos. Isso se deve ao fato de ser gasto menos mensagens como mostrado na Figura 5.21, logo sendo gasto menos mensagens, menor é o custo. Com estes resultados, pode-

se inferir que para interesse mais populares, onde a maior parte dos nós estariam interessados, este custo seria muito inferior.

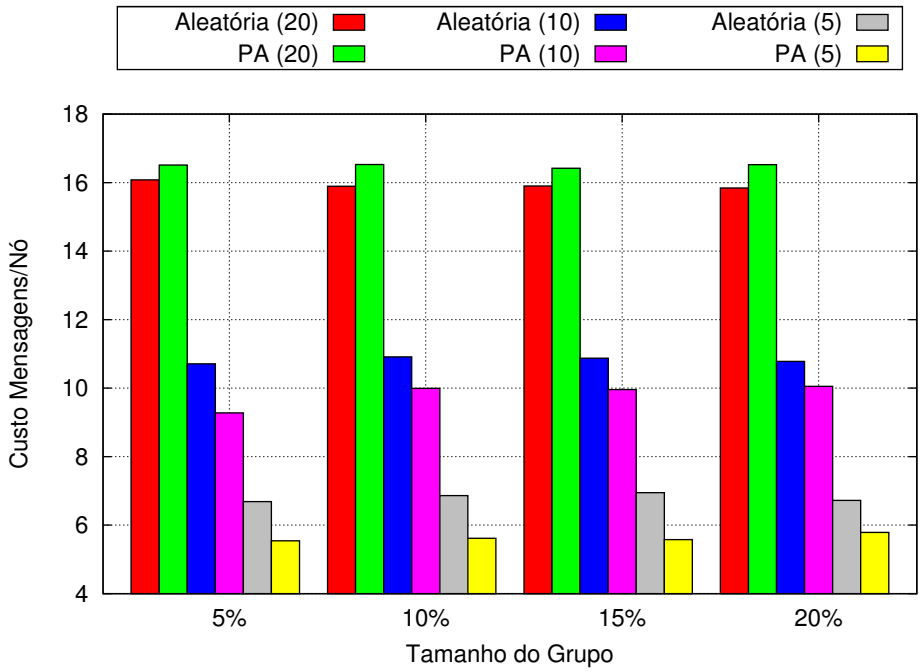


Figura 5.24: REPI-4096: Custo em mensagens por nó (C_{RPN}) x Percentual dos 4096 nós da rede (Grupo de interesse)

Apresentamos na Figura 5.24 um custo em mensagens que cada nó na rede recebeu em média para poder atingir todos os interessados (C_{RPN}). O custo por nó representa a quantidade média de mensagens que cada nó na rede recebeu. O resultado mostra que no pior caso cada nó na rede recebeu em torno de 16 mensagens (casos $Aleatória(20)$ e $PA(20)$) e em torno de 6 mensagens nos melhores casos ($Aleatória(5)$ e $PA(5)$).

O tempo de entrega das mensagens foi bem superior nos casos $Aleatória(20)$ e $PA(20)$, como mostra a Figura 5.25, devido ao fato de, nestes casos, haver uma quantidade média de vizinhos superior aos demais casos (Figura 5.22). Como dito anteriormente ter uma quantidade grande de vizinhos faz com que as mensagens, ao serem encaminhadas, sofram um atraso a cada salto, pois as mensagens são encaminhadas individualmente a cada vizinho, dessa forma as últimas mensagens devem aguardar o envio de todas as anteriores sendo penalizadas com um tempo superior às demais.

Logo, ter menos vizinhos reduz o tempo para entrega das mensagens, como mostra os resultados de $Aleatória(5)$ e $PA(5)$ que entregam em menos de $100ms$, pois a quantidade média de vizinhos é inferior a 15 nós.

A Figura 5.26 apresenta a quantidade média de saltos para entrega das mensagens de interesse. Este resultado confirma a afirmação de que a quantidade maior

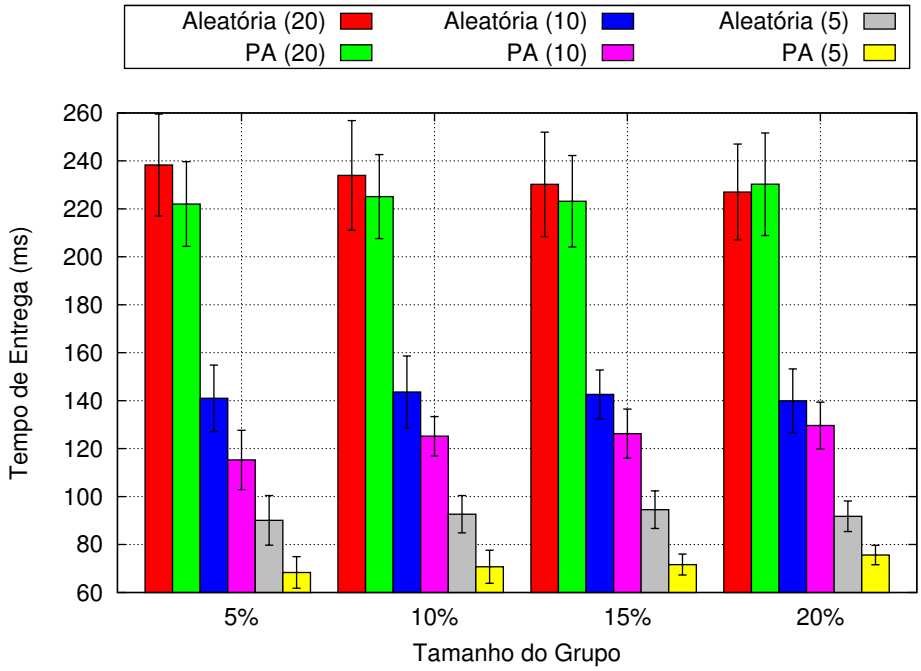


Figura 5.25: REPI-4096: Tempo médio para entrega das mensagens (T) x Percentual dos 4096 nós da rede (Grupo de interesse)

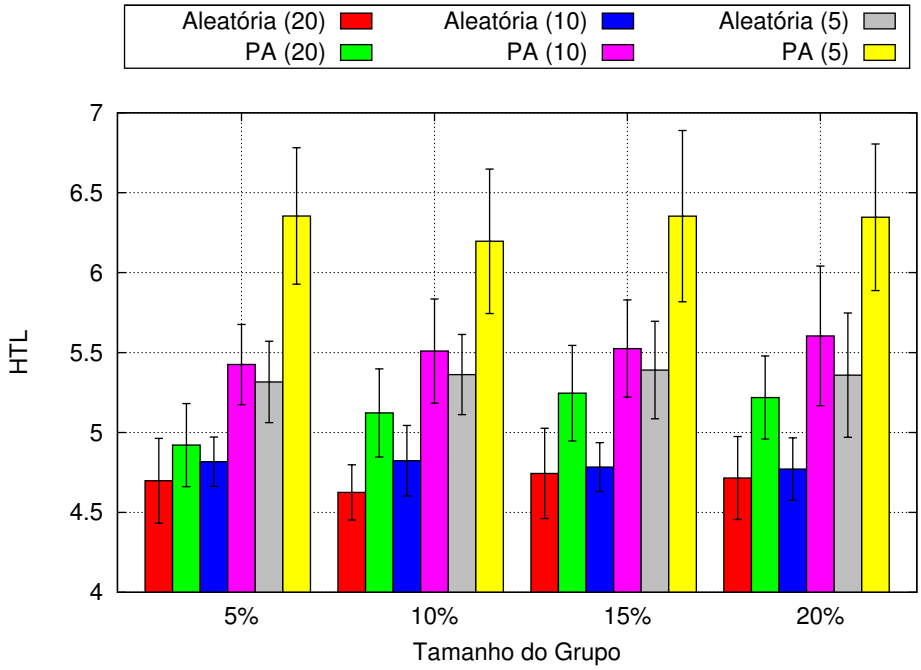


Figura 5.26: REPI-4096: Número médio de saltos (HTL) x Percentual dos 4096 nós da rede (Grupo de interesse)

de vizinhos tem grande impacto no tempo de entrega das mensagens, pois o caso $PA(5)$ obteve o melhor tempo de entrega (inferior a $100ms$) ao passo que foi necessário uma quantidade maior de salto para entrega das mensagens, em média um salto a mais que os outros casos.

REPI com 10240 nós

Baseado nos resultados apresentados anteriormente, realizamos um conjunto de testes reduzido com 10240 nós simulados. Nesta avaliação almejamos verificar a taxa de crescimento da quantidade de mensagens trocadas na rede observando os resultados obtidos anteriormente. Para isso realizou-se testes utilizando apenas o critério de vizinhança aleatória, pois foi o algoritmo com maior taxa de entrega nos casos anteriores. O grupo composto de aproximadamente 5% dos nós (500 nós) interessados na mensagem foi selecionado aleatoriamente. Nos casos anteriores os melhores resultados relativos a custo em mensagens foi para o caso onde o algoritmo de busca no mínimo 5 vizinhos, realizamos esta avaliação com este mesmo parâmetro. Os resultados estão dispostos na tabela 5.2.

Tabela 5.2: Resultados 10240 nós simulados

	Valores	**Desvio**
Número de Vizinhos (NV)	10,575	$\pm 0,087$
Taxa de Colaboração (T_C)	0,657	-
Taxa de Entrega (T_E)	0,963	-
Custo Mensagens por nó (C_{RPN})	5,756	-
Custo Mensagens por nó interessado (C_{PNI})	122,23	-
Total de Mensagens de interesse (MI)	58941,8	$\pm 5313,82$
Saltos (HTL)	7,06	$\pm 0,64$
Tempo de Entrega (T)	80,277 ms	$\pm 8,024$

Este resultado foi uma média de 10 execuções da simulação para os parâmetros apresentados anteriormente.

Algoritmo de Inundação

Avaliamos o algoritmo de inundação sobre a rede sobreposta criada. Este algoritmo de disseminação de mensagens é o limite superior para a REPI, avaliamos simplesmente para mostrar que a REPI consegue atingir uma taxa de entrega de mensagens trocando uma quantidade inferior de mensagens e com um tempo de entrega também inferior.

Os resultados apresentados para a inundação são apenas utilizando o critério de vizinhança aleatória, pois nesse caso não existe PA. Foram utilizados 4096 nós simulados. As simulações foram realizadas utilizando grupos de interesses consistindo de, aproximadamente, 5% (204), 10% (409), 15% (600) e 20% (760) dos nós simulados, como no caso anterior.

Como esperado, inundar a rede com mensagens obtém-se uma entrega superior a 99%, como ilustra a Figura 5.27. Em todos os casos as mensagens foram entregue a

quase todos os nós, pois as mensagens tendem a passar por todos os nós inundando a rede.

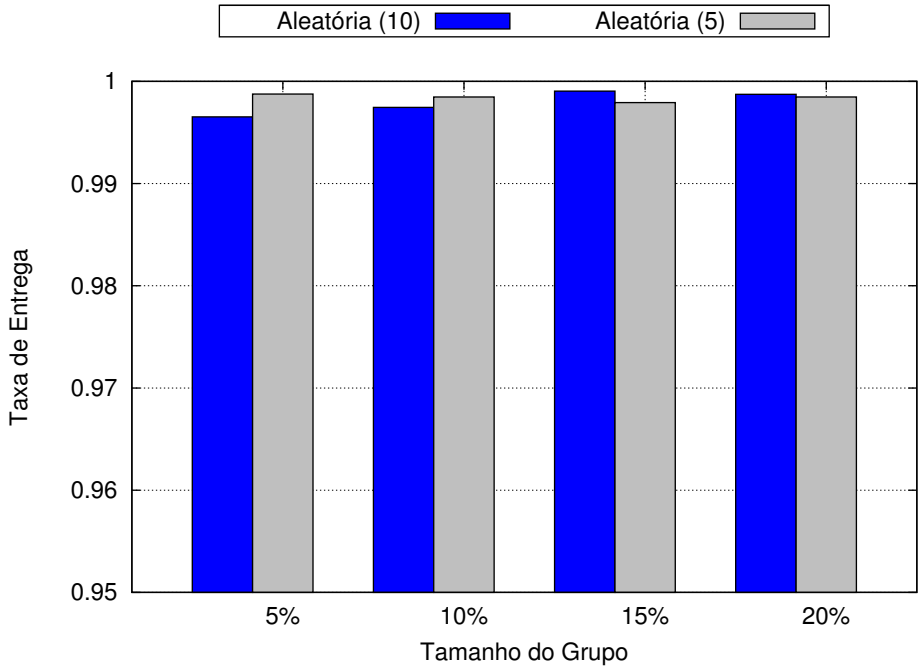


Figura 5.27: Inundação-4096: Taxa de entrega (T_E) x Percentual dos 4096 nós da rede (Destinatários)

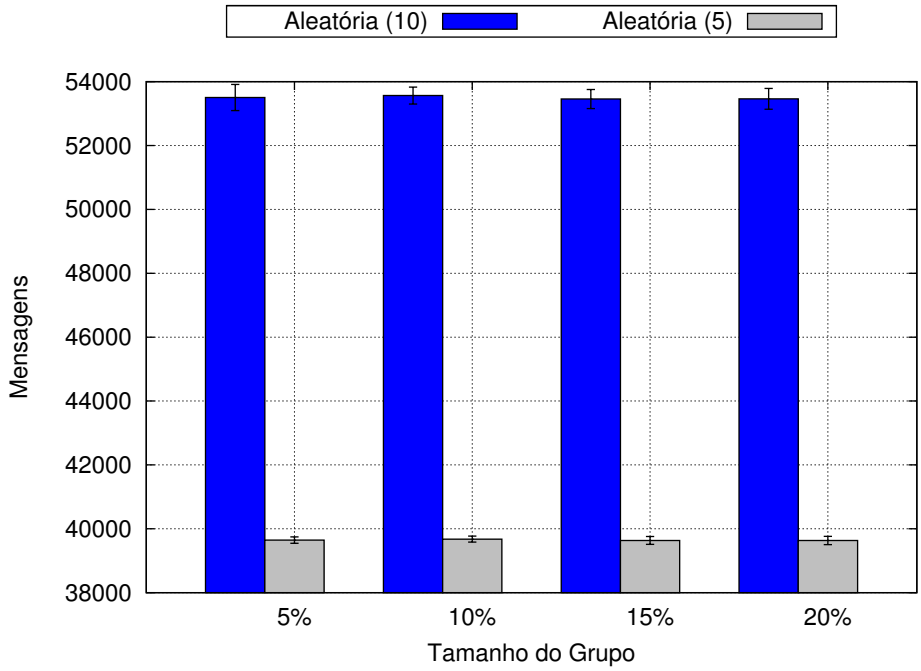


Figura 5.28: Inundação-4096: Total de mensagens x Percentual dos 4096 nós da rede (Destinatários)

Por esse motivo, a quantidade de mensagens na rede é bem alta, chegando próximo dos 54000 no caso *Aleatória*(10) e 40000 no caso *Aleatória*(5), indepen-

dente da quantidade de nós interessados. Como nos casos anteriores, a quantidade de mensagens está diretamente ligada à quantidade de vizinhos, quanto mais vizinhos mais mensagens trocadas na rede (5.28 e 5.29).

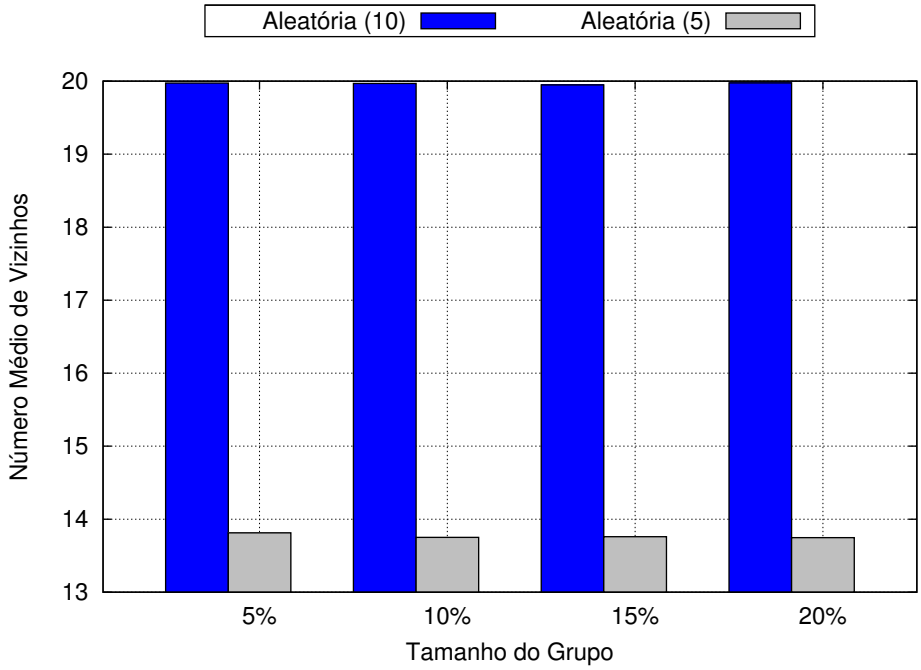


Figura 5.29: Inundação-4096: Quantidade média de vizinhos (N_V) x Percentual dos 4096 nós da rede (Destinatários)

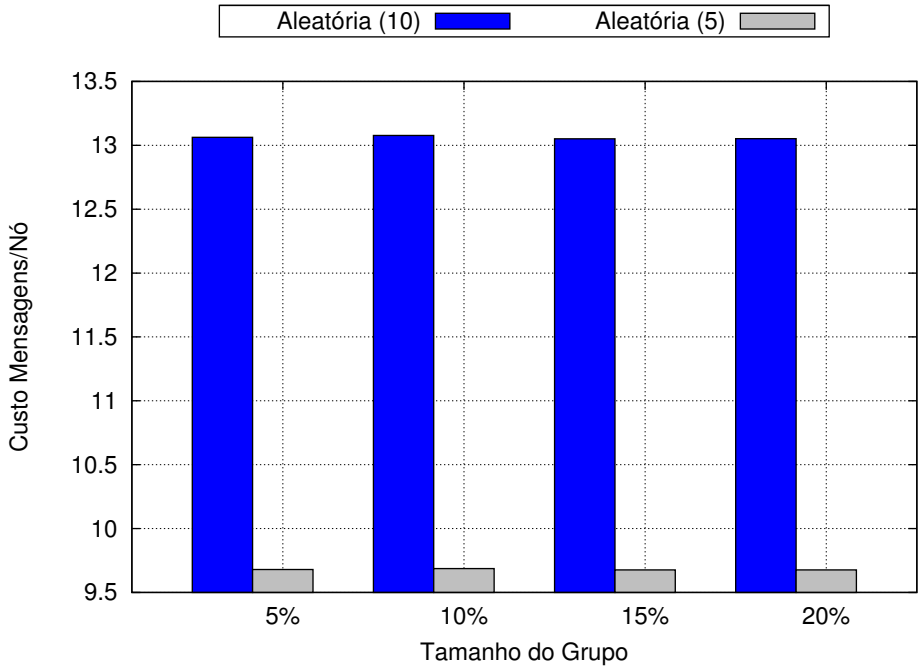


Figura 5.30: Inundação-4096: Custo em mensagens por nó (C_{PN}) x Percentual dos 4096 nós da rede (Destinatários)

A quantidade de vizinhos descoberto está relacionado com o algoritmo de busca

de vizinhos e o critério de vizinhança utilizado. Neste caso, como utilizamos apenas um critério de vizinhança na avaliação, o parâmetro para o algoritmo de busca de vizinhos é fator determinante na quantidade de vizinhos e consequentemente na quantidade de mensagens na rede. O custo em mensagens por nó foi de 13 mensagens no caso *Aleatória*(10) e de $9,6$ para o caso *Aleatória*(5). Como mostra a Figura 5.30.

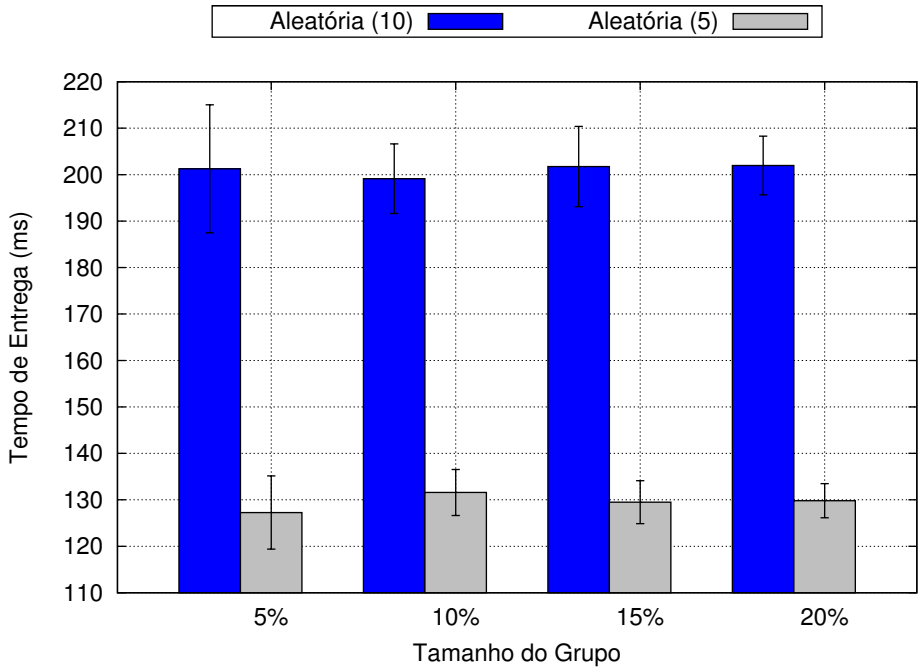


Figura 5.31: Inundação-4096: Tempo médio para entrega das mensagens (T) x Percentual dos 4096 nós da rede (Destinatários)

A Figura 5.31 apresenta os tempos para entrega de mensagens. O resultado mostra que o tempo de entrega para o caso *Aleatória*(10) foi de $200ms$ aproximadamente para todos os casos de tamanho de grupos. No caso *Aleatória*(5) a latência foi de aproximadamente $130ms$. Como descrito nos casos anteriores, a quantidade de vizinhos influencia diretamente no tempo de entrega das mensagens e mais uma vez é possível ver este fato nos resultados encontrados.

A quantidade de saltos para a entrega das mensagens foi praticamente a mesma, ficando entre 4 e 5 mensagens, como mostra a Figura 5.32. Todos os resultados não tiveram a influência da quantidade de destinatários, pois este algoritmo apenas inunda a rede com mensagens com o objetivo de atingir o máximo de nós desperdiçando recursos da rede.

Nossa expectativa é que os resultados da REPI sejam melhores que os resultados apresentados para o algoritmo de inundação. Pois diferente do algoritmo de inundação, nosso algoritmo realiza o encaminhamento das mensagens probabilisticamente. Utilizamos a mesma rede sobreposta e os mesmo parâmetros utilizados na

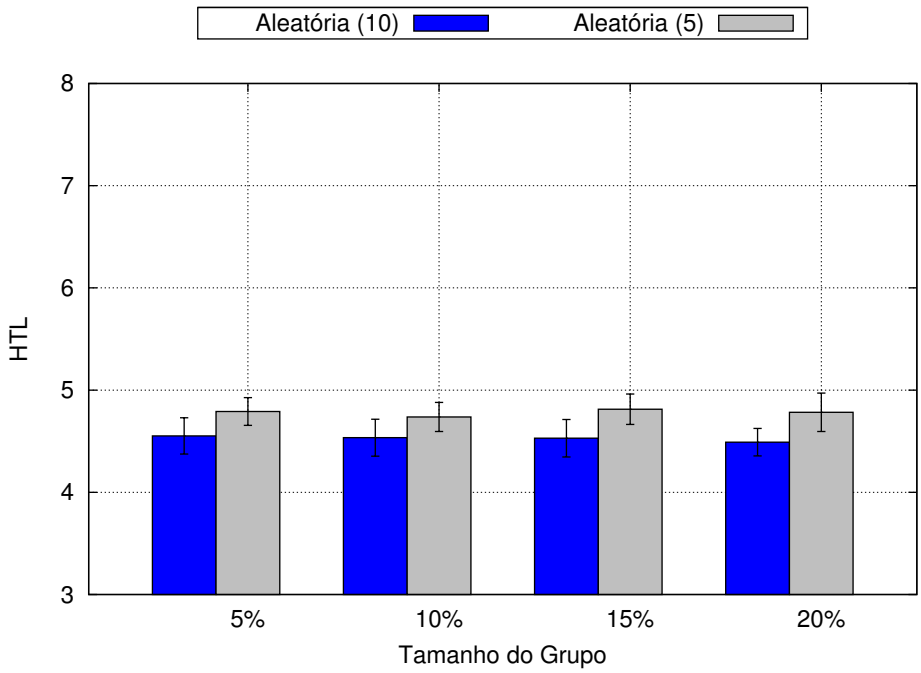


Figura 5.32: Inundação-4096: Número médio de saltos (HTL) x Percentual dos 4096 nós da rede (Destinatários)

avaliação REPI com o objetivo de avaliar apenas a essência de cada algoritmo. A seguir é feita uma análise dos resultados obtidos, apresentados nesta seção.

Análise

Nesta seção realizamos uma análise comparativa com os resultados obtidos nas simulações. Comparamos e avaliamos o comportamento da REPI nos cenários avaliados e comparamos com o algoritmo de inundação, algoritmo utilizado pelo Gnutella no processo de busca de informação na rede.

Em todos os casos avaliados a REPI consegue atingir mais de 95% dos nós interessados, chegando a mais de 99% nos casos com 1024 e 4096 nós simulados. O algoritmo de inundação, como esperado, atinge mais de 99% dos nós interessados, porém seu custo é superior gastando em torno de 2 mensagens a mais por nó nos casos avaliados (valores apresentados nas figuras 5.24 e 5.30).

Comparando o algoritmo de inundação com a REPI, nos cenários com 4096 nós simulados no caso *Aleatória*(10) vemos a diferença de aproximadamente 8000 mensagens, enquanto que no caso *Aleatória*(5) essa diferença sobe para 12000. O mecanismo probabilístico de encaminhamento de mensagens da REPI, consegue reduzir a quantidade de mensagens na rede ao passo que mantém a mesma taxa de entrega em comparação com a inundação.

O tempo de entrega das mensagens foi superior no caso da inundação sendo de $200ms$ no caso *Aleatória*(10) e $126ms$ para o caso *Aleatória*(5), enquanto que a

REPI entrega nos tempos de $140ms$ e $90ms$ respectivamente, resultados apresentados em 5.25 e 5.31. Essa redução no tempo se deve ao fato de haver menos mensagens em transito na rede, devido ao encaminhamento probabilístico, o que reduz o tempo para encaminhar as mensagens. A quantidade de mensagens na rede influencia diretamente no tempo de entrega das mensagens, devido ao fato delas ficarem em espera nos dispositivos aguardando para serem encaminhadas. Este tempo é aumentado a cada salto da mensagem.

Como o cenário é semelhante nos casos comparados da REPI e inundação, a formação da rede seguiu o mesmo comportamento atingindo um número médio de vizinhos igual em ambos os casos.

No intuito de tentar avaliar a escalabilidade da REPI, realizamos simulações com 1024, 4096 e 10240 nós. Em relação à quantidade de mensagens, comparando os três cenários no caso *Aleatória*(5), o crescimento é em taxa constante seguindo um comportamento linear, como ilustra a Figura 5.33.

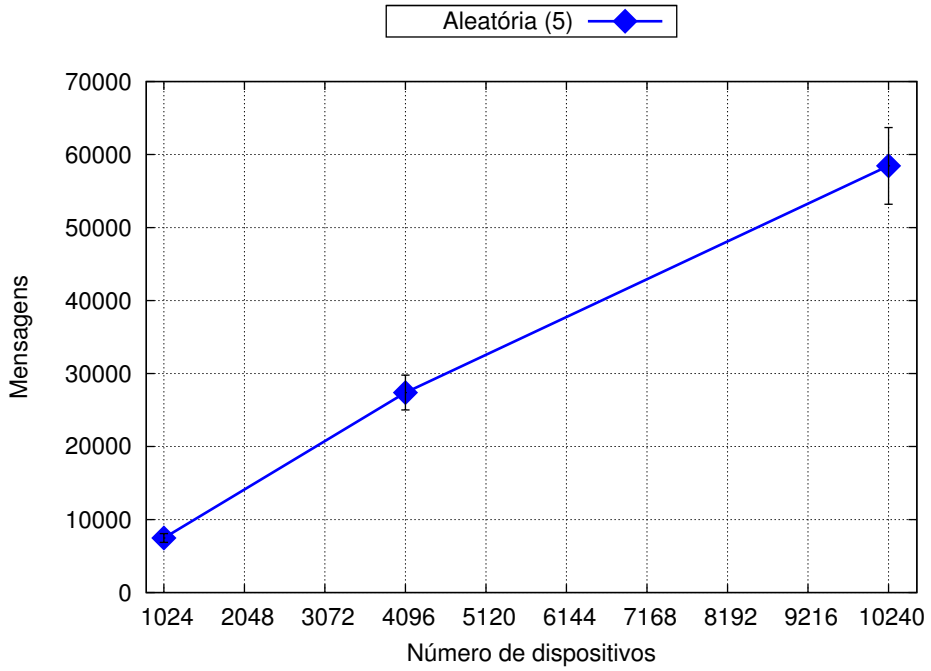


Figura 5.33: Análise: Quantidade de mensagens

Esse comportamento linear no crescimento na quantidade de mensagens sugere a potencial escalabilidade do sistema. É necessário um estudo mais detalhado com REPI maiores para assegurar sua escalabilidade. Apesar do crescimento linear de mensagens, o tempo de entrega permaneceu praticamente constante (5.35). A entrega reduziu em 3% ao aumentar de 4096 para 10240 nós simulados, valor relativamente pequeno em comparação com o aumento do tamanho da rede (5.34).

Como mostra as figuras 5.34 e 5.35, vemos uma ligeira queda no tempo de entrega das mensagens com o aumento na quantidade de dispositivos na rede, mas essa

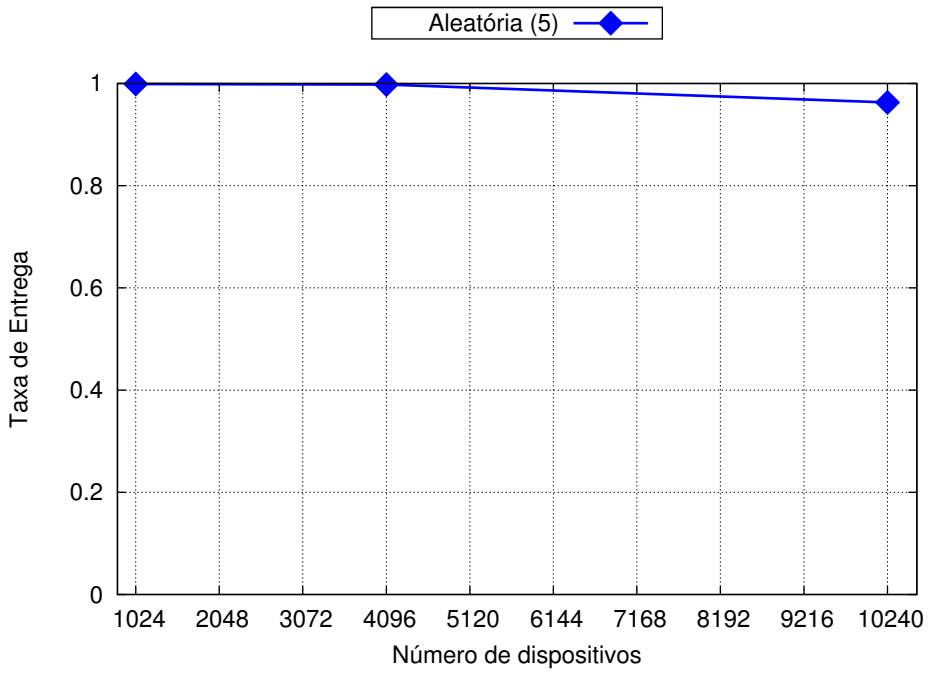


Figura 5.34: Análise: Taxa de entrega

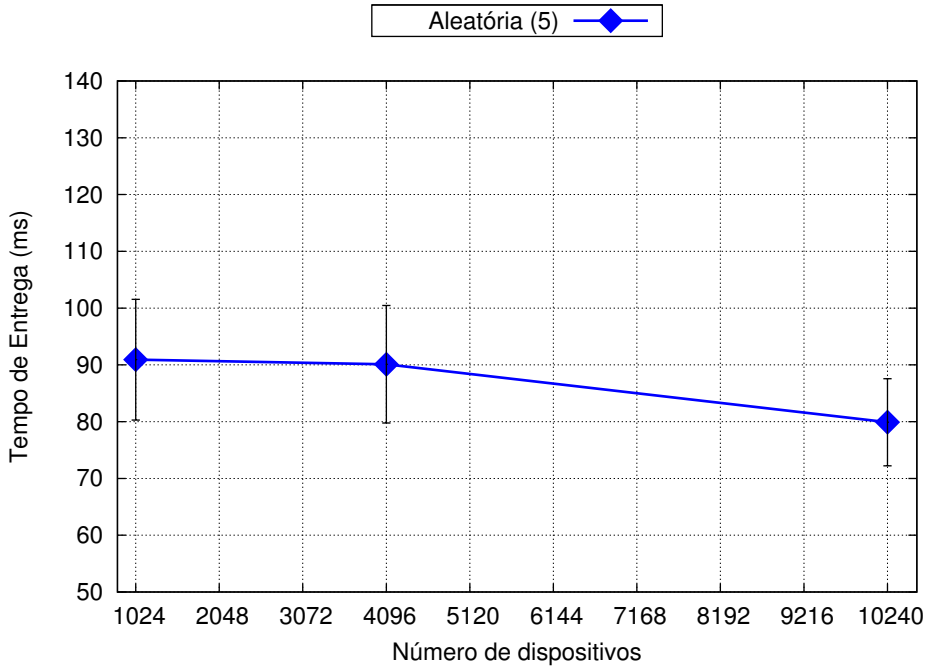


Figura 5.35: Análise: Tempo médio de entrega das mensagens

queda é pequena em levando em consideração a variação dos resultados. Esta ligeira queda no tempo pode ser explicado por haver uma redução na quantidade de nós recebedores, chega a ser 3% menor no caso com 10240 nós simulados.

Mesmo as mensagens sendo entregues com uma quantidade superior de saltos, aumentando de 4, 5 e 7 nos casos 1024 (5.18), 4096 (5.26) e 10240 (5.2) respectivamente, o tempo permanece praticamente constante considerando a variação dos

resultados. Isso se deve a característica da REPI de encaminhar as mensagens através de vários caminhos distintos.

Com estes resultados concluímos que usar a REPI como algoritmo probabilístico para busca de informação em uma rede, ao invés da inundação como Gnutella, se torna viável ao passo que se consegue atingir uma alta taxa de nós interessados, ou seja, nós com um determinado serviço buscado ao passo que mantém linear crescimento da quantidade de mensagens trocadas na rede. Outro fator interessante e importante é o baixo tempo de entrega destas mensagens o que faz com que um serviço buscado seja encontrado com alta probabilidade e em um tempo bem curto. Diferente de outros protocolos probabilísticos, a REPI carrega em seu cabeçalho informações úteis para a aplicação ao mesmo tempo que as utiliza na decisão de encaminhamento das mensagens.

O PA é essencial para o funcionamento da REPI, pois é o elemento que carrega informações do usuário e que direciona as mensagens através de vários caminhos na rede. O PA é o responsável pela alta taxa de entrega de mensagens em tempos curtos.

5.3 Discussão

Realizamos avaliações de formação da rede e da troca de mensagens REPI sobre a rede formada. Verificamos o baixo custo para o estabelecimento e manutenção de uma rede P2P através do algoritmo apresentado na seção 4.2.1. Este algoritmo de formação de uma rede sobreposta pode ser utilizado em vários outros contextos, além do utilizado neste trabalho. Um exemplo de utilização, seria para avaliar outros mecanismos de distribuição e busca de conteúdos.

Os resultados apresentados foram bem conservadores em relação à avaliação analítica realizada na seção 4.3. Com isso podemos conhecer o custo total de mensagens de controle em uma rede apenas conhecendo seu tamanho em quantidade de nós. Nas simulações vemos que a rede com 5 vizinhos descobertos obteve o menor custo de mensagens para a formação da rede ao passo que obteve ótimos resultados em relação à entrega de mensagens REPI com tempos relativamente baixos.

Vimos que o critério de vizinhança tem grande impacto na formação da rede, ou seja, implementar e avaliar outros critérios é um trabalho interessante sabendo que através dele é possível atingir ou mesmo melhorar alguma funcionalidade do sistema. Em nosso caso, utilizamos o PA como um critério de vizinhança com o intuito de aumentar a probabilidade de encaminhamento de mensagens aumentando, assim, a taxa de entrega, porém a entrega já foi praticamente 100% no caso de vizinhança aleatória, o que torna o critério ineficaz. Outros critérios podem ser implementados para poder aumentar ou reduzir a quantidade de saltos, o tempo de entrega de

mensagens, a colaboração da rede, quantidade de mensagens trocadas, entre outras variáveis do sistema.

Os resultados da simulação para o protocolo REPI nessa rede sobreposta formada mostram a eficiência na troca de mensagens, pois se consegue atingir mais de 95% dos nós interessados nos casos avaliados com um tempo de entrega entre $70ms$ a $100ms$. A grande vantagem da REPI é o fato das mensagens percorrerem vários caminhos distintos simultaneamente e probabilisticamente, o que reduz o tempo de entrega das mensagens ao passo que atinge quase todos os nós interessados na mensagem.

As últimas figuras apresentadas 5.33, 5.34 e 5.35 mostraram o impacto no aumento de nós na rede. Como discutido, a entrega degrada em 3%, proporção relativamente pequena em comparação com o aumento da rede. O tempo de entrega de mensagens permanece praticamente constante, o que mostra que o tamanho da rede não tem impacto sobre o tempo de entrega devido aos vários caminhos pelos quais as mensagens percorrem. A quantidade de mensagens cresce linearmente com a quantidade de dispositivos na rede, o que nos induz a pensar na questão de escalabilidade. Porém os resultados obtidos não são suficientes para assegurar a escalabilidade do sistema, apenas a sugerem.

Os resultados apresentados para a REPI foram melhores em relação ao algoritmo de inundação avaliado neste trabalho. Apesar de o algoritmo de inundação ser o pior caso para disseminação de mensagens, alguns sistemas ainda a utilizam para encaminhamento de mensagens, como o caso do Gnutella. Nossa análise para o algoritmo de inundação não levou em consideração as várias otimizações para reduzir o custo em mensagens do algoritmo, mas estas otimizações [60, 61] também podem ser aplicadas na formação da rede sobre a qual a REPI está aplicada.

Melhorar a rede sobreposta formada através de outros critérios de vizinhança ou outros mecanismos de formação da rede, pode melhorar ainda mais o funcionamento da REPI, reduzindo a quantidade de mensagens na rede e até mesmo reduzir o tempo de entrega das mensagens.

Capítulo 6

Conclusão

Neste trabalho introduzimos o algoritmo REPI para formação de uma rede sobreposta P2P com o intuito de avaliar seu uso como um novo protocolo para disseminação de mensagens por interesses, o protocolo REPI onde uma mensagem REPI não utiliza IP e é composta de características e interesses. Um objetivo deste trabalho foi desenvolver e avaliar o comportamento do protocolo REPI sobre a Internet. Sobre o algoritmo de formação da rede realizamos uma avaliação analítica e simulações; enquanto que para o protocolo REPI realizamos várias simulações. As contribuições resultantes foram:

- Uma aplicação social de *chat* sobre a Internet utilizando o algoritmo apresentado para a formação da rede P2P sobreposta e o protocolo REPI para a troca de mensagens. Este é o primeiro protótipo de aplicação utilizando a REPI sobre a Internet e está disponibilizada para a comunidade, denominada aplicação mensageira REPI-Internet;

- A aplicação implementada no simulador. O algoritmo de formação da rede P2P e sobre ela o protocolo REPI para troca de mensagens. Com a aplicação implementada no simulador NS-3.8 realizamos as avaliações apresentadas neste trabalho;

- Os resultados simulados e avaliados demostram o potencial do protocolo REPI aplicado sobre a Internet. Foi atingida uma taxa de entrega acima de 96% nos cenários avaliados. Conseguimos resultados que sugerem que a REPI é escalável nos ambientes estudados, um estudo mais detalhado deve ser realizado para realmente afirmar este fato;

- Nossa avaliação comparativa com o algoritmo de inundação demonstra o poder de utilização da REPI, pois a REPI consegue atingir uma taxa de entrega de mensagens semelhante com um custo em mensagens inferior ao algoritmo de inundação em um tempo de entrega das mensagens bem inferior;

- Os resultados obtidos demonstraram a existência de colaboração entre os dispositivos o que contribuiu para a alta taxa de entrega das mensagens. Diferentemente de outros mecanismos de colaboração, a REPI ao utilizar Prefixos Ativos como meio de disseminação das mensagens faz com que todos os nós colaborem de forma homogênea e imparcial, ou seja, os nós encaminhadores de mensagens são escolhidos de maneira aleatória seguindo uma distribuição normal das características presentes em seu PA;

Além da avaliação realizada neste trabalho, a aplicação desenvolvida e as inclusões feitas no simulador são contribuições que estarão disponibilizadas na *web* com o intuito de permitir e facilitar a evolução da pesquisa desse novo protocolo.

6.1 Trabalhos Futuros

Além do trabalho apresentado, é possível desenvolver uma gama de novos trabalhos neste contexto. A REPI, como um original protocolo para troca de mensagens, podemos desenvolver novas e originais aplicações utilizando esta diferente abordagem de troca de mensagens. As oportunidades de trabalhos futuros relacionados à REPI-Internet são:

- Pesquisar e desenvolver um conjunto de novas aplicações utilizando o protocolo REPI na troca de mensagens. Para isto vê-se necessário o desenvolvimento de um *framework* com um conjunto de API's (*Application Programming Interface*) para facilitar este desenvolvimento;

- Desenvolver novos critérios de formação de vizinhança, ou mecanismos de formação da rede sobreposta. Utilizar algoritmos DHT (*Distributed Hash Table*) com o intuito de formar uma rede P2P estruturada e assim avaliar o custo de formação da rede e o impacto sobre a REPI;

- Avaliar o impacto da REPI em outros cenários, dado que utilizamos apenas dois cenários em nossas avaliações (cenário preliminar e cenário Rede Rio). Os cenários avaliados podem ter sido favoráveis à REPI o que levou a obter os bons resultados apresentados;

- Avaliar outros contextos onde a REPI seria aplicável, como por exemplo, utilizar a REPI como mecanismo de resolução de nomes (DNS - *Domain Name System*) para descobrir na rede a localização de servidores;

- Avaliar outras configurações da REPI alterando, por exemplo, a quantidade de campos no PA, as distribuições utilizadas para a seleção dos campos, quantidade e tipos de interesses utilizados pelo protocolo. Avaliar estes e outros

parâmetros da REPI podem fazê-la se comportar melhor em cada contexto das aplicações utilizadas;

- Realizar experimentos em um ambiente real, como por exemplo, utilizar o *testbed* PlanetLab [62]. Pois os resultados apresentados neste trabalho foram obtidos através de simulações. Para validar esses resultados simulados seria interessante a execução da REPI em um ambiente real.

Referências Bibliográficas

[1] "Rede Rio". Disponível em: <http://www.rederio.br> . Último acesso em outubro de 2010.

[2] WASSERMAN, S., FAUST, K. *Social network analysis : methods and applications*. Structural analysis in the social sciences, 8. 1 ed. , Cambridge University Press, nov. 1994. ISBN: 0521387078. Disponível em: <http://www.worldcat.org/isbn/0521387078> .

[3] SOCOLOFSKY, T., KALE, C. "TCP/IP tutorial". RFC 1180 (Informational), jan. 1991. Disponível em: <http://www.ietf.org/rfc/rfc1180.txt> .

[4] CALLON, R. "Use of OSI IS-IS for routing in TCP/IP and dual environments". RFC 1195 (Proposed Standard), dez. 1990. Disponível em: <http://www.ietf.org/rfc/rfc1195.txt> . Updated by RFCs 1349, 5302, 5304.

[5] ROSCOE, T. "The End of Internet Architecture". In: *Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets'06)*, 2006.

[6] CROWCROFT, J., HAND, S., MORTIER, R., et al. "Plutarch: an argument for network pluralism". In: *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, FDNA'03, pp. 258–266, New York, NY, USA, 2003. ACM. ISBN: 1-58113-748-6. doi: http://doi.acm.org/10.1145/944759.944763. Disponível em: <http://doi.acm.org/10.1145/944759.944763> .

[7] KOPONEN, T., CHAWLA, M., CHUN, B.-G., et al. "A data-oriented (and beyond) network architecture". In: *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pp. 181–192, New York, NY, USA, 2007. ACM. ISBN: 978-1-59593-713-1. doi: http://doi.acm.org/10.1145/1282380.1282402. Disponível em: <http://doi.acm.org/10.1145/1282380.1282402> .

[8] ZHANG, L., DEERING, S., ESTRIN, D. "RSVP: A New Resource ReSerVation Protocol", *IEEE network*, v. 7, n. 5, set. 1993. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.25.8884> .

[9] "Alexa". Disponível em: <http://www.alexa.com/> . Último acesso em abril de 2011.

[10] "Google". Disponível em: <http://www.google.com> . Último acesso em abril de 2011.

[11] "Yahoo!" Disponível em: <http://www.yahoo.com> . Último acesso em abril de 2011.

[12] "YouTube". Disponível em: <http://www.youtube.com> . Último acesso em abril de 2011.

[13] "Facebook". Disponível em: <http://www.facebook.com> . Último acesso em outubro de 2010.

[14] "LinkedIn". Disponível em: <http://www.linkedin.com> . Último acesso em abril de 2011.

[15] CLARK, D. "The design philosophy of the DARPA internet protocols". In: *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pp. 106–114, New York, NY, USA, 1988. ACM. ISBN: 0-89791-279-9. doi: 10.1145/52324.52336. Disponível em: <http://dx.doi.org/10.1145/52324.52336> .

[16] JACOBSON, V., SMETTERS, D. K., THORNTON, J. D., et al. "Networking Named Content". In: *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pp. 1–12, New York, NY, USA, 2009. ACM. ISBN: 978-1-60558-636-6. doi: http://doi.acm.org/10.1145/1658939.1658941. Disponível em: <http://doi.acm.org/10.1145/1658939.1658941> .

[17] KROL, E., HOFFMAN, E. "FYI on "What is the Internet?"". RFC 1462 (Informational), maio 1993. Disponível em: <http://www.ietf.org/rfc/rfc1462.txt> .

[18] JACOBSON, V., MOSKO, M., SMETTERS, D., et al. "Content-Centric Networking: Whitepaper Describing Future Assurable Global Networks". Response to DARPA RFI SN07-12, 2007.

[19] CARZANIGA, A., WOLF, A. L. "Content-based Networking: A New Communication Infrastructure". In: *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, n. 2538, Lecture Notes in Computer Science, pp. 59–68, Scottsdale, Arizona, out. 2001. Springer-Verlag.

[20] ARIANFAR, S., NIKANDER, P., OTT, J. "On content-centric router design and implications". In: *Proceedings of the Re-Architecting the Internet Workshop*, ReARCH '10, pp. 5:1–5:6, New York, NY, USA, 2010. ACM. ISBN: 978-1-4503-0469-6. doi: http://doi.acm.org/10.1145/1921233.1921240. Disponível em: <http://doi.acm.org/10.1145/1921233.1921240> .

[21] DUTRA, R. D. C., MORAES, H. F. D., AMORIM, C. L., et al. "REPI: Rede de comunicação Endereçada Por Interesses". In: *WP2P 2010: Anais do VI Workshop de Redes Dinâmicas e Sistemas P2P . Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC2010)*, may 2010.

[22] DUTRA, R. C., AMORIM, C. L. *Modelo de comunicação endereçada por interesses.* Relatório técnico, ES 733 / PESC - COPPE - UFRJ, 2010.

[23] "Aplicação REPI-Internet". Disponível em: <http://repi-internet.lcp.coppe.ufrj.br/> . Último acesso em Março de 2011.

[24] "NS-3". . Disponível em: <http://www.nsnam.org> . Último acesso em outubro de 2010.

[25] ZHANG, L., ESTRIN, D., BURKE, J., et al. *Named Data Networking (NDN) Project.* Relatório técnico, University of California (Los Angeles), Palo Alto Research Center (PARC), University of Arizona, University of California (Irvine), University of California (San Diego), Colorado State University, University of Illinois at Urbana-Champaign, University of Memphis, Washington University, Yale University, 2010. Disponível em: <http://www.named-data.net/> .

[26] TH, P., FELBER, P. A., GUERRAOUI, R., et al. "The many faces of publish/subscribe", *ACM Comput. Surv.*, v. 35, n. 2, pp. 114–131, jun. 2003. ISSN: 0360-0300. doi: 10.1145/857076.857078. Disponível em: <http://dx.doi.org/10.1145/857076.857078> .

[27] LUA, E. K., CROWCROFT, J., PIAS, M., et al. "A Survey and Comparison of Peer-to-Peer Overlay Network Schemes", *IEEE Communications Surveys and Tutorials*, v. 7, pp. 72–93, 2005.

[28] STOICA, I., MORRIS, R., LIBEN-NOWELL, D., et al. "Chord: a scalable peer-to-peer lookup protocol for internet applications", *IEEE/ACM Trans. Netw.*, v. 11, n. 1, pp. 17–32, 2003. ISSN: 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2002.808407.

[29] ROWSTRON, A., DRUSCHEL, P. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". In: *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, Springer-Verlag, pp. 329–350, 2001. Disponível em: <http://www.springerlink.com/content/7y5mjjep0hqlctv6> .

[30] MONNERAT, L., AMORIM, C. "D1HT: a distributed one hop hash table", *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, p. 10 pp., April 2006. doi: 10.1109/IPDPS.2006.1639278.

[31] "BitTorrent". Disponível em: <http://www.bittorrent.com> . Último acesso em outubro de 2010.

[32] "Gnutella". Disponível em: <http://rfc-gnutella.sourceforge.net> . Último acesso em outubro de 2010.

[33] COHEN, B. "Incentives Build Robustness in BitTorrent", *In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.1911> .

[34] GIORDANO, S. *Mobile ad hoc networks*. New York, NY, USA, John Wiley & Sons, Inc., 2002. ISBN: 0-471-41902-8.

[35] CHANGLING LIU, J. K. *A Survey of Mobile Ad Hoc network Routing Protocols*. Relatório técnico, University of Magdeburg, 2005.

[36] GRANJA, R. S. *Protocolos para Redes de Comunicação Ad hoc Endereçadas por Interesses*. Tese de Mestrado, PESC/COPPE/Universidade Federal do Rio de Janeiro, 2010.

[37] MOTEIV. "Tmote Sky Datasheet". 2006. Disponível em: <http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf> . Último acesso em julho de 2010.

[38] IEEE. "802.15.4 Standard, 2003". 2003. Disponível em: <http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf> . Último acesso em julho de 2010.

[39] IEEE. “802.15.4 Standard, 2006”. 2006. Disponível em: <http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf> . Último acesso em junho de 2010.

[40] “ZigBee Alliance”. 2009. Disponível em: <http://www.zigbee.org/> . Último acesso em junho de 2010.

[41] “Limewire”. Disponível em: <http://www.limewire.com> . Último acesso em outubro de 2010.

[42] “Kazaa”. Disponível em: <http://www.kazaa.com> . Último acesso em outubro de 2010.

[43] HOLDREGE, M., SRISURESH, P. “Protocol Complications with the IP Network Address Translator”. RFC 3027 (Informational), jan. 2001. Disponível em: <http://www.ietf.org/rfc/rfc3027.txt> .

[44] FORD, B., SRISURESH, P., KEGEL, D. “Peer-to-peer communication across network address translators”. In: *ATEC ’05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pp. 13–13, Berkeley, CA, USA, 2005. USENIX Association.

[45] YOSHIMI, H., ENOMOTO, N., CUI, Z., et al. “NAT Traversal Technology of Reducing Load on Relaying Server for P2P Connections”. pp. 100 –104, jan. 2007. doi: 10.1109/CCNC.2007.27.

[46] NETWORK WORKING GROUP. *RFC 5389 - Session Traversal Utilities for NAT (STUN)*. Relatório técnico, IETF, October 2008. Disponível em: <http://tools.ietf.org/html/rfc5389> .

[47] MAHY, R., MATTHEWS, P., ROSENBERG, J. “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”. RFC 5766, abr. 2010. Disponível em: <http://www.ietf.org/rfc/rfc5766.txt> .

[48] POSTEL, J. “User Datagram Protocol”. RFC 768 (Standard), ago. 1980. Disponível em: <http://www.ietf.org/rfc/rfc768.txt> .

[49] POSTEL, J. “Transmission Control Protocol”. RFC 793 (Standard), set. 1981. Disponível em: <http://www.ietf.org/rfc/rfc793.txt> . Updated by RFCs 1122, 3168.

[50] BERGNER, M. *Improving Performance of Modern Peer-to-peer services*. Relatório técnico, Umea University, 2003.

[51] "Emule". Disponível em: <http://www.emule-project.net> . Último acesso em outubro de 2010.

[52] AUDET, F., JENNINGS, C. "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP". RFC 4787 (Best Current Practice), jan. 2007. Disponível em: <http://www.ietf.org/rfc/rfc4787.txt> .

[53] WEINGARTNER, E., VOM LEHN, H., WEHRLE, K. "A performance comparison of recent network simulators". In: *Proceedings of the IEEE International Conference on Communications 2009 (ICC 2009)*, Dresden, Germany, 2009. IEEE. Disponível em: <http://ds.informatik.rwth-aachen.de/members/weingaertner/publications/2008-06-Weingaertner-ICC-NetworkSimulatorComparison> .

[54] "Download ns-3". . Disponível em: <http://www.nsnam.org/releases> . Último acesso em outubro de 2010.

[55] PERKINS, C. E., ROYER, E. M. "Ad-hoc On-Demand Distance Vector Routing". In: *2nd IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '99, New Orleans, Lousiana, USA*, pp. 90–100. IEEE, IEEE, 1999.

[56] JACQUET, P., MUHLETHALER, P., CLAUSEN, T., et al. "Optimized Link State Routing Protocol for Ad Hoc Networks". In: *Multi Topic Conference, 2001. IEEE INMIC 2001 Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pp. 62 – 68, Lahore Pakistan, 2001. IEEE. doi: 10.1109/INMIC.2001.995315. Disponível em: <http://hal.inria.fr/inria-00471622/PDF/inmic2001.pdf> . Best paper award.

[57] "The Message Passing Interface (MPI) standard". Disponível em: <http://www.mcs.anl.gov/research/projects/mpi> . Último acesso em outubro de 2010.

[58] "Manual ns-3". . Disponível em: <http://www.nsnam.org/docs/release/manual.html> . Último acesso em outubro de 2010.

[59] "NACAD (Núcleo de Atendimento em Computação de Alto Desempenho)". Disponível em: <http://www.nacad.ufrj.br> . Último acesso em abril de 2011.

[60] LE-TING, T., TAO, T. "A search algorithm on Gnutella Networks Based on Resource Replication". In: *Computer Application and System Modeling*

(ICCASM), 2010 International Conference on, v. 15, pp. V15–107 –V15–110, oct. 2010. doi: 10.1109/ICCASM.2010.5622534.

[61] CHANDRA, J., SHAW, S. K., GANGULY, N. "HPC5: An efficient topology generation mechanism for Gnutella networks", *Computer Networks*, v. 54, pp. 1440–1459, June 2010. ISSN: 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2009.11.017. Disponível em: <`http://dx.doi.org/10.1016/j.comnet.2009.11.017`> .

[62] "PlanetLab". Disponível em: <`http://www.planet-lab.org/`> . Último acesso em abril de 2011.

Apêndice A

Aplicação REPI-Internet

A REPI é uma rede P2P que utiliza um original protocolo de passagem de mensagens, onde uma mensagem contém um prefixo que informa os interesses nela contidos. As mensagens são encaminhadas de dispositivo para dispositivo, utilizando seus prefixos de acordo com um critério de encaminhamento pré-estabelecido. Além disso, uma mensagem recebida por um dispositivo que tenha um ou mais interesses coincidentes com o do seu prefixo, será copiada pelo dispositivo e entregue e tratada pela aplicação local correspondente.

O objetivo do protocolo de comunicação REPI é mudar o foco de comunicação do dispositivo para o usuário, assim sendo, o direcionamento das mensagens é controlado pelos interesses que elas encerram. Especificamente, numa REPI, o dispositivo de cada usuário conectado, além de colaborar no encaminhamento das mensagens recebidas, ele também funciona como um filtro de modo a identificar e entregar mensagens que o usuário tenha pré-definido como de seu interesse.

O protocolo REPI não armazena as mensagens, assim um usuário que se conectar a uma REPI já estabelecida não terá acesso às mensagens anteriores que foram trocadas pelos outros usuários. O protocolo encaminha uma mensagem através dos usuários que estão presentes na rede no momento do envio da mensagem.

A aplicação REPI-Internet faz uso do protocolo REPI para realizar a troca de mensagens entre os usuários na rede. Para que os usuários possam participar desta rede é necessário ter a aplicação REPI-Internet sendo executada. A aplicação pode ser executada em qualquer sistema operacional desde que tenha conexão com a Internet e a máquina virtual Java (no mínimo a versão Java 1.5) instalada. A aplicação está disponível em [23].

Para realizar a troca de mensagens utilizando o protocolo REPI descrito neste trabalho, a aplicação REPI-Internet usa o algoritmo apresentado em 4.2.1 para realizar a formação da rede *overlay* . O Laboratório de Computação Paralela (LCP) fornece a aplicação utilizando como nó origem uma de suas máquinas com um IP público, mas nada impede que se utiliza outro nó origem com endereço diferente. A

aplicação gera um arquivo de configuração na pasta local do usuário, neste arquivo é possível alterar o endereço do nó origem, podendo adicionar um ou vários outros endereços.

A aplicação funciona em redes locais sem a necessidade do nó origem, dessa forma pode-se criar redes REPI locais em ambientes privativos ou corporativos. Neste caso, a rede *overlay* formada será composta apenas pelos dispositivos presentes naquela região que estejam fazendo uso da aplicação. A única limitação desta aplicação em ambientes locais é o uso da porta de comunicação que deve ser a mesma em todas as aplicações. Por padrão a aplicação faz uso de uma porta UDP alta (UDP:61374), mas a porta usado pode ser alterada no arquivo de configuração gerado pela aplicação.

A.1 Interface REPI-Internet

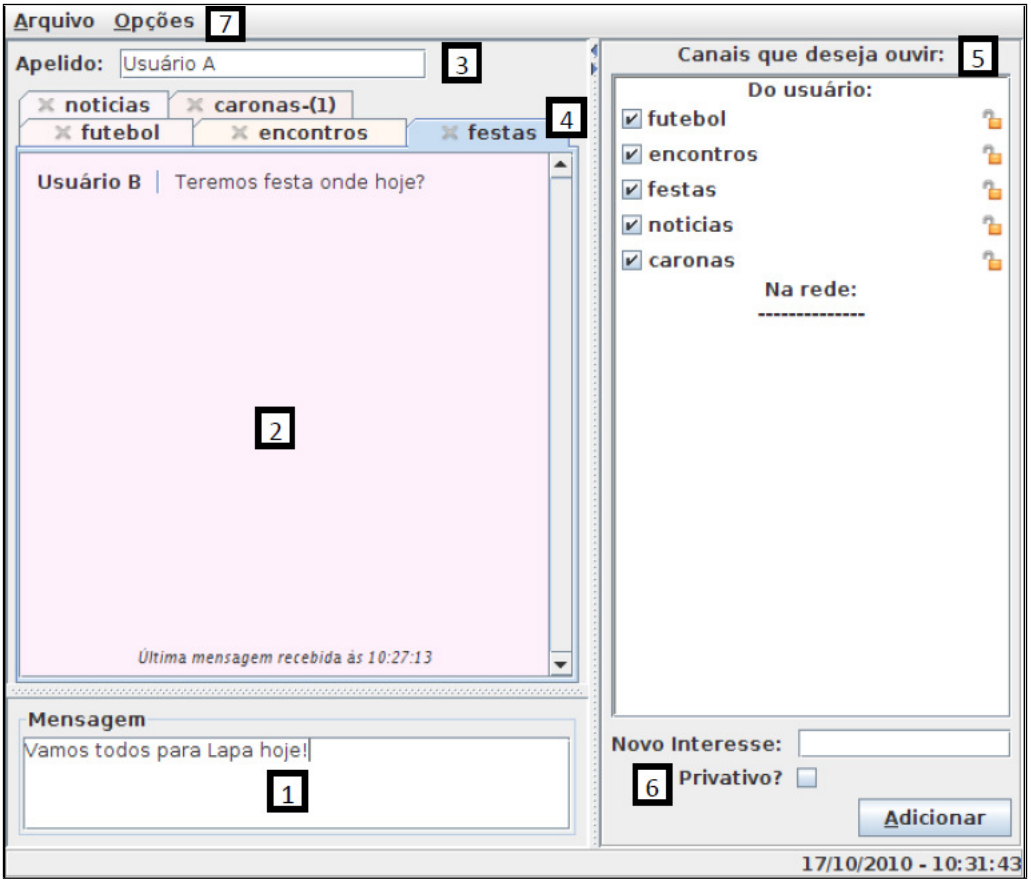


Figura A.1: Tela principal da aplicação REPI-Internet

Ao executar a aplicação, a janela semelhante à da figura A.1 será exibida. A interface da aplicação é equivalente a de várias outras aplicações de trocas de mensagens, porém com algumas especificidades. A figura A.1 foi divida em regiões, a segui mais detalhes sobre cada região:

1. Este campo é usado para escrever uma mensagem para ser enviada pelo canal selecionado na região 4;

2. Neste campo são mostradas as mensagens enviadas e recebidas no canal selecionado na região 4;

3. Campo que indica o apelido a ser usado nas mensagens do usuário; Caso o usuário não preencha esses campos, as mensagens serão exibidas como se fossem anônimas;

4. Região onde o usuário pode selecionar um dos canais escolhidos em 5; Tanto para ler as mensagens recebidas com o interesse selecionado como também poder enviar mensagens para esse canal. O número ao lado do nome de cada canal indica a quantidade de novas mensagens recebidas naquele canal, caso este não seja a aba atualmente selecionada (caso ilustrado no canal "caronas");

5. Região onde o usuário pode selecionar quais canais pelos quais deseja receber mensagens, ilustrado na figura A.2. Esta região é dividida em duas partes: canais do usuário e canais na rede. Os canais do usuário são os selecionados por ele para poder habilitar o recebimento das mensagens. Os canais na rede mostram aqueles criados por outros usuários; Caso o usuário demonstre interesse por um desses canais na rede, há a possibilidade de passar a receber as mensagens de um canal na rede simplesmente clicando no nome do canal;

6. O usuário pode estar interessado em conversar sobre outro assunto de interesse ainda não presente na rede, logo em 6 é possível criar novos canais de interesse. Caso o usuário queira privacidade é possível colocar uma senha em cada canal criado, porém essa senha deve ser combinada previamente pelos usuários. Para se criar um canal com senha basta selecionar a opção "Privativo" que irá aparecer um campo para se colocar uma senha.

7. Nesta região está o menu da aplicação, com as seguintes opções:

    - Em "Arquivo" tem a opção de sair da aplicação;
    - Em "Opções" temos:
        - **Mensagens Antigas:** esta opção permite rever as mensagens trocadas de acordo com o interesse e a data das mensagens como ilustra a figura A.3;
        - **Enviar Feedback:** esta opção permite enviar uma mensagem para o desenvolvedor do sistema com dúvidas, sugestões, críticas, bugs, entre outros;]

– **About:** esta opção mostra uma janela com os idealizadores deste projeto, assim como a versão da aplicação.

A seguir detalhes de uso da aplicação.

A.2 Utilizando a Aplicação

Existe a possibilidade de transformar um canal público em canal privativo, ou vice-versa. Em cada canal apresentado na aplicação existe um ícone de cadeado representando se o canal é público ou não. Na figura A.2, todos os canais de interesse do usuário com exceção do canal "privativo" são canais públicos representados pelo ícone de cadeado aberto. Já o canal "privativo", representado pelo cadeado fechado, é um canal cujo usuário definiu uma senha.

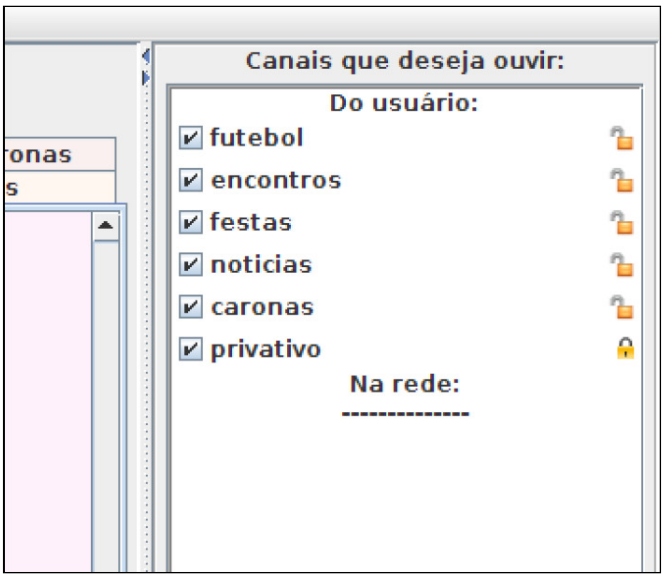


Figura A.2: Região com os canais de interesse

Para definir uma senha para os canais públicos, ou retirar a senha de um canal reservado, basta clicar no ícone de cadeado presente na frente de cada canal. Dessa forma, irá aparecer uma janela pedindo a nova senha para o canal; caso a nova senha seja vazio o canal passará a ser público.

Para se criar um novo canal de interesse, basta ir à região representada pela figura A.3 e definir o novo interesse. Caso esse novo interesse deva ser tratado como um canal privativo, existe a opção "Privativo". Ao selecionar a opção para criar um canal reservado, o campo de senha aparecerá, neste campo deve ser colocada uma senha que irá proteger o canal. Vale ressaltar que, neste caso, apenas os usuários que tiverem este canal com a mesma senha realizarão a troca de mensagens.

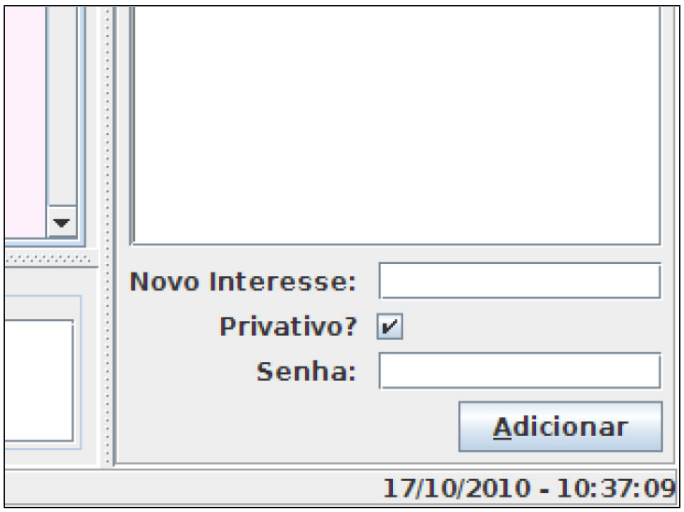


Figura A.3: Região para a criação dos canais de interesse

A.3 Revendo Conversas

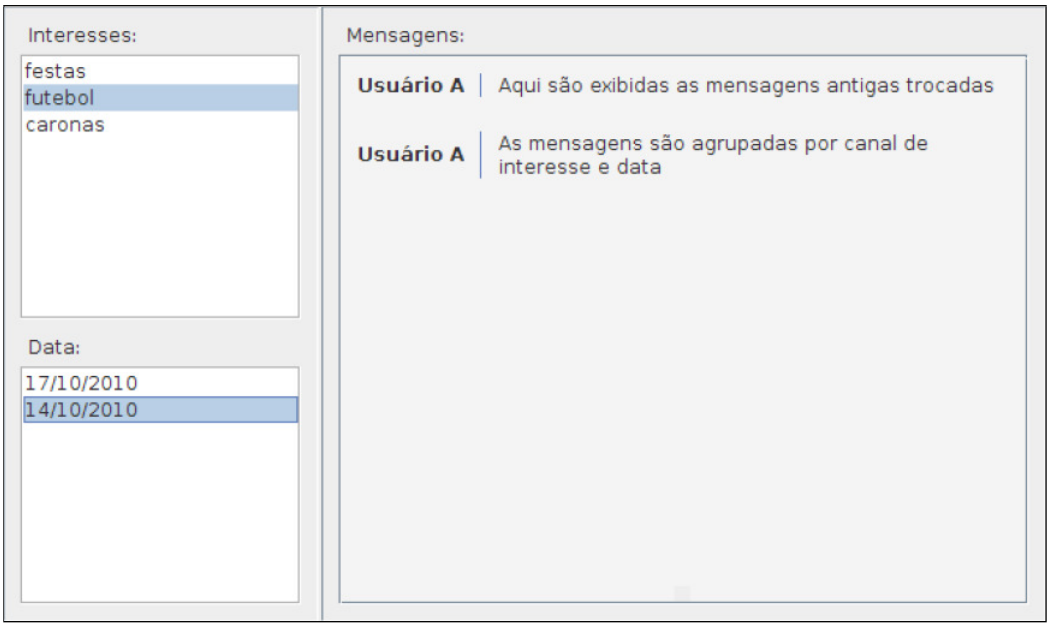


Figura A.4: Janela para rever mensagens antigas

As mensagens dos canais de interesses ao qual o usuário já realizou troca de mensagens são armazenadas pela aplicação para permitir que o usuário possa rever suas conversas no futuro. No menu existe a opção "Opções→Mensagens Antigas". Ao selecionar esta opção irá abrir uma janela semelhante ao ilustrado na figura A.4. Nesta janela, existem três regiões:

- **Interesses:** São exibidos os canais de interesses que o usuário já realizou troca de mensagens;

- **Data:** São exibidas as datas que foram realizadas as conversas por canal de interesse;
- **Mensagens:** São exibidas as mensagens já trocadas.

A.4 Enviando *Feedback*

Durante a utilização da aplicação REPI-Internet o usuário pode encontrar alguns problemas de execução, detectar bugs, ou até mesmo sugerir melhorias. Para criar um canal de comunicação com os desenvolvedores do sistema, foi criado uma opção presente no menu "Opções→Enviar Feedback". Ao selecionar esta opção a janela ilustrada na figura A.5 será exibida. Nesta janela existem dois campos:

- **Email:** Este campo é usado para definir o email do usuário, caso ele deseje receber uma resposta. Não é obrigatório seu preenchimento;
- **Mensagem:** Neste campo deve ser preenchida com as dúvidas, sugestões, propostas ou bugs.

Essa mensagem é enviada para os desenvolvedores que irão respondê-la assim que for possível.

Nesta seção foi descrito as funcionalidades da aplicação REPI-Internet de forma demonstrativa. Foi mostrada a interface da aplicação e a função de cada item presente nela. Essa aplicação faz uso de um modelo relativamente novo de comunicação e através desta aplicação é fácil compreender como o protocolo funciona.

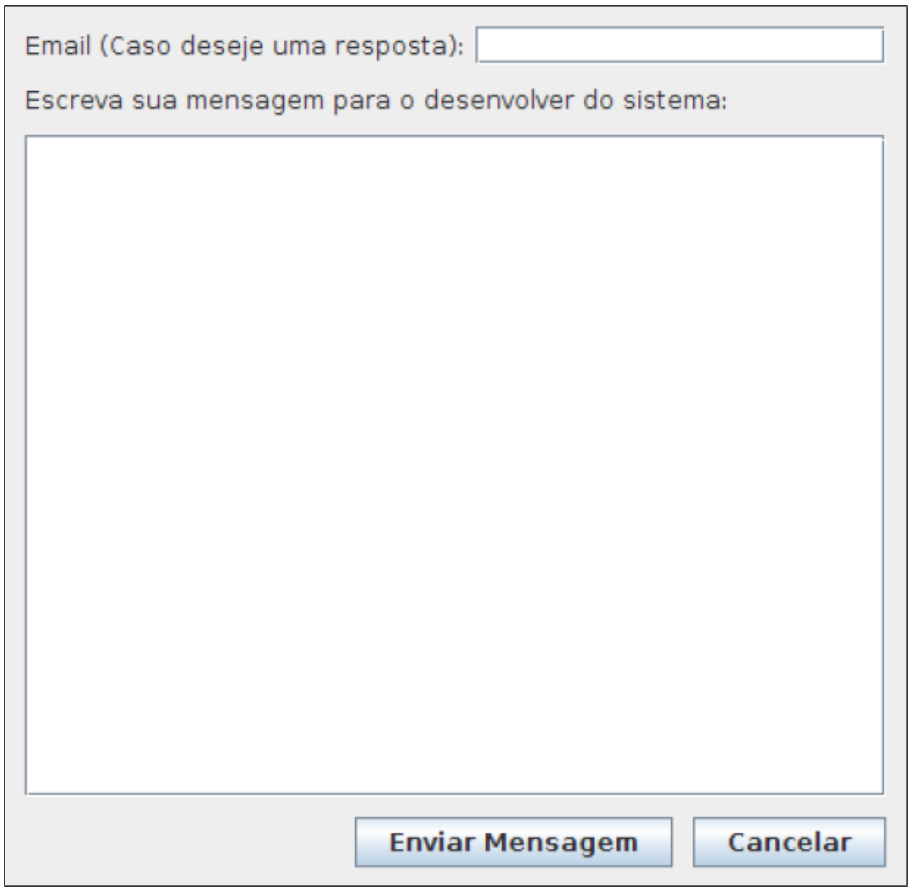


Figura A.5: Janela para enviar mensagens aos desenvolvedores

Apêndice B

Implementação do Algoritmo REPI-Internet

Neste apêndice apresentamos os módulos criados para realizar a implementação do algoritmo REPI-Internet, descrito na seção 4.2.1. Esta implementação foi realizada em componentes, onde cada um deles é responsável por uma tarefa do algoritmo. O algoritmo se divide nas seguintes tarefas:

- Coleta de estatísticas (*Statistics*): responsável pela coleta dos dados das variáveis do sistema para avaliação;
- Gerenciamento de mensagens (*MessageManager*): responsável pelo tratamento das mensagens de interesse. Nele ficam armazenadas as últimas mensagens de interesse recebidas pelos nós do sistema, para isso é criada uma memória local em cada nó;
- Gerenciamento da lista de vizinhos (*ListPeerManager*): responsável pelas informações de cada vizinho descoberto na rede, além ser responsável pela manutenção das sessões NAT estabelecidas para cada vizinho;
- Gerenciamento de Log (*LogManager*): responsável por armazenar os logs do sistema, logs tanto de funcionamento para fins de depuração quanto de troca de mensagens do sistema;
- Filtro de casamento (*MatchFilter*): responsável pela decisão de encaminhar as mensagens recebidas e por verificar se tem interesse em aceitar a mensagem;
- Sockets de comunicação (*AsyncUDPServer*, *AsyncUDPClient*, *AsyncUDPHandleMsg*): responsáveis por realizar a comunicação, tanto no recebimento quanto no envio das mensagens na rede;

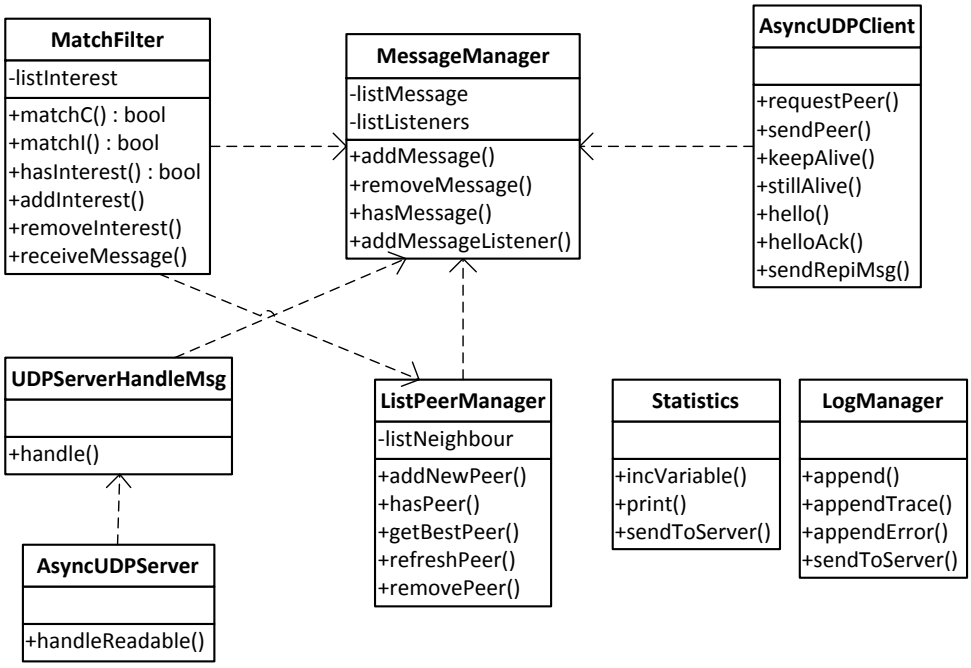


Figura B.1: Lista de componentes do sistema

As tarefas descritas acima compõem o núcleo do sistema de troca de mensagens por interesse. A figura B.1 ilustra a dependência entre os módulos implementados. A seguir será descrito cada componente em detalhes.

B.1 Coleta de Estatísticas

Este componente foi criado com o intuito de facilitar a coleta das variáveis do sistema. Estas variáveis são necessárias para o cálculo das métricas utilizadas para a avaliação do sistema. As variáveis computadas pelo sistema para realizar as avaliações são as seguintes:

- Quantidade de mensagens enviadas e recebidas de cada tipo: *Hello*, *Hello-Ack*, *RequestPeer*, *SendPeer*, *KeepAlive* e *StillAlive* que somados compõem as mensagens de controle (MC), e REPI a mensagem de interesse (MI);
- Quantidade de mensagens aceitas, quando ocorre casamento de I (MA);
- Quantidade de mensagens descartadas pelo não casamento da parte C, limite de saltos atingido, perda de pacotes na rede (MD);

- Quantidade de mensagens encaminhadas, totais (MET) e colaborativas (MEC). Utilizadas para calcular a quantidade de nós colaboradores (NC);
- Quantidade de vizinhos (NV);
- Quantidade média de saltos de uma mensagem REPI (HTL);
- Quantidade de nós colaboradores (NC);
- Tempo para a entrega das mensagens REPI (T).

Estas variáveis são coletadas e armazenadas em arquivo. Caso esteja realizando testes com a aplicação, estes dados coletados e armazenados em arquivo são enviados para um servidor que irá realizar a consolidação de todos os dados coletados.

B.2 Gerenciamento de Mensagens

A tarefa de gerenciamento de mensagens é feita pelo componente *MessageManager*. Este componente é o responsável pela gerência das mensagens de interesse recebidas no sistema. Como o algoritmo REPI Internet necessita da existência de uma memória local em todos os dispositivos, este módulo é o responsável pelo armazenamento destas. São armazenadas todas as últimas 100 mensagens distintas recebidas pelo dispositivo.

Como todas as mensagens de interesses que chegam ao sistema são repassadas para esse gerente, todos os outros componentes que desejam receber esse tipo de mensagem devem se registrar nele. Os componentes registrados são armazenados no *listListeners* do gerente de mensagens. Assim sendo, quando novas mensagens chegarem ao gerente, serão repassadas para os componentes registrados.

Quando o componente *AsyncUDPHandleMsg* detecta o recebimento de uma mensagem de interesse, essa mensagem é repassada para o gerente de mensagem que a repassa a todos os componentes do sistema registrados nele.

B.3 Gerenciamento da Lista de Vizinhos

A tarefa de gerenciar os vizinhos descobertos pelo algoritmo de formação da rede é feita pelo componente *ListPeerManager*. Este componente contém uma lista (*listNeigbour*) com informações de todos os vizinhos encontrados na rede.

Os critérios de seleção de vizinhos são implementados neste componente. Como apresentado nesta dissertação, foram implementados os critérios de seleção aleatória de vizinhos ou utilizando o prefixo como meio de seleção. A função *getBestPeer*()

seleciona um vizinho na lista baseado no critério de vizinhança que está sendo utilizado pelo sistema.

Novos vizinhos descobertos são adicionados à lista deste gerente. De tempos em tempos, caso nenhuma mensagem seja trocada com cada vizinho na lista, o gerente é responsável por enviar uma mensagem de *KeepAlive* a cada um membro presente na lista. Esta mensagem é enviada com o objetivo de manter sempre ativa as sessões NAT estabelecidas. Esta mensagem também tem o objetivo de verificar se o vizinho ainda está presente na rede, caso seja enviada mais de duas dessas mensagens sem obter nenhuma resposta, o vizinho é removido da lista.

B.4 Gerenciamento de Log

A tarefa de manipular os logs gerados pelo sistema são feitos pelo componente *LogManager*. Através desse componente, três tipos de logs são gerados pelo sistema. O primeiro deles é feito pela função *append*(), que é utilizado para gerar log de depuração do sistema. O segundo é feito pela função *appendTrace*(), que é utilizado para gerar log da troca de mensagens realizadas no sistema, tanto o recebimento quanto envio das mensagens. E por último, realizado pela função *appendError*(), os logs de erros do sistema.

Todos os componentes do sistema geram logs que são tratados pela gerente de log. Estes logs podem ser gravados em arquivos, caso a opção de log do sistema esteja habilitado. Este log pode ser enviado para um servidor da mesma maneira que as estatísticas coletadas no sistema. Esta opção é para permitir que erros encontrados no sistema possam ser enviados para os desenvolvedores e assim serem corrigidos.

B.5 Filtro de Casamento

O filtro de casamento utilizado neste trabalho foi implementado no componente *MatchFilter*. Neste componente os campos do prefixo ativo contidos na mensagem de interesse são avaliados individualmente. Estes campos são comparados, campo a campo, com o prefixo ativo do dispositivo. Como é o *MatchFilter* o responsável por decidir quanto ao encaminhamento e aceitação das mensagens, nele devem estar registrados todos os interesses do usuário.

A função *matchC*() avalia os campos de características de ambos os prefixos (do dispositivo e da mensagem). Caso ocorra ao menos um campo semelhante o *MatchFilter* decide encaminhar a mensagem.

A função *matchI*() avalia os interesses do dispositivo e verifica se o usuário quer a mensagem recebida. Caso a mensagem seja de interesse do usuário, ela é aceita repassada para a aplicação exibi-la ao usuário.

Este componente se registra no *MessageManager* para poder receber as mensagens que chegam ao dispositivo e poder avaliar seu prefixo.

B.6 Sockets de Comunicação

Os sockets de comunicação são os responsáveis pelo envio e recebimento de mensagens da rede. Foram utilizados dois sockets: um socket cliente (*AsyncUDPClient*), responsável pelo envio das mensagens na rede; e um socket servidor (*AsyncUDP-Server*), responsável pelo recebimento das mensagens.

O *AsyncUDPClient* tem implementado todas as funções para o envio de cada tipo de mensagem existente no sistema. Quando uma dessas funções é executada, a mensagem é colocada em uma fila para que seja inserida na rede. Existe um *thread* responsável por enviar todas as mensagens presentes nesta fila. Este *thread* acorda sempre que uma nova mensagem chega à fila e volta a dormir quando a fila fica vazia.

O *AsyncUDPServer* é um *thread* que fica aguardando o recebimento de alguma mensagem. Ao receber uma mensagem, ela é repassada para *UDPServerHandleMsg* que tem implementado o algoritmo apresentado na seção 4.2.1. Este algoritmo é o responsável por tratar cada mensagem recebida. Nele está implementado o algoritmo de busca de vizinhos, a técnica de *hole punching* e o repasse das mensagens de interesse para o *MessageManager*.

Estes componentes compõem o núcleo do sistema, a aplicação de troca de mensagens implementada neste trabalho se comunica com estes componentes. A interface gráfica da aplicação registra no *MatchFilter* os interesses definidos pelo usuário. O *MatchFilter* ao verificar uma nova mensagem de interesse repassa para a interface gráfica que a apresenta para o usuário. Novos interesses que o *MatchFilter* recebe, também são repassados para a interface gráfica, assim sendo o usuário pode ver os interesses que estão sendo utilizados na rede e pode passar a participar desses novos grupos de interesse.