UNIVERSIDADE FEDERAL FLUMINENSE

Igor Cesar Gonzalez Ribeiro

Verificação Probabilística de Assinaturas para a Mitigação de Ataques de Poluição em Redes Centradas em Conteúdo

NITERÓI

2013

UNIVERSIDADE FEDERAL FLUMINENSE

Igor Cesar Gonzalez Ribeiro

Verificação Probabilística de Assinaturas para a Mitigação de Ataques de Poluição em Redes Centradas em Conteúdo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação.
Área de concentração: Redes e Sistemas Distribuídos e Paralelos.

Orientador:
Célio Vinicius Neves Albuquerque

Co-orientador:
Antônio Augusto de Aragão Rocha

NITERÓI

2013

Igor Cesar Gonzalez Ribeiro

Verificação Probabilística de Assinaturas para a Mitigação de Ataques de Poluição em Redes Centradas em Conteúdo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Redes e Sistemas Distribuídos e Paralelos.

Aprovada em Agosto de 2013.

BANCA EXAMINADORA

Prof. Célio V. N. Albuquerque - Orientador, UFF

Prof. Antônio Augusto de A. Rocha - Coorientador, UFF

Prof. Igor M. Moraes, UFF

Profª. Flávia C. Delicato, UFRJ

Niterói

2013

A Gabriela, que me deu inspiração para a realização desse trabalho.

Agradecimentos

Apesar de sua única autoria, esta dissertação é, na verdade, produto da colaboração de diversas pessoas e entidades. Mesmo sabendo que serei injusto com alguns, arriscarei aqui lembrar daqueles que, direta ou indiretamente, contribuíram para que este trabalho deixasse o mundo das ideias e se transformasse em algo factível.

Primeiramente, gostaria de agradecer aos meus pais pelo apoio, carinho e dedicação que sempre me dispensaram. Durante minha criação, a educação sempre foi prioridade e essa dissertação nada mais é do que o resultado das bases sólidas que eles me proporcionaram.

A Gabriela, mulher incrível com quem tenho tido o privilégio de dividir os últimos 9 anos da minha vida. Seu carinho e seus conselhos foram fundamentais para me manter focado, mesmo nos momentos mais difíceis do mestrado. Muito obrigado pela paciência de ouvir os meus monólogos sobre o meu trabalho. Sei que você não compreendeu muito do que eu disse, mas conversar contigo me trouxe inspiração e otimismo.

Aos amigos que fiz durante esta etapa da minha vida. Edelberto e Joel vocês contribuíram de diversas formas para este trabalho. As nossas conversas ao redor da garrafa térmica sempre foram muito produtivas, fossem elas sobre questões acadêmicas ou sobre amenidades. Gostaria de agradecer também ao Diego, cuja ajuda foi imprescindível para a realização deste trabalho.

Ao Flávio, um grande amigo que a vida acadêmica me proporcionou. Ninguém ouviu tanto os meus lamentos, minhas dúvidas e minhas ideias quanto você. Sua amizade tornou os dias de aula e de estudo mais agradáveis. Espero que mantenhamos contato e que a nossa amizade ultrapasse os muros da UFF.

Ao Toscano, por ter me dado força ao longo desses anos. Nossas conversas sobre segurança são sempre muito produtivas.

A equipe do laboratório midiacom, que me permitiu desenvolver o meu trabalho em um ambiente confortável e propício ao desenvolvimento de novas ideias.

A querida Marister Outão, pessoa maravilhosa, sempre disposta a ajudar. Sua dedi-

cação e carinho tornaram o ambiente acadêmico mais familiar. Eu teria cedido ao sono por diversas vezes não fosse o café quentinho que você sempre faz. Muito obrigado pelo carinho e, é claro, pelo casaco de couro.

Ao prof. Célio, por ter aceitado me orientar e ter exercido esta função em todo o seu significado. Obrigado pelos conselhos, pelas revisões, pela dedicação e, é claro, pelos puxões de orelha, sempre muito sutis, mas também muito oportunos.

Ao prof. Antônio Augusto, que tão gentilmente me coorientou durante o mestrado. Mesmo não concordando com a rapidez dos acontecimentos, sempre me apoiou. Sua orientação foi importantíssima para o resultado deste trabalho.

Aos professores Igor Moraes e Flávia Delicato, que gentilmente aceitaram participar da minha banca de defesa de mestrado.

Aos professores do instituto de computação, que, em sua maioria, desempenharam essa função com a dedicação que ela merece. Sou muito grato pelo conhecimento e pela experiência que adquiri através de suas aulas ao longo dos anos.

Aos funcionários do instituto de computação e da Universidade Federal Fluminense como um todo.

Ao CNPq pela ajuda financeira, sem a qual, não teria sido possível ir até o fim nos meus objetivos.

Por fim, agradeço a todos aqueles que de alguma maneira, direta ou indiretamente, contribuíram para a realização deste trabalho.

Resumo

A Rede Centrada em Conteúdo constitui uma proposta para a Internet do futuro, onde os dados são identificados e requisitados pelo nome. Qualquer repositório que armazene um conteúdo com este nome pode responder ao usuário. Para garantir a integridade e autenticidade, os conteúdos são assinados por seus publicadores. Apesar disso, verificar a assinatura de todos os conteúdos nos roteadores introduz um *overhead* de processamento significativo. Assim, a verificação de assinaturas pelos roteadores é opcional e, por padrão, não é realizada. Como consequência, publicadores maliciosos podem criar versões poluídas dos conteúdos e estas poderão ser encaminhadas até o consumidor. Por sua vez, os consumidores verificam a assinatura de todos os conteúdos que recebem e, por isso, podem descartar facilmente qualquer conteúdo poluído. Entretanto, se apenas conteúdos poluídos forem recebidos pelo consumidor, o serviço de recuperação de conteúdos oferecido pela CCN será negado a ele. Além disso, o encaminhamento de conteúdos poluídos na rede causa o desperdício de seus recursos. Para mitigar este problema, uma possível solução seria fazer com que os roteadores da rede verificassem probabilisticamente a assinatura dos conteúdos. Assim, este trabalho propõe e analisa o `CCNCheck`, um mecanismo implementado na pilha de protocolos da CCN que realiza a verificação de assinaturas de acordo com determinada probabilidade. Como resultado, foi possível concluir que em todos os cenários avaliados, o `CCNCheck` permitiu que os consumidores recuperassem uma fração maior dos conteúdos requisitados e proporcionou a redução do desperdício de recursos causado pelo encaminhamento de conteúdos poluídos.

Palavras-chave: Rede Centrada em Conteúdo, Poluição de Conteúdos, Verificação Probabilística de Assinaturas.

Abstract

The Content-Centric Networking is a proposal for the future Internet, where the data is identified and requested based on its name. Any node that stores a content with the same name can answer to the user. To ensure the data's integrity and authenticity, all contents in the network are digitally signed by their publishers. Nevertheless, the requirement for routers to check the signature of all contents imposes a significative processing overhead. For this reason, the signature verification is optional for routers, and it is not executed by default. Based on this behavior, malicious producers can create polluted versions of contents, that can be forwarded to the consumer. Then consumers, in turn, check the signature of all contents received, and because of this, they can drop any polluted version easily. However, if only polluted contents are received by the consumer, the content retrieval service offered by the CCN will be denied. Furthermore, the resources used to forward useless polluted contents will be wasted. To mitigate this problem, one possible solution would be to have routers checking the contents' signature probabilistically. Therefore, this work proposes and analyses `CCNCheck`, a mechanism implemented within the CCN protocol stack. `CCNCheck` provides the signature verification according to a certain probability. We concluded that in all evaluated scenarios, `CCNCheck` allows consumers to receive a greater fraction of requested contents and reduce the wastage of network resources caused by the forwarding of polluted versions.

Keywords: Content-Centric Networking, Content Pollution, Probabilistic Signature Verification.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CCN	:	Content-Centric Networking;
CDN	:	Content Distribution Networks;
DDoS	:	Distributed Denial of Service;
DHT	:	Distributed Hash Table;
DNS	:	Domain Name System;
CS	:	Content Store;
FIB	:	Forwarding Information Base;
IP	:	Internet Protocol;
ISP	:	Internet Service Provider;
LAN	:	Local Area Network;
P2P	:	Peer-to-Peer;
PGP	:	Pretty Good Privacy;
PIT	:	Pending Interest Table;
RTT	:	Round Trip Time;
SDN	:	Software Defined Network;
SDSI	:	Simple Distributed Security Infrastructure;
SPKI	:	Simple public key infrastructure;
TCP	:	Transmission Control Protocol;
URL	:	Uniform Resource Locator;

Sumário

Capítulo 1

Introdução

A Internet é, sem dúvidas, uma das criações tecnológicas mais bem sucedidas do último século. Sua popularização permitiu que a comunicação ultrapassasse as barreiras do espaço e do tempo de uma maneira nunca antes vista. Atualmente, pessoas do mundo inteiro trocam mensagens instantâneas, fazem *download* de músicas e vídeos e utilizam serviços como o *e-commerce*. Em 1967, ano em que a então, ARPANET, dava seus primeiros passos, era impossível prever a quantidade e a variedade de aplicações que são proporcionadas pela Internet atualmente. O objetivo naquela época era basicamente interconectar computadores para permitir o compartilhamento de arquivos e programas científicos e militares. O *E-mail*, por exemplo, criado em 1972, foi a primeira aplicação de grande relevância da Internet e seu principal objetivo era facilitar a coordenação dos desenvolvedores da ARPANET [14].

A criação do TCP - *Transmission Control Protocol* e a popularização de computadores pessoais e das LANs - *Local Area Networks* nos anos 80, permitiu que cada vez mais redes fossem adicionadas à Internet, aumentando seu tamanho e a complexidade de seu gerenciamento. Para se comunicar com uma estação remota, era necessário saber o endereço IP - *Internet Protocol* da mesma. Com tantos *hosts* conectados à rede, decorar o endereço IP de um ou mais deles se tornou impraticável. Como resultado, *hosts* passaram a ser nomeados e foi criado o DNS - *Domain Name System* para fazer a tradução desses nomes para endereços IP.

Com o passar dos anos e do crescimento contínuo da Internet, o perfil dos usuários e das aplicações mudou. Atualmente, a Internet é utilizada, principalmente, para geração, compartilhamento e recuperação de conteúdos [19, 21]. Os usuários não estão mais interessados em acessar determinado servidor na rede, mas sim nos conteúdos em si. Aplicações como o *Youtube* e o *Facebook*, onde os consumidores também são publicadores

de conteúdo, conhecidos como *prosumidores*, ilustram este fato. Apesar dessa mudança de paradigma, a arquitetura da Internet não foi alterada. Por essa razão, para obter determinado conteúdo, os usuários ainda são obrigados a conhecer a localização do seu repositório, estabelecer uma comunicação *host-to-host* com o servidor para só então poder requisitar o conteúdo desejado. Em suma, enquanto os usuários se tornam cada vez mais orientados à conteúdos, a Internet se mantém orientada à localização.

Neste contexto em que, pouco importa aos usuários a origem dos dados recebidos, mas sim se o conteúdo é de fato o que ele busca, foram desenvolvidas algumas soluções que funcionam como uma camada entre as necessidades do usuário e a arquitetura da internet. As redes P2P - *Peer-to-Peer*, por exemplo, criam uma topologia sobreposta à topologia física da Internet. Nela, os nós, chamados de *peers*, colaboram entre si fornecendo parte de seus recursos (largura de banda, espaço em disco, etc.) durante o processo de recuperação de conteúdos. Em geral, as redes P2P são descentralizadas e utilizam estruturas de dados distribuídas, como DHTs - *Distributed Hash Tables*, para prover a busca e a recuperação de conteúdos. O primeiro passo para requisitar um conteúdo é buscá-lo na rede. Esta busca pode ser feita com base em alguma palavra chave, ou em alguma característica do conteúdo desejado. Muitos resultados poderão ser obtidos e o usuário deverá escolher dentre eles baseando-se em alguma métrica. Em geral, o usuário sabe quantos são os *peers* que possuem determinado conteúdo, mas não quem são eles. Ao receber o conteúdo, o usuário não sabe de que nó ele se originou. Na realidade, o conteúdo pode, inclusive, ser recuperado de mais de um *peer* ao mesmo tempo.

Outro exemplo de solução para esse paradigma em que a Internet atual se encontra, onde o conteúdo em si é o ente mais importante e não a sua origem, são as CDNs - Content Distribution Networks [19]. Tal solução visa a redução do tráfego entre ISPs, o aumento da disponibilidade dos conteúdos e a diminuição do tempo de recuperação de dados para os usuários. Para tanto, réplicas dos conteúdos são distribuídas por diversos pontos da rede, incluindo diferentes ISPs - *Internet Service Providers*. As requisições dos usuários são então redirecionadas para algum repositório contendo uma réplica do conteúdo desejado. Para qual repositório a requisição deve ser redirecionada é uma decisão que depende de alguns fatores, como por exemplo, a proximidade com o usuário.

A complexidade na adaptação das necessidades atuais dos usuários à Internet tem impulsionado a pesquisa no sentido de uma nova arquitetura para a Internet do futuro. Dois dos principais requisitos para tais arquiteturas são a flexibilidade e a adaptabilidade. A nova Internet precisa comportar diversos tipos diferentes de aplicações e se manter

atual mesmo com a evolução da demanda dos usuários. A maior parte das propostas para a Internet do futuro pode ser dividida em dois conjuntos: arquiteturas puristas e pluralistas. As arquiteturas puristas são monolíticas e representam uma ruptura com a arquitetura atual da Internet. Por outro lado, as arquiteturas pluralistas partem da premissa de que seria muito difícil implementar uma arquitetura flexível e adaptável ao ponto de não necessitar ser alterada ao longo dos próximos anos. Por essa razão, as arquiteturas pluralistas são, em geral, definidas como várias arquiteturas virtuais implementadas sobre uma mesma infraestrutura física. Neste caso, a migração da Internet atual para a nova arquitetura seria facilitada, pois esta poderia ser mais uma arquitetura virtual suportada. Como exemplo de arquitetura pluralista, é possível citar a SDN - *Software Defined Network*. É dentro do contexto das arquiteturas pluralistas que se enquadra a CCN - *Content-Centric Networking* [12, 11].

O objetivo da CCN - *Content-Centric Networking* [12, 11] é trazer o foco da rede para os conteúdos compartilhados e não para a localização física dos mesmos. Assim, para requisitar um conteúdo, não é necessário saber em que local ele está armazenado, mas somente seu nome. Analisando o cenário geral, é possível se perguntar qual a diferença entre a CCN e as redes P2P, já que no fim das contas, ambas oferecem um serviço de recuperação de conteúdos desacoplado da localização física destes. As redes P2P fornecem apenas uma abstração de orientação ao conteúdo para os usuários. Na prática, a Internet continua a oferecer o serviço orientado a localização costumeiro. Além disso, nas redes P2P somente os sistemas finais compartilham conteúdos. Já a arquitetura proposta para CCN fornece um serviço orientado ao conteúdo na camada de rede, onde os dados trocados são encaminhados com base em seu nome, e não em endereços IP. Somente os sistemas finais consomem e publicam conteúdos, mas todos os nós da rede podem armazená-los em *cache* e, portanto, podem compartilhá-los com o resto da rede.

Além de alinhar a arquitetura da rede às necessidades dos usuários quanto a recuperação de conteúdos, a CCN, como uma arquitetura proposta para a Internet do futuro, também precisa fornecer mecanismos para que os dados possam ser trocados de maneira segura. Dessa forma, para garantir a integridade e autenticidade dos dados, todos os conteúdos disponibilizados na rede carregam consigo uma assinatura digital gerada por seu publicador. Além disso, o crescimento do número de ataques de negação de serviço distribuídos (DDoS - *Distributed Denial of Service*) bem sucedidos praticados contra a Internet atual, tem evidenciado a necessidade do núcleo da rede prover algum conjunto mínimo de mecanismos para a mitigação de tais ataques [17]. Neste sentido, a CCN emprega a técnica de agregação de pacotes, o que impede que múltiplas requisições para o

mesmo conteúdo inundem a rede.

1.1 Motivação

Apesar da CCN implementar mecanismos que mitigam alguns ataques comumente executados contra a Internet atual, muitos deles podem ser adaptados para o contexto da orientação ao conteúdo. Por outro lado, ataques de poluição de conteúdo, muito comuns em redes P2P [16, 15, 13], podem ser empregados diretamente na CCN. Entretanto, a eficácia e o objetivo destes ataques são diferentes nestas arquiteturas. O aumento na disponibilidade dos conteúdos, proporcionado pela realização de *cache* por todos os nós da rede, vale tanto para conteúdos legítimos quanto para poluídos. Sem uma maneira de diferenciá-los, os roteadores encaminharão prontamente conteúdos poluídos até o usuário, propagando a poluição por todo o caminho percorrido pelos dados.

Na realidade, o fato dos conteúdos serem assinados, fornece o ferramental necessário para que qualquer nó o identifique como poluído ou legítimo. Assim, se todos os roteadores verificarem a assinatura dos conteúdos antes de armazená-los em *cache* e encaminhá-los, então a poluição seria contida já na infraestrutura da rede e o usuário não seria afetado. Como resultado, o problema da poluição estaria resolvido. Entretanto, a verificação de assinaturas pelos roteadores demanda recursos que são compartilhados com o processamento e encaminhamento de pacotes. Consequentemente, dependendo da carga de tráfego em determinado roteador, verificar a assinatura de todos os dados recebidos por ele é inviável [9]. Por essa razão, somente os consumidores de conteúdo são obrigados a verificar a assinatura dos dados que recebem[1]. Nos roteadores, tal verificação é opcional e por padrão não executada.

Assim, é possível que ao requisitar um conteúdo, o consumidor obtenha como resposta uma versão poluída e inútil. Porém, como os consumidores sempre verificam os conteúdos que recebem, qual é então o efeito da poluição de conteúdos para os mesmos? Dependendo da disseminação de conteúdos poluídos pela rede, pode ocorrer do consumidor não conseguir recuperar qualquer conteúdo legítimo, caracterizando assim um ataque de negação de serviço. Além disso, o encaminhamento de conteúdos poluídos causa o desperdício de recursos da rede que poderiam ser utilizados para transportar carga útil.

[1]Para que os consumidores possam verificar a assinatura dos conteúdos, é necessário conhecer a chave pública do publicador. Essa chave pode ser obtida com o próprio conteúdo, ou recuperada utilizando-se algum nome especial. No contexto deste trabalho, assume-se que a chave pública do consumidor é sempre conhecida.

1.2 Objetivo

Uma vez que a verificação de todos os conteúdos pelos roteadores não é viável, objetivo deste trabalho é propor e analisar um mecanismo para mitigação de ataques de poluição de conteúdos baseado na verificação probabilística de assinaturas pelos roteadores.

1.3 Metas

Para este trabalho, as seguintes metas foram traçadas:

i) Propor e implementar o `CCNCheck`, um mecanismo que realiza a verificação probabilística de assinaturas. De acordo com determinada probabilidade, o roteador deve verificar a assinatura de um conteúdo antes de encaminhá-lo;

ii) Definir abordagens de utilização para o `CCNCheck`. Na abordagem mais simples, todos os nós da rede verificam conteúdos com a mesma probabilidade p. Outra possibilidade é configurar os roteadores de borda para verificarem conteúdos com probabilidade alta p_{borda}, enquanto os demais roteadores verificam conteúdos com o mesmo valor pequeno p. Na terceira abordagem, o valor de p_{borda} é calculado dinamicamente, de acordo com o nível de poluição experimentado pelo roteador;

iii) Propor um mecanismo simples para o cálculo do valor dinâmico de p_{borda};

iv) Analisar a influência da topologia da rede no desempenho do `CCNCheck`;

v) Avaliar o desempenho do `CCNCheck` nas abordagens de utilização propostas.

1.4 Organização do Texto

Este trabalho está organizado da seguinte forma: o Capítulo 2 fornece uma visão geral sobre a arquitetura da CCN, seus mecanismos de segurança e possíveis ataques que podem ser executados em sua arquitetura. Além disso, este capítulo analisa os ataques de poluição de conteúdos na CCN, o qual constitui o problema central deste trabalho. No Capítulo 3, são apresentadas três versões diferentes para o mecanismo `CCNCheck`, além de três abordagens para sua utilização. No Capítulo 4, o modelo de simulação utilizado para avaliar o `CCNCheck` é descrito. O Capítulo 5 apresenta uma análise matemática da influência da topologia na eficiência do `CCNCheck`. No Capítulo 6, são analisados os

resultados obtidos através da execução de simulações para as três abordagens de utilização do `CCNCheck` propostos no Capítulo 3. Por fim, no Capítulo 7, são apresentadas as conclusões e as perspectivas futuras deste trabalho.

Capítulo 2

Segurança em CCN

A Internet atual tem sido alvo de diversos tipos de ataque ao longo dos anos. Quando as tecnologias que a suportam foram criadas, não havia preocupação com a segurança, já que as aplicações não eram críticas e os usuários da rede eram basicamente acadêmicos com o conhecimento técnico adequado. Com sua popularização, esse cenário mudou e a Internet se mostrou vulnerável à criatividade de usuários mal intencionados.

Devido a experiência adquirida com a Internet, a proposta da CCN já traz nativamente alguns mecanismos de segurança. Apesar disso, como é de costume, novas tecnologias tendem a trazer novas vulnerabilidades e oportunidades de ataque e com a CCN não é diferente. Este capítulo versa sobre visão geral do funcionamento da CCN, seus mecanismos de segurança e ataques propostos para esta nova arquitetura de rede.

2.1 Visão Geral da Arquitetura

Para alcançar seus objetivos, a CCN implementa algumas políticas, mecanismos e estruturas que a tornam bastante distinta das redes IP atuais. De maneira geral, seu funcionamento pode ser dividido em três categorias: nomeação de conteúdos, requisição/resposta e características de segurança.

2.1.1 Nomeação de Conteúdos

Na CCN, os roteadores utilizam os nomes de conteúdo para consultarem sua tabela de encaminhamento e saberem por quais interfaces um determinado pacote pode ser encaminhado. Para que seja possível interpretar e extrair informações desses nomes, os roteadores precisam conhecer o padrão de nomeação utilizado. No padrão adotado pela

CCN, conteúdos são nomeados utilizando-se uma estrutura semelhante às URLs - *Uniform Resource Locator*.

Os nomes são formados por um conjunto de componentes separados entre si pelo caractere "/", representando uma hierarquia [12]. Por exemplo, suponha que um grupo de alunos queira obter o vídeo da primeira aula da disciplina de Segurança de Redes. Assim, o Instituto de Computação da Universidade Federal Fluminense poderia publicá-lo através do nome `/br.uff/ic/segredes/aulas/aula1.mp4`.

É importante notar que, na ausência de motores de busca, os usuários precisam conhecer os nomes dos conteúdos que desejam requisitar. Dessa maneira, é importante que os nomes reflitam o significado dos conteúdos. Por exemplo, considerando que o conteúdo a ser recuperado é um filme do `Batman`, não faz sentido nomeá-lo como `/com.dccomics/Superman`.

A utilização desse modelo de nomeação torna fácil o estabelecimento de relacionamentos entre fragmentos de um mesmo conteúdo. Por exemplo, o terceiro segmento da primeira versão do vídeo `intro.avi` poderia ser nomeado como `/br.uff/videos/intro.avi/1/3`. Dessa forma, uma requisição para o conteúdo `intro.avi` poderia remeter ao primeiro segmento deste conteúdo e usando informações contidas neste, juntamente com o conhecimento do método de nomeação utilizado pelo publicador, o consumidor poderia então requisitar os segmentos seguintes.

2.1.2 Requisição/Resposta

A recuperação de conteúdos na CCN é baseada no modelo de requisição/resposta, onde entidades, conhecidas como **publicadores**, disponibilizam conteúdos na rede e os clientes, conhecidos como **consumidores**, requisitam esses conteúdos. Assim, para recuperar determinado conteúdo, foram definidos dois tipos de pacote, um para transportar a requisição e outro para o conteúdo propriamente dito. Esses pacotes recebem o nome de **pacotes de interesse**, doravante chamado apenas de interesse, e **pacotes de dados**, respectivamente.

Visando proporcionar uma maior disponibilidade dos dados na rede, além de um menor tempo de resposta, a CCN estabelece que todos os nós, incluindo os roteadores, devem fazer *cache* de conteúdos. Ao receber um pacote de dados, o nó o armazena em uma estrutura conhecida como `CS` - *Content Store*. Futuras requisições para este conteúdo podem ser respondidas diretamente utilizando o conteúdo armazenado no `CS`.

Além do `CS`, os nós da CCN também possuem outra estrutura muito importante conhecida como `PIT` - *Pending Interest Table*. A PIT pode ser entendida, de maneira simplificada, como uma tabela indexada por nomes de conteúdo. Cada uma de suas entradas guarda uma lista de interfaces por onde interesses para um mesmo conteúdo foram recebidos. Ao receber um pacote de dados, o roteador utiliza o nome de conteúdo para indexar a PIT e obter a lista de interfaces por onde interesses para este conteúdo foram recebidos. O pacote de dados é então encaminhado a todas as interfaces contidas nesta lista. Como consequência, os pacotes de dados retornam aos consumidores seguindo o caminho reverso do interesse que os gerou.

Devido à utilização da PIT, o roteador, ao encaminhar um interesse, tem a certeza de que, se o conteúdo requisitado existir, então o pacote de dados resultante irá passar por ele em seu caminho de volta para o consumidor. Por essa razão, a CCN realiza a agregação de pacotes de interesse, onde ao receber dois ou mais interesses para o mesmo conteúdo, o roteador adiciona as interfaces de entrada destes na PIT, mas somente um dos interesses é encaminhado para o próximo salto. Assim, quando o pacote de dados com o conteúdo requisitado for recebido pelo roteador, todos os interesses para tal conteúdo serão atendidos de uma só vez.

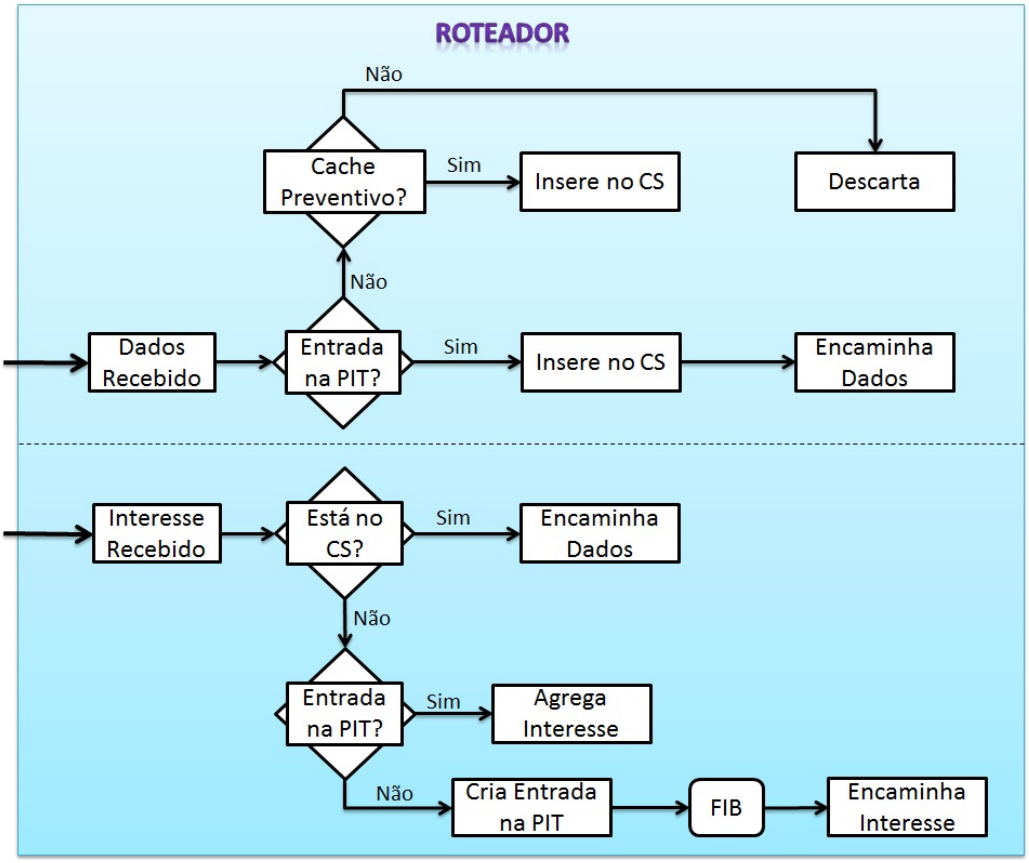

Figura 2.1: Processamento dos pacotes de dados e interesse pelos roteadores da rede.

A Figura 2.1 ilustra um diagrama do processamento dos pacotes de dados e de interesse realizado pelos roteadores da rede. Para requisitar um conteúdo, o consumidor precisa conhecer seu nome, que é então inserido em um interesse e enviado à rede. Ao receber um interesse, o roteador realiza uma busca em seu `CS` utilizando o nome de conteúdo informado no pacote. Se uma correspondência for encontrada, o conteúdo é enviado para o consumidor. Caso contrário, o roteador verifica se já existe um registro na `PIT` correspondente ao nome do conteúdo requisitado. Se tal registro for encontrado, o roteador simplesmente adiciona a interface de entrada do interesse à lista de interfaces contida do registro. Em seguida, o interesse é descartado. Se o registro não existir, ele é criado já contendo a interface de entrada do interesse. Neste caso, o interesse não é descartado e deve ser encaminhado de acordo com a tabela de encaminhamento do roteador. Tal tabela é conhecida, no jargão da CCN, como `FIB` - *Forwarding Information Base*.

Diferentemente das tabelas de encaminhamento das redes IP, as `FIBs` da CCN associam prefixos de nome de conteúdo a múltiplas interfaces de saída, como ilustrado na Figura 2.2. Para evitar a formação de *loops*, devido a possibilidade de encaminhamento de interesses por múltiplas interfaces, tais pacotes possuem um número de sequência (*nonce*) que é utilizado para diferenciar novos interesses de interesses anteriormente recebidos.

Forwarding Information Base (FIB)	
Nome de Conteúdo	Interfaces de Saída
/br.uff	0, 1, 2
/br.ufrj	1, 2

Figura 2.2: Entradas na FIB de um roteador.

Ao receber um pacote de dados, o roteador utiliza o nome de conteúdo contido neste para indexar a `PIT` e obter a lista de interfaces de entrada correspondente. Se nenhum registro for encontrado, então o pacote de dados recebido não fora requisitado ou já fora recebido anteriormente. Neste caso, é possível armazenar tal pacote no `CS` do roteador de forma preventiva e depois descartá-lo ou simplesmente descartá-lo imediatamente. Caso o registro seja encontrado, o pacote de dados é armazenado no `CS` e encaminhado por todas as interfaces contidas no registro da `PIT`.

2.1.3 Características de Segurança

Para garantir a integridade e a autenticidade dos conteúdos trocados na rede, a CCN requer que todos os publicadores de conteúdo possuam um par de chaves, uma mantida

secreta (k^-) e a outra pública (k^+). Para publicar um conteúdo, o publicador deve antes assiná-lo e fornecer esta assinatura, juntamente com o conteúdo em si, para o consumidor através do pacote de dados. Uma vez que os conteúdos são requisitados pelo nome e os nomes são criados a gosto do publicador, é necessário estabelecer, de alguma maneira, uma relação forte entre o conteúdo e seu nome. Na CCN, essa relação forte é alcançada fazendo com que o publicador assine não somente o conteúdo, mas a associação entre o conteúdo e seu nome [23]. A Figura 2.3 exemplifica este processo, onde a associação entre o conteúdo e seu nome é estabelecida através da concatenação de ambos. Esta associação é então assinada, transformando-a em uma relação forte.

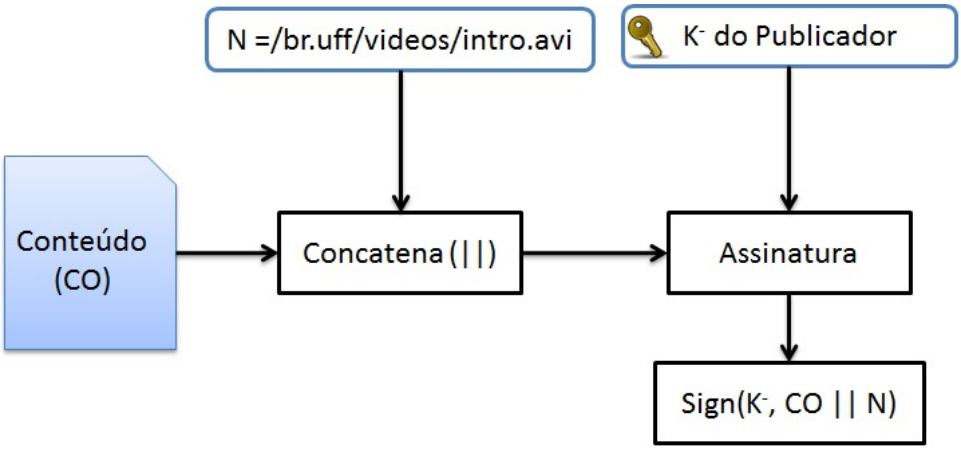

Figura 2.3: Exemplo do estabelecimento de uma relação forte entre o conteúdo e seu nome.

Para que o consumidor possa verificar a assinatura presente em um pacote de dados recebido, é necessário que ele obtenha, de alguma forma, a chave k^+ do publicador. Essa informação pode ser fornecida através do próprio pacote de dados. Após verificar a assinatura, o consumidor tem a garantia de que o conteúdo obtido é íntegro, mas sua autenticidade ainda precisa ser comprovada. Para tanto, é necessário o estabelecimento de um mecanismo de gerenciamento de confiança que associe a chave pública do publicador com sua identidade no mundo real [1].

Outra característica de segurança importante na CCN é sua proteção natural contra os ataques de negação de serviço comumente praticados na Internet atual. Em tais ataques, o objetivo do atacante, em geral, é inundar um servidor com milhares de requisições. Consequentemente, os servidores podem começar a ignorar requisições de usuários legítimos, ou em casos mais graves, parar de responder completamente. Para realizar tais

[1] A maneira pela qual k^+ é obtida, além de como sua autenticidade é garantida está fora do escopo deste trabalho. Entretanto, existem diversas técnicas conhecidas na literatura que podem ser utilizadas para tanto. Exemplos de tais técnicas são o PGP [26], X.509 [18] e SDSI/SPKI [3, 8]

ataques na CCN, os atacantes só têm à disposição os pacotes de interesse e de dados. Ataques de inundação baseados em pacotes de dados não são possíveis, pois pacotes de dados somente são enviados em resposta a um interesse. Por outro lado, ataques baseados em interesses possuem três dificuldades principais:

1. pacotes de interesse não carregam informações de origem ou destino. Por essa razão, não é possível, ao menos diretamente, direcionar o ataque a um alvo específico;
2. pacotes de interesse para um mesmo conteúdo são agregados nos roteadores. Como consequência, mesmo enviando muitos pacotes de interesse, é possível que apenas um deles seja recebido pelo publicador.
3. os *caches* da rede podem fazer com que um interesse seja atendido antes de chegar no publicador.

Assim, para efetuar ataques de negação de serviço na CCN é necessário utilizar outras técnicas mais adaptadas à sua arquitetura. A Seção 2.2 discute brevemente estas técnicas e outras utilizadas em diferentes tipos de ataque contra a CCN.

2.2 Ataques e Propostas de Contramedida

Apesar de resolver alguns dos problemas de segurança que permeiam a Internet atual, a CCN apresenta novas vulnerabilidades e oportunidades de ataque. Nesta seção serão discutidos brevemente alguns ataques propostos para a CCN, principalmente o ataque de poluição de conteúdos que é o problema central deste trabalho.

2.2.1 Censura de Conteúdos

Para descrever este ataque, considere que o atacante não é um usuário, mas sim um país ditatorial com o controle de todo o núcleo da rede. Isso significa que é possível para o atacante, monitorar todas as atividades de rede em seu domínio geográfico. Suponha também que o atacante queira proibir a distribuição em massa de alguns conteúdos alvo e também queira determinar quais usuários os estão requisitando. Além disso, o ataque deve ocorrer em tempo real, ou seja, a censura de conteúdos deve ocorrer ao mesmo tempo em que os pacotes de interesse e de dados trafegam na rede. Neste contexto, *Arianfar et al.* propõem dois tipos de ataque [5]:

Ataque baseado em lista: o atacante possui uma lista negra de nomes de conteúdo que ele deseja censurar. Quando pacotes de interesse ou dados são capturados, o adversário utiliza o nome de conteúdo contido neles para fazer uma busca na lista negra e verificar se existe alguma correspondência. Se existir, o adversário saberá que algum usuário, ou grupo de usuários, está acessando um conteúdo classificado como sensível.

Ataque de análise de conteúdo: neste ataque, o adversário analisa os pacotes de dados capturados em busca de características e padrões que os identifiquem como passíveis de censura.

Para mitigar os ataques baseados em lista e de análise de conteúdo, foi proposto um processo que aumenta a privacidade dos consumidores mesmo na presença de um atacante com controle pleno da rede. Primeiramente o conteúdo T a ser publicado é dividido em n blocos: $T = t_1, t_2, ..., t_n$. Esses blocos são então reordenados aleatoriamente por uma função $r(.)$. Logo após, escolhe-se um conteúdo C, conhecido como *cover*, que é dividido em m blocos: $C = c_1, c_2, ..., c_m$. Tais blocos também são reordenados aleatoriamente pela função $r(.)$. Tanto os usuários quanto os atacantes conhecem os nomes de ambos os conteúdos. Em seguida, para cada k-tupla formada por blocos do conteúdo T e do *cover* C é realizado uma operação de ou-exclusivo entre os k elementos. Por exemplo, se $k = 2$ e os blocos considerados são $r(t_1), r(t_2), r(c_1)$ e $r(c_2)$, o publicador deverá calcular e publicar os *chunks* $r(t_1) \oplus r(t_2)$, $r(t_1) \oplus r(c_1)$, $r(t_1) \oplus r(c_2)$, $r(t_2) \oplus r(c_1)$, etc. Para obter um conteúdo, o consumidor precisa recuperar um conjunto de *chunks* cuja execução da operação de ou-exclusivo entre todos os seus elementos retorne o conteúdo desejado.

Os nomes dos blocos aleatórios e dos *chunks* gerados são computados através da seguinte metodologia:

- O nome $n(t, i)$ para o bloco t_i é $n(t, i) = H(H(T), i)$, onde $H(.)$ é uma função *hash* bem conhecida.
- O nome dos *chunks* é calculado executando-se a função *hash* $H(.)$ dos nomes dos blocos constituintes. Assim, o nome do *chunk* $r(t_1) \oplus r(c_1)$ é $H(n(t, 1), n(c, 1))$.

A força dessa abordagem está baseada no fato de que se um atacante capturar pacotes de interesse para *chunks* específicos, ele não saberá que conteúdos estão sendo requisitados, o que dificulta o ataque baseado em lista. Muito embora o adversário possa capturar vários *chunks*, conseguir combinar todos os possíveis *chunks* para obter o conteúdo original é

computacionalmente mais custoso para o atacante do que para os consumidores [5]. Por essa razão, os ataques baseados em análise de conteúdos também são dificultados.

Outra proposta de contramedida que visa aumentar a privacidade dos consumidores é o ANDANA [6]. Este mecanismo utiliza a técnica de roteamento em cebola de maneira semelhante ao Tor [7]. Seu objetivo é impedir que atacantes locais tomem conhecimento sobre que conteúdos determinado consumidor está requisitando.

O funcionamento do Andana está ilustrado na Figura 2.4. Antes de enviar qualquer interesse, os usuários devem escolher dois entre vários roteadores habilitados para o ANDANA[2]. A maneira de fazer essa escolha é livre, mas é preferível que os roteadores sejam escolhidos uniformemente, já que assim, todos os roteadores terão a mesma probabilidade de serem escolhidos, garantindo um maior anonimato [6]. Os dois roteadores escolhidos formam então um circuito efêmero que é utilizado para transportar apenas um ou alguns poucos interesses criptografados. Tal circuito é desfeito quando o conteúdo é recebido ou quando um pequeno período de tempo se esgota. O primeiro roteador nesse circuito é conhecido como Roteador de Entrada (RE) e o segundo como Roteador de Saída (RS). Para dificultar que atacantes consigam relacionar entradas no RE com as saídas no RS, os dois roteadores pertencentes ao circuito devem ser, preferencialmente, de domínios administrativos diferentes.

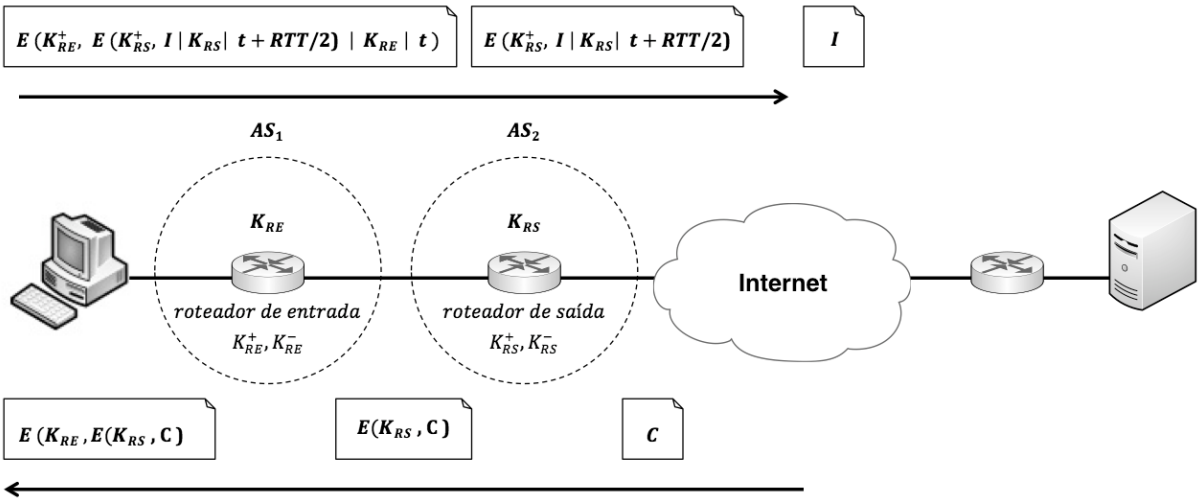

Figura 2.4: Esquema de funcionamento do ANDANA [20].

Uma vez construído o circuito, o usuário deve conhecer as chaves públicas dos dois roteadores que o compõem. Sejam K^+_{RE} a chave pública do roteador RE e K^+_{RS} a chave pública do roteador RS. Dessa forma, um interesse é primeiramente criptografado com

[2]Roteadores habilitados para o Andana são apenas roteadores de conteúdo da CCN que executam os procedimentos definidos pelo Andana

K^+_{RE} e em seguida novamente com K^+_{RS}. Após este processo, o interesse terá duas camadas de criptografia e um adversário local não será capaz de saber o que está sendo requisitado. Cada roteador no circuito, além das funções tradicionais correspondentes aos roteadores de conteúdo, deve decifrar o interesse utilizando sua chave privada. Dessa forma, a cada salto no circuito, uma camada de criptografia é removida até que o interesse seja encaminhado pelo roteador RS em texto claro, da maneira como a CCN espera.

Para garantir a privacidade dos usuários, além dos pacotes de interesse, os pacotes de dados também precisam trafegar de maneira criptografada na rede local. Dessa forma, cada roteador deverá fazer o processo inverso, ou seja, introduzir uma camada de criptografia no pacote de dados. Para tanto, é necessário que o usuário estabeleça um segredo compartilhado com cada roteador no circuito. Estes segredos são as chaves criptográficas K_{RE} e K_{RS}, geradas aleatoriamente pelo usuário. Para realizar a distribuição desses segredos para os roteadores, o usuário adiciona, em cada camada de criptografia do pacote de interesse, a chave correspondente ao roteador que, após remover a camada anterior, obterá o segredo.

Para exemplificar o processo, suponha que I seja um pacote de interesse para determinado conteúdo C. Suponha também que E^1 seja o algoritmo criptográfico utilizado para criptografar/decifrar interesses e E^2 o algoritmo criptográfico utilizado para criptografar/decifrar pacotes de dados. Para qualquer algoritmo E, $E(K, D)$ significa a cifragem da informação D através do algoritmo E utilizando a chave K.

O roteador RE receberá o interesse I criptografado pelo consumidor da seguinte maneira: $E^1(K^+_{RE}, E^1(K^+_{RS}, (I + K_{RS}) + K_{RS}) + K_{RE})$. Em seguida ele utilizará sua chave privada K^-_{RE} para decifrar o interesse e obterá duas informações. A primeira é o interesse com uma camada a menos de criptografia, ou seja, $E^1(K^+_{RS}, (I + K_{RS})$. Este interesse será encaminhado para o roteador RS. A segunda informação é o segredo K_{RE}, que será utilizado posteriormente para criptografar o pacote de dados. O roteador RS, por sua vez, fará o mesmo processo, ou seja, decifrar o interesse recebido com sua chave privada K^-_{RS} . Mais uma vez, duas informações serão obtidas: o interesse em texto claro I e o segredo K_{RS}.

Uma vez que o pacote de dados C tenha sido recebido pelo roteador RS, ele será criptografado utilizando o segredo K_{RS}. O resultado deste processo, ou seja, $E^2(K_{RS}, C)$, será encaminhado para o roteador RE. Por sua vez, RE criptografará os dados recebidos usando o segredo K_{RE}, o que resultará em $E^2(K_{RE}, E^2(K_{RS}, C))$. Logo após, RE encaminhará os dados para o consumidor, que já possui K_{RE} e K_{RS} e sabe a ordem em que as

camadas de criptografia foram adicionadas. Dessa forma, o consumidor pode, finalmente, decifrar os dados recebidos, camada a camada, até obter C.

2.2.2 *Cache Snooping*

Na Internet, diversos serviços fazem uso de *caches* para aumentar sua eficácia. Conteúdos requisitados com maior frequência são guardados em dispositivos intermediários, de forma que o acesso aos mesmos ocorra mais rapidamente no futuro. Navegadores *web*, por exemplo, armazenam localmente páginas recentemente requisitadas. Consequentemente, futuras requisições para essas páginas são prontamente atendidas, sem a necessidade de serem encaminhadas até servidor *web* original.

Apesar de aumentar a eficiência na recuperação de conteúdos, a utilização de *caches* também pode introduzir problemas à privacidade dos usuários. Os conteúdos armazenados em um determinado *cache* podem ser vistos como um histórico de requisições dos usuários associados a ele. Por essa razão, se um usuário malicioso tiver acesso a essa informação, ele será capaz de inferir que conteúdos os usuários associados ao *cache* estão requisitando. A técnica de tentar extrair rastros de comunicação dos *caches* para inferir informações sobre seus usuários é conhecida como *cache snooping*.

A CCN expande a utilização de *caches* para todos os roteadores da rede. Dessa forma, é natural imaginar que se o *cache snooping* é um problema para a Internet atual, ele será ainda mais grave no contexto da CCN. Diferentemente das redes IP, a CCN não registra informações sobre a origem das requisições. Isto implica em uma maior garantia à privacidade dos usuários, já que mesmo que um adversário explore o *cache* para obter informações sobre os conteúdos nele presentes, nenhuma informação sobre quem os requisitou poderá ser extraída. Entretanto, a hierarquia de *caches* produzida pela CCN faz com que um número reduzido de usuários esteja associado a determinado *cache*, aumentando assim sua exposição [22].

2.2.3 Ataques de Negação de Serviço

Em tais ataques, os atacantes geram um grande número de pacotes de interesse para conteúdos específicos, esperando causar o mal funcionamento da infraestrutura da rede ou da fonte de conteúdos. No caso do alvo do ataque ser a infraestrutura da rede, o objetivo do adversário é fazer com que os interesses gerados por ele consumam toda a memória disponível na `PIT` dos roteadores. Dessa forma, assumindo que a CCN implemente a política

de substituição *Tail Drop* [25], os interesses enviados por usuários legítimos encontrarão a PIT com recursos exauridos e serão descartados. Da mesma forma, o objetivo do ataque às fontes de conteúdo também é causar a recusa de interesses legítimos. Entretanto, tais interesses são descartados pelas fontes de conteúdo, e não pelos roteadores da rede. Em ambos os casos, o adversário pode utilizar *botnets* para amplificar a eficiência do ataque.

Os pacotes trafegados na CCN não transportam informações de origem e destino em seus cabeçalhos. Assim, da mesma forma que a rede é orientada ao conteúdo, os ataques também precisam ser. Ao invés de escolher o endereço de destino do alvo, o adversário precisa decidir quais conteúdos requisitar para alcançar seu objetivo. Os ataques de negação de serviço por inundação de interesses podem ser divididos em três categorias, dependendo do tipo de conteúdo requisitado no ataque [9]:

1. Estático ou existente: são conteúdos que foram publicados e estão disponíveis na rede para serem requisitados pelos usuários.
2. Gerado dinamicamente: são conteúdos gerados apenas quando requisitados através de pacotes de interesse.
3. Não existentes: são conteúdos cujos interesses nunca serão atendidos.

Ataques de inundação de interesses do tipo 1 tem efetividade limitada e são mais eficientes se utilizados para atacar a infraestrutura da rede, e não as fontes de conteúdo. Devido a implementação de *caches* de conteúdo nos roteadores, os interesses para um conteúdo estático somente atingirão a fonte em um primeiro momento. Nas próximas requisições, os interesses serão atendidos pelos *caches*.

Os ataques do tipo 2 são mais adequados quando o alvo do atacante é a fonte de conteúdos . Uma vez que os dados são gerados dinamicamente, eles não são armazenados nos *caches* da rede. Consequentemente, interesses para tais conteúdos são sempre roteados até a fonte. Se a geração de conteúdos dinâmicos for computacionalmente custosa, tanto em memória quanto em processamento, dependendo do número de interesses de ataque que a atinjam, a fonte de conteúdos pode se tornar inoperante. Por outro lado, o impacto deste ataque nos roteadores depende da sua proximidade em relação à fonte de conteúdos. Os roteadores mais próximos tendem a manter um maior número de interesses pendentes maliciosos em sua `PIT`, devido a concentração do tráfego destinado à fonte de conteúdos.

Os ataques do tipo 3 poderão ser mais eficientes caso o objetivo do adversário seja afetar a infraestrutura da rede. Uma vez que os conteúdos são inexistentes, os atacantes

tem a garantia de que seus interesses não serão atendidos por nenhum *cahe* da rede. Além disso, se o ataque for realizado de forma que cada interesse gerado requisite um conteúdo inexistente diferente, o atacante também poderá burlar o mecanismo de agregação de interesses da CCN. Consequentemente, cada interesse enviado pelos atacantes será encaminhado até a fonte de conteúdos. Como os dados requisitados são inexistentes, a fonte de conteúdos pode descartá-los rapidamente, evitando assim o desperdício de recursos. Porém, até chegarem ao seu destino, os interesses gerarão entradas maliciosas na `PIT` dos roteadores. Se as entradas da `PIT` forem exauridas pelos interesses maliciosos, nenhum interesse legítimo poderá ser atendido.

A ausência de informações que identifiquem a origem dos pacotes de interesse dificulta o rastreamento dos atacantes. Assim, qualquer usuário pode gerar um grande fluxo de pacotes de interesse e manter seu anonimato. Para contornar este problema, seria possível adotar uma solução onde os consumidores fossem obrigados a assinar os pacotes de interesse por eles gerados. Dessa forma, em um ataque de inundação de interesses, seria possível descobrir a(s) fonte(s) do ataque e tomar as devidas contramedidas. Porém, além de ferir a privacidade dos usuários, esta abordagem não seria completamente efetiva, já que os ataques podem ser distribuídos. A maior parte das propostas de mitigação encontradas na literatura são baseadas na utilização de estatísticas geradas pelos roteadores [9] e em mecanismos de *Push-Back* [10].

2.3 Poluição de Conteúdos

Além dos ataques abordados na Seção 2.2, a CCN também é vulnerável a ataques de poluição de conteúdos, que é o problema central deste trabalho. Devido ao seu paradigma orientado a conteúdo, o principal objetivo da CCN é permitir que os usuários possam obter os conteúdos desejadas de forma mais simples e rápida. Para aumentar a disponibilidade dos dados, todos os roteadores da rede realizam *cache* de conteúdos. Essa característica amplia o conceito das CDNs para uma escala global e permite que os usuários recuperem conteúdos mais rapidamente.

A exemplo do que ocorre nas redes P2P, nem todos os conteúdos publicados na rede são efetivamente legítimos. Atacantes podem aproveitar a alta disponibilidade gerada pela CCN para disseminar conteúdos maliciosos, visando prejudicar o usuário final ou a própria infraestrutura da rede. No contexto deste trabalho, são definidos os seguintes ataques de poluição de conteúdos na CCN:

Renomeação: os conteúdos são publicados com um nome diferente do original. Por exemplo, o atacante poderia publicar um vírus utilizando o nome de um vídeo popular, com o objetivo de ludibriar o consumidor. Para publicar o conteúdo poluído, o atacante precisaria assiná-lo com sua chave privada. Assim, a identificação de conteúdos poluídos gerados a partir deste ataque depende do mecanismo de gerenciamento de confiança utilizado;

Corrupção: os dados são publicados com seus *bits* ou metadados alterados. Dessa forma, os consumidores podem receber conteúdos inúteis como resposta às suas requisições. Esse tipo de ataque viola a integridade dos dados e pode ser facilmente detectado através da verificação da assinatura do conteúdo;

Falsificação: neste caso, existem duas possibilidades. Na primeira, a chave pública informada no pacote de dados não corresponde à chave privada utilizada na assinatura do conteúdo. Na segunda alternativa, a chave pública informada corresponde à chave privada utilizada na assinatura, mas não à entidade do mundo real esperada. Por exemplo, ao requisitar o filme `/com.marvel/avengers.mp4`, é plausível esperar que o conteúdo resultante tenha sido assinado pelo publicador Marvel. Caso a chave pública informada não pertença à Marvel, então o conteúdo poderá ser considerado poluído. Assim como na Renomeação, a identificação desse tipo poluição depende do mecanismo de gerenciamento de confiança empregado.

Como pode ser observado, através da verificação da assinatura dos dados, o consumidor é capaz, na maior parte dos casos, de identificar se um conteúdo recebido é ou não uma versão poluída. Por esta razão, o objetivo dos ataques de poluição de conteúdo não é enganar o usuário, mas sim atrapalhá-lo no processo de recuperação de conteúdos.

Ao ser encaminhado até o consumidor, o conteúdo poluído é armazenado em *cache* pelos roteadores no caminho. Ao receber o conteúdo, o consumidor verifica sua assinatura, detecta que este é uma versão poluída e, por fim, o descarta. O consumidor pode, em seguida, desejar realizar uma nova tentativa de obter uma versão legítima do conteúdo desejado. Entretanto, como o roteador de borda associado ao consumidor já possui a versão poluída do conteúdo em *cache*, é muito provável que o consumidor obtenha a versão poluída novamente. Assim, dependendo do grau de poluição da rede, os usuários podem se ver impedidos de recuperar conteúdos legítimos, resultando em um ataque de negação de serviço. Além disso, o encaminhamento de pacotes de dados com conteúdos poluídos e, portanto inúteis, gera desperdício de recursos, como memória e processamento.

Para evitar que versões poluídas de determinado conteúdo sejam sempre retornadas ao consumidor pelos *caches* da rede, é possível pensar em alguma maneira de filtrar as respostas para determinado interesse. Atualmente, a CCN dispõe de três mecanismos com este objetivo. Trata-se dos campos *AnswerOriginKind*, *Exclude* e *PublisherPublicKeyDigest*, presentes no cabeçalho dos pacotes de interesse.

O campo *AnswerOriginKind* pode ser utilizado para fazer com que nenhum *cache* responda ao interesse. Como resultado, todos os interesses com esta definição são sempre encaminhados até o publicador. Esse campo poderia ser usado para contornar o problema do conteúdo poluído armazenado nos *caches* dos roteadores. Ao receber uma versão poluída, o consumidor envia novamente um interesse para o mesmo conteúdo, mas com o campo *AnswerOriginKind* definido para ignorar os `CSs` dos roteadores. Assim, aumentam-se as chances do consumidor receber uma versão legítima do conteúdo, já que agora o interesse pode ser recebido e respondido por um publicador legítimo. Apesar de seu aparente benefício, o campo *AnswerOriginKind* representa, na verdade, uma vulnerabilidade que pode ser explorada para a realização de ataques de negação de serviço baseados em inundação de interesses. Por essa razão, este campo deve ser limitado ou eliminado em revisões futuras da arquitetura da CCN.

O campo *Exclude* tem o objetivo de filtrar as possíveis respostas para determinado interesse. Para tanto, o consumidor adiciona a este campo, componentes de nome de conteúdo que não devem fazer parte do nome do conteúdo retornado. Por exemplo, se o campo *Exclude* de um interesse para o conteúdo */br.uff/disciplinas/* contiver o componente */redes1*, o conteúdo */br.uff/disciplinas/redes1/aula1.mp4* não será enviado ao consumidor. Em ataques de poluição de conteúdos baseados na corrupção dos dados, por exemplo, o nome do conteúdo original é mantido intacto. Assim, não existe qualquer trecho no nome do conteúdo poluído que o diferencie de sua versão legítima, tornando ineficaz a utilização do campo *Exclude* como contramedida para a poluição de conteúdos.

A função do campo *PublisherPublicKeyDigest* é permitir que o consumidor possa definir que um interesse só seja atendido por um conteúdo publicado pelo publicador especificado. Para tanto, o campo *PublisherPublicKeyDigest* carrega o *hash* (SHA-256) da chave pública do publicador desejado. Por sua vez, os pacotes de dados também contém um campo *PublisherPublicKeyDigest* contendo o *hash* da chave pública do seu publicador. Assim, um pacote de dados só é retornado ao consumidor caso os valores do *PublisherPublicKeyDigest* em ambos os pacotes sejam coincidentes. Na realidade, um publicador malicioso poderia facilmente utilizar a chave pública de um publicador

legítimo para definir o campo *PublisherPublicKeyDigest* nos pacotes de dados por ele gerados. Consequentemente, a utilização deste campo não possui muita utilidade neste contexto. Maiores informações sobre os campos existentes em um interesse podem ser encontradas em [1].

Uma ideia mais interessante do que filtrar respostas para interesses é simplesmente impedir que conteúdos poluídos sejam armazenados em *cache*. Para tanto, é preciso que os roteadores da rede verifiquem a assinatura de cada conteúdo recebido antes de adicioná-los ao seu *CS*. Se o conteúdo for identificado como poluído, o roteador poderia simplesmente descartá-lo e não encaminhá-lo para o próximo salto. Neste caso, como a CCN permite que interesses e pacotes de dados se propaguem por mais de um caminho, a inibição da propagação de um conteúdo poluído pode aumentar as chances do consumidor receber uma cópia legítima.

Através da verificação de assinaturas, o problema da poluição de conteúdos na CCN parece estar resolvido. Entretanto, enquanto os consumidores precisam verificar apenas a assinatura dos conteúdos requisitados por eles, os roteadores da rede possuem uma carga de tráfego muito maior. Devido ao *overhead* inerente ao processo de verificação de assinaturas, é inviável para os roteadores realizarem tal verificação para todos os conteúdos recebidos por eles [9]. Por essa razão, a verificação de assinaturas é mandatória para os consumidores, mas opcional (e por padrão não realizada) para os roteadores.

Uma vez que não é viável para os roteadores verificarem a assinatura de todos os conteúdos recebidos por eles, por que não verificar uma fração destas? Esta é justamente a ideia da verificação probabilística de assinaturas. Os roteadores da rede passam a verificar a assinatura dos conteúdos, mas de acordo com determinada probabilidade. O valor desta probabilidade deve ser definido com cuidado para garantir um balanceamento eficiente entre o *overhead* de verificação e a eficácia na redução da disseminação de conteúdos poluídos.

2.4 Conclusão do Capítulo

O objetivo deste capítulo foi trazer ao leitor uma visão geral sobre o funcionamento da CCN e questões relativas à sua segurança. Na Seção 2.1 foi apresentada uma visão geral sobre o funcionamento da arquitetura da CCN. Foi mostrado como a CCN nomeia conteúdos utilizando um padrão hierárquico com componentes separados pelo caractere "/", muito semelhante às URLs. Em seguida, foi abordado o processo de recuperação de

conteúdos. Usuários requisitam conteúdos pelo nome através de um pacote de interesse. Os roteadores, por sua vez, utilizam o nome do conteúdo para saber por quais interfaces um pacote de dados deve ser encaminhado. Ao retornar para o consumidor, os pacotes de dados seguem o caminho reverso do respectivo interesse. Para que esse comportamento seja possível, os roteadores mantêm informações sobre as interfaces de entrada e saída dos interesses recebidos. Por fim, foram apresentados alguns mecanismos de segurança implementados nativamente na CCN. Todos os conteúdos publicados na rede são assinados por seus publicadores. Como resultado, o consumidor tem a garantia da integridade e autenticidade dos dados que recebe. Na verdade, a autenticidade depende ainda da implementação de um mecanismo de gerenciamento de confiança. A CCN também realiza a agregação de pacotes de interesse, o que evita que interesses para um mesmo conteúdo inundem a rede. Por essa razão, ataques de negação de serviços tradicionalmente executados na Internet são mitigados. Além disso, os pacotes trocados na CCN não possuem informações de origem e destino, o que aumenta a privacidade dos usuários.

Na Seção 2.2 foram abordados três tipos de ataque que podem ser empregados contra a CCN. Na censura de conteúdos, o objetivo do atacante é impedir que determinados conteúdos sejam massivamente distribuídos na rede. Além disso, o atacante também gostaria de descobrir que consumidores requisitaram os conteúdos censurados. O segundo ataque analisado foi o *cache snooping*, onde os atacantes tentam extrair rastros de comunicação dos *caches* para inferir informações sobre os usuários associados a eles. Em seguida, foi apresentado como os ataques de negação de serviço tradicionais podem ser adaptados para o paradigma da CCN. Para alcançar seus objetivos, os atacantes precisam escolher com cuidado que tipo de dados requisitar, tornando os ataques de negação de serviço orientados à conteúdo.

Finalmente, na Seção 2.3 foram abordados os ataques de poluição de conteúdo. Assim como acontece nas redes P2P e em aplicação orientadas a conteúdos de uma maneira geral, a poluição de conteúdos é um assunto relevante também para a CCN. Na realidade, devido a realização de *cache* de conteúdos por todos os nós da rede, a disseminação de conteúdos poluídos é catalizada na CCN. Apesar de suas potenciais implicações, a poluição de conteúdos na CCN ainda é um assunto pouco discutido na literatura. Neste sentido, este trabalho constitui um esforço inicial na busca de uma solução para este problema. O Capítulo 3 apresenta o `CCNCheck`, um mecanismo para mitigação de ataques de poluição de conteúdos baseado na verificação probabilística de assinaturas.

Capítulo 3

CCNCheck

A verificação da assinatura de conteúdos nos roteadores não é realizada por padrão na CCN. Por essa razão, publicadores maliciosos podem inserir na rede versões poluídas de conteúdos populares que poderão ser recuperadas por alguns consumidores. Ao serem encaminhadas na rede, essas versões poluídas são armazenadas no `CS` dos roteadores, aumentando assim sua disponibilidade. Os consumidores, por sua vez, verificam a assinatura de todos os conteúdos que recebem e podem simplesmente descartar as versões poluídas. Entretanto, se um consumidor tentar recuperar novamente um conteúdo recebido poluído anteriormente, existem grandes chances dele obter uma versão poluída novamente oriunda de algum *cache* da rede.

Para impedir a disseminação de conteúdos poluídos na rede, uma solução simples seria fazer com que os roteadores verificassem a assinatura de todos os conteúdos por ele recebidos. Entretanto, esta abordagem não é viável devido ao *overhead* gerado pela verificação de assinaturas. Assim, se os roteadores não verificarem conteúdo algum, conteúdos poluídos poderão se alastrar facilmente na rede. Por outro lado, a verificação de todos os conteúdos pelos roteadores é inviável. Sendo assim, este trabalho propõe uma solução intermediária, onde uma fração aleatória dos conteúdos é verificada pelos roteadores.

A implementação de tal proposta resultou no `CCNCheck`, um mecanismo para o combate à poluição de conteúdos na CCN. Sua abordagem se baseia na verificação probabilística de assinaturas para permitir o aumento do número de conteúdos legítimos recuperados pelos consumidores e a redução do desperdício de recursos da rede introduzido pelo encaminhamento de conteúdos poluídos.

Neste capítulo são propostas três versões para o `CCNCheck`. O `CCNCheck v.1` atua somente quando o roteador recebe um pacote de dados como resposta a um interesse

anteriormente encaminhado. Por outro lado, o `CCNCheck v.2` possui as funcionalidades do `CCNCheck v.1` mais a verificação de conteúdos oriundos do *CS* que seriam utilizados como resposta a algum interesse recebido. Já o `CCNCheck v.3` implementa as funcionalidades do `CCNCheck v.2` mais a verificação periódica de todos os conteúdos em *cache*.

3.1 CCNCheck v.1

O funcionamento do `CCNCheck v.1` é ilustrado na Figura 3.1. Quando o roteador recebe um pacote de dados, este é imediatamente direcionado para o módulo do `CCNCheck`. De acordo com a probabilidade definida, a assinatura do conteúdo pode ou não ser verificada. Se o conteúdo não for verificado, ou se for verificado e classificado como legítimo, o pacote de dados é processado da maneira usual na CCN. Se o conteúdo for classificado como poluído, ele é simplesmente descartado pelo roteador.

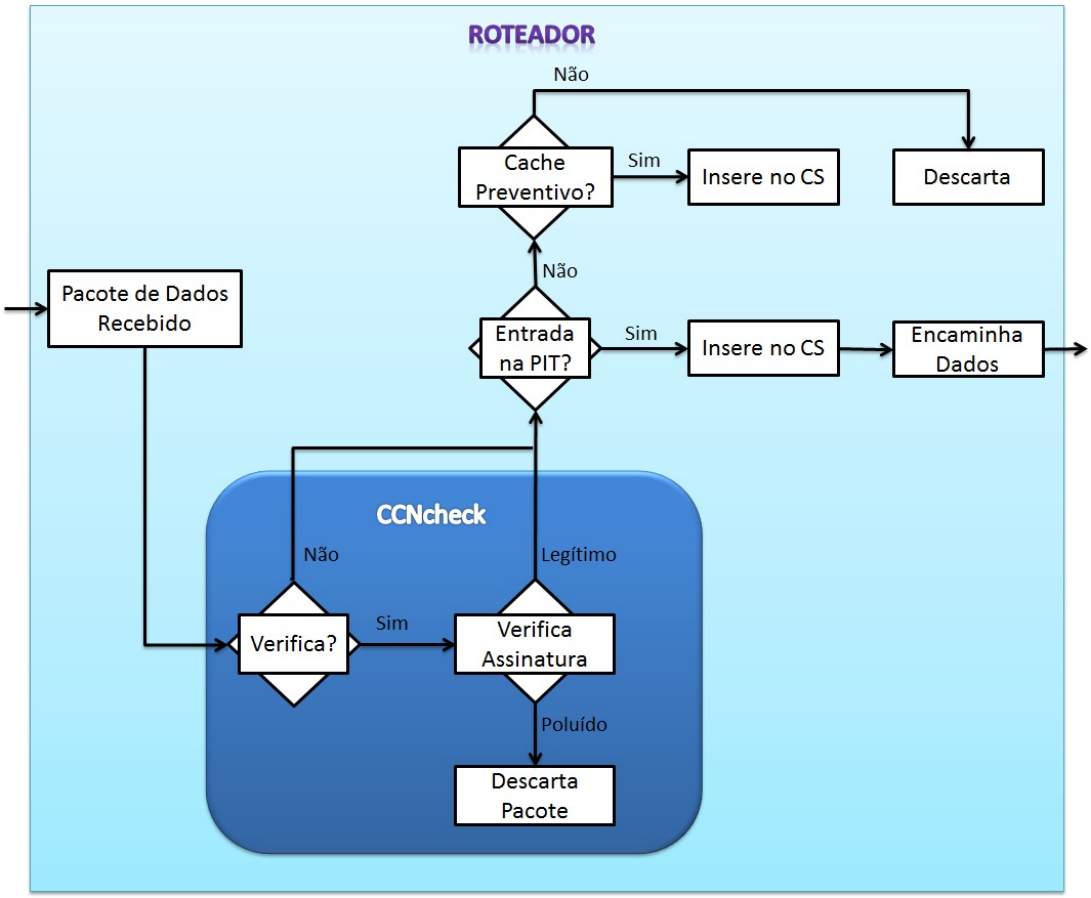

Figura 3.1: Papel do `CCNCheck v.1` no processo de encaminhamento de pacotes.

3.2 CCNCheck v.2

Os conteúdos poluídos não verificados pelo `CCNCheck v.1` são armazenados no `CS` dos roteadores. Como a probabilidade de verificação em cada roteador deve ser baixa para reduzir o *overhead* gerado, é possível que um conteúdo poluído se espalhe por diversos nós da rede. Como exemplo, suponha um roteador de borda ligado a alguns consumidores. Suponha também que um dos consumidores requisitou o conteúdo CO e que ele recebeu uma versão poluída do mesmo. Neste caso, nenhum roteador no caminho verificou CO e, como consequência, todos esses roteadores possuem agora uma versão poluída de CO em seu `CS`. Suponha agora que um segundo consumidor também requisite o conteúdo CO. Neste caso, o roteador de borda irá responder com a versão poluída que está em seu `CS`. Logo, nenhum consumidor ligado ao roteador de borda em questão conseguirá receber uma cópia legítima de CO.

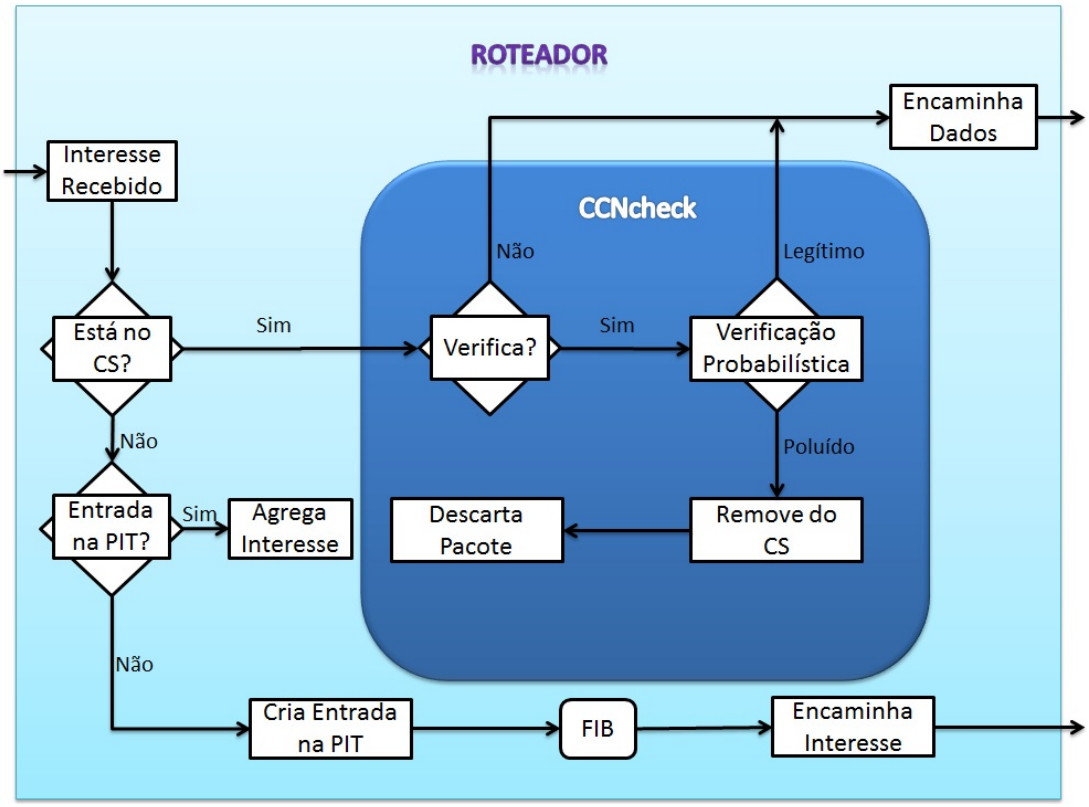

Figura 3.2: Papel do `CCNCheck v.1` no processo de encaminhamento de pacotes.

Pensando nessa dificuldade, foi criada uma segunda versão do `CCNCheck`. Ao receber um pacote de dados, o `CCNCheck v.2` age da mesma maneira que o `CCNCheck v.1` (Figura 3.1). Entretanto, ao receber um pacote de interesse o `CCNCheck v.2` procede da maneira mostrada na Figura 3.2. Se o conteúdo requisitado não estiver em *cache*, então o interesse será processado como de costume pela CCN. Caso contrário, antes de ser encaminhado, o

pacote de dados oriundo do *cache* é direcionado ao módulo do `CCNCheck`. O `CCNCheck v.2` atua sobre os dados recebidos de maneira muito semelhante ao `CCNCheck v.1`, mas quando um conteúdo poluído é verificado, antes de ser descartado, a entrada correspondente no *CS* é removida. Em seguida, uma nova entrada na textttPIT é criada para o interesse, que é então encaminhado de acordo com a `FIB` e com a política de encaminhamento utilizada. Assim, o `CCNCheck v.2` cria oportunidades para que novas tentativas de recuperar uma versão legítima de determinado conteúdo sejam bem sucedidas.

3.3 CCNCheck v.3

Mesmo com a adoção do `CCNCheck v.2`, devido a necessidade de se usar baixos valores para a probabilidade de verificação, é possível que um conteúdo poluído permaneça em *cache* nos roteadores por um longo período. Esse período pode ser longo o suficiente para impedir que boa parte dos consumidores seja capaz de recuperar uma versão válida do conteúdo.

Neste contexto, `Gasti et al.` propõem um mecanismo que realiza a verificação de um subconjunto aleatório dos conteúdos armazenados no `CS` dos roteadores [9]. Os conteúdos legítimos em *cache* são então marcados e não são verificados novamente. Enquanto o objetivo do `CCNCheck` é impedir que conteúdos maliciosos se propaguem pela rede, a proposta de `Gasti et al.` tenta reduzir a disponibilidade dos conteúdos poluídos através da limpeza dos *caches* da rede. Pode-se perceber então que tais abordagens são complementares.

Assim, o `CCNCheck v.3` funciona de maneira semelhante ao `CCNCheck v.2`, mas inclui também uma verificação periódica dos conteúdos armazenados no `CS` dos roteadores. Entretanto, ao invés de implementar a proposta de `Gasti et al.`, o `CCNCheck v.3` supõe que os roteadores possuem dois processadores, um para o processamento de pacotes e o segundo para a verificação dos conteúdos em *cache*. Sendo assim, como os processadores são independentes, a verificação dos conteúdos em *cache* não gera atrasos no processamento dos pacotes. Por essa razão, ao invés de verificar apenas uma fração das entradas do `CS`, o `CCNCheck v.3` verifica o *cache* por completo.

3.4 Implementação da Verificação Probabilística de Assinaturas

Após o pacote de dados ser recebido no módulo do `CCNCheck`, o Algoritmo 1 é executado. Como entrada, são esperados dois parâmetros, a assinatura da associação entre o conteúdo e seu nome e a chave pública do publicador. Como saída, o algoritmo informa se o conteúdo é ou não uma versão poluída. Na linha 2, a função *GetProbability* retorna a probabilidade de verificação de assinaturas para o nó em questão. Essa função pode ser implementada para retornar um valor estático, ou calcular um valor dinâmico, de acordo com as características da rede.

Algoritmo 1: ProbabilisticCheck

Data: Assinatura do conteúdo: $Sign(C, N)$.
Data: Chave pública do publicador: K_{PUB}.
Result: $True$, se o conteúdo for legítimo e $False$, caso contrário.

```
1  begin
2      p ⟵ GetProbability
3      k ⟵ p · (MAX + 1)
4      r ⟵ rand()
5      result ⟵ True
6      if r < k then
7          signatureOk ⟵ CheckSignature(K_PUB, Sign(C, N))
8          publicKeyOk ⟵ CheckPublicKey(K_PUB)
9          result ⟵ signatureOk and publicKeyOk
10     end
11     return result
12 end
```

A variável r representa um valor inteiro aleatório sorteado uniformemente a partir do intervalo $[0, MAX]$. O corpo da instrução IF (linha 6) deve ser executado com probabilidade p. Por essa razão, a probabilidade da variável r ser menor que o valor k ($P_r[r < k]$) deve ser igual a p, como mostrado na Equação 3.1.

$$P_r[r < k] = p \tag{3.1}$$

Assim, para implementar a verificação probabilística de assinaturas é preciso encontrar um valor de k que satisfaça a Equação 3.1. Uma vez que r é uma variável discreta, $P_r[r < k]$ pode ser dada pela Equação 3.2.

$$P_r[r < k] = \sum_{i=0}^{k-1} P_r[r = i] \tag{3.2}$$

Além disso, como r é uma variável aleatória uniforme, a probabilidade de qualquer valor no intervalo $[0, MAX]$ ser sorteado é dada pela Equação 3.3.

$$P_r[r = i] = \frac{1}{(MAX + 1)}, \forall i \in [0, MAX] \tag{3.3}$$

Combinando-se as Equações 3.1, 3.2 e 3.3 na Equação 3.4, é possível calcular o valor de k que satisfaz a Equação 3.1.

$$P_r[r < k] = k \cdot \frac{1}{(MAX + 1)} = p \therefore k = p \cdot (MAX + 1) \tag{3.4}$$

A função *CheckSignature* verifica a assinatura propriamente dita e a função *CheckPublicKey* verifica a autenticidade da chave pública do publicador, de acordo com o mecanismo de gerenciamento de confiança implementado. Se ambas as verificações retornarem *True*, então é posssível concluir que o conteúdo é legítimo. Caso contrário, o conteúdo é identificado como poluído.

3.5 Abordagens para a Escolha da Probabilidade de Verificação de Assinaturas

O `CCNCheck` em si é apenas um mecanismo para a verificação probabilística de assinaturas. Ele não limita quais devem ser os valores de p e nem de que maneira tais valores devem ser escolhidos. Por essa razão, este trabalho propõe e analisa três abordagens diferentes para tal escolha que poderiam ser utilizadas na prática.

3.5.1 Abordagem I: Probabilidades Estáticas e Iguais

Esta abordagem é a mais simples e direta, onde todos os roteadores da rede são configurados com o mesmo valor para a probabilidade de verificação de assinaturas p. Neste caso, a eficiência do `CCNCheck` depende da topologia da rede, pois o valor de p em cada roteador deve ser pequeno. Uma análise da influência da topologia na eficiência do `CCNCheck` é apresentada na Seção 5.

3.5.2 Abordagem II: Probabilidade Estática e Mais Alta na Borda

Para esta abordagem, os roteadores da rede são divididos em dois grupos: roteadores de borda e roteadores do núcleo da rede. Os roteadores de borda são aqueles conectados diretamente a um consumidor ou publicador. Por outro lado, os roteadores do núcleo da rede só são conectados a outros roteadores do mesmo grupo ou a roteadores de borda. Todos os roteadores do núcleo da rede verificam conteúdos de acordo com uma probabilidade p. Já os roteadores de borda verificam conteúdos com probabilidade p_{borda}. Uma vez que os roteadores de borda possuem uma carga de tráfego de conteúdos consideravelmente menor do que a dos roteadores do núcleo da rede, é possível escolher p e p_{borda} de maneira que $p \ll p_{borda}$.

3.5.3 Abordagem III: Probabilidade Dinâmica na Borda

Além de reconhecer que os roteadores de borda e do núcleo da rede podem verificar conteúdos com probabilidades diferentes, é possível também estabelecer uma diferenciação entre os roteadores de borda quanto a sua exposição a conteúdos poluídos. Utilizando a abordagem II, todos os roteadores de borda devem verificar conteúdos com a mesma probabilidade alta p_{borda}. Isso significa que se, por exemplo, um roteador de borda for configurado com $p_{borda} = 0.8$ e estiver conectado a apenas um publicador malicioso e 3 legítimos, então apenas 20% das verificações de assinatura serão úteis, já que 60% delas serão realizadas para conteúdos legítimos.

Assim, para ser mais eficiente, roteadores de borda diferentes podem ser configurados com valores de p_{borda} diferentes, de acordo com sua exposição à poluição. Esses valores diferentes de p_{borda} poderiam ser definidos estaticamente, de acordo com as condição atuais na rede. Entretanto, uma ideia mais interessante é permitir que p_{borda} seja calculado de maneira adaptativa, aumentando ou diminuindo de acordo com o crescimento ou redução do nível de poluição ao qual o roteador está sendo submetido.

Neste sentido, este trabalho propõe um procedimento simples e empírico para calcular dinamicamente a probabilidade de verificação de assinaturas para os roteadores de borda da rede. Inicialmente, todos os roteadores de borda são configurados com $p_{borda} = p_{min}$, onde p_{min} é a menor probabilidade com que o roteador verifica conteúdos. Quando um conteúdo é verificado, o valor de p_{borda} é atualizado segundo a Equação 3.5. Os valores de k_1 e k_2 determinam o quanto o valor de p_{borda} é afetado pela recepção de um conteúdo poluído e de um conteúdo legítimo, respectivamente. Por exemplo, se $k_1 = 0.01$ e $k_2 =$

0.001, então seria necessário receber 10 conteúdos poluídos, sem verificar um conteúdo legítimo para que p_{borda} aumentasse em 10%. Por outro lado, para reduzir p_{borda} em 10% seria necessário receber 100 conteúdos legítimos sem verificar um conteúdo poluído. Uma vez que a recepção de um conteúdo poluído representa um evento mais importante do que a recepção de um conteúdo legítimo, k_2 deve ser razoavelmente menor que k_1.

$$p_{borda} = \begin{cases} \max(1, (p_{borda} + k_1)) & \text{para c poluído} \\ \max(p_{min}, (p_{borda} - k_2)) & \text{para c legítimo} \end{cases} \tag{3.5}$$

3.6 Conclusão do Capítulo

O objetivo deste capítulo foi apresentar o `CCNCheck`, um mecanismo para mitigação de ataques de poluição de conteúdo na CCN. Seu funcionamento é baseado na verificação probabilística de assinaturas pelos roteadores da rede, impedindo assim que uma porção dos conteúdos poluídos se propague na rede. Foram propostas três versões para o `CCNCheck`. O `CCNCheck v.1` verifica a assinatura de conteúdos quando um pacote de dados é recebido pelo roteador. Como resultado, conteúdos legítimos ou não verificados são encaminhados como de costume pela CCN e conteúdos poluídos são simplesmente descartados. Por outro lado, além de implementar as funcionalidades do `CCNCheck v.1`, o `CCNCheck v.2` também verifica conteúdos oriundos do *cache* que seriam encaminhados como resposta a um interesse recebido. Neste caso, se o conteúdo for uma versão poluída, antes de descartá-lo o `CCNCheck v.2` remove a entrada correspondente do `CS`. O `CCNCheck v.3` executa as mesmas funções do `CCNCheck v.2`, mas também realiza uma verificação periódica de todos os conteúdos armazenados em *cache*. Também foi apresentado neste capítulo o algoritmo utilizado para realizar a verificação probabilística de assinaturas utilizado pelas três versões do `CCNCheck`.

Por fim, foram propostas três abordagens para a utilização do `CCNCheck`. Na primeira, todos os roteadores da rede verificam conteúdos de acordo com o mesmo valor de p definido estaticamente. Na segunda abordagem, os roteadores de borda são configurados com um valor de p alto (p_{borda}), enquanto os demais roteadores são configurados com um valor pequeno para p. Por fim, no terceiro cenário o valor de p_{borda} é calculado de maneira adaptativa, de acordo com o nível de poluição ao qual o roteador está associado. Por outro lado, os demais roteadores são configurados um valor pequeno e estático para p. Para o cálculo dinâmico de p_{borda} nos roteadores de borda, foi proposto um procedimento simples baseado na Equação 3.5.

Uma vez definidas as versões do `CCNCheck`, faz-se necessário avaliá-las para cada uma das abordagens de utilização propostas neste capítulo. Tais avaliações são realizadas a partir da execução de simulações, cujo modelo é descrito no próximo capítulo.

Capítulo 4

Modelo de Simulação

Para que seja possível entender os resultados obtidos neste trabalho, é preciso entender de que maneira as simulações realizadas foram modeladas. Para tanto, a Seção 4.1 apresenta uma breve descrição do funcionamento do módulo NDNSim [4], utilizado para prover a implementação da CCN no simulador NS3 [2]. De maneira geral, a dinâmica das simulações pode ser descrita pela topologia utilizada e pelo comportamento dos nós que a compõem. Por essa razão, tais temas são abordados nas Seções 4.2 e 4.3, respectivamente. Por fim, a Seção 4.4 apresenta uma conclusão para este capítulo.

Para avaliar a eficiência das três versões do CCNcheck, foram realizadas simulações utilizando o simulador NS3 [2], através do módulo NDNSim [4], que implementa a pilha de protocolos da CCN. Este capítulo aborda, primeiramente, o funcionamento do módulo NDNSim. Em seguida, é definida a maneira na qual os nós da rede se comportam durante as simulações. Por fim, são apresentadas as topologias utilizadas nos cenários de simulação.

4.1 Funcionamento do NDNSim

O NDNSim não é um simulador, mas sim um módulo do simulador NS3. A Figura 4.1 mostra um esquema de sua arquitetura. Na parte central da figura está o núcleo do NDNSim, onde está implementado o protocolo de camada de rede da CCN (`L3Protocol`). Este componente age como sua espinha dorsal, unindo e controlando todas as demais estruturas. Na parte inferior da mesma figura, podem ser vistas as abstrações do NDNSim para o `CS`, `PIT`, `FIB` e para as estratégias de encaminhamento. As abstrações do `CS` e das estratégias de encaminhamento fornecem uma interface que pode ser utilizada para conectar diferentes implementações do `CS` e das estratégias de encaminhamento ao núcleo

do NDNSim. Já as abstrações para a `PIT` e para a `FIB` fornecem uma implementação padrão contendo alguns parâmetros configuráveis, como por exemplo o tamanho da `PIT` e o algoritmo de busca na `FIB`.

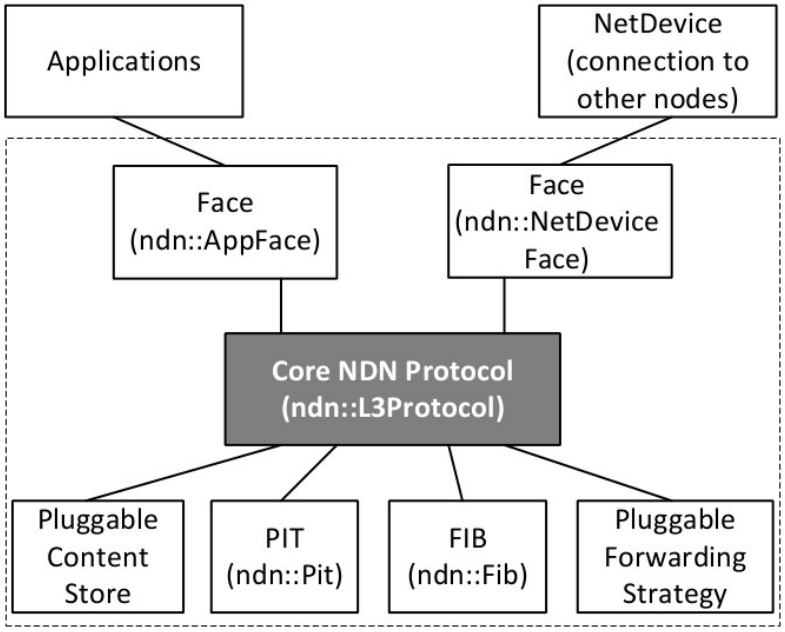

Figura 4.1: Esquema arquitetural do NDNSim [4].

O NDNSim é orientado a eventos. Isso significa que os componentes devem implementar funções pré-definidas contendo a lógica necessária para tratar um evento específico. Por exemplo, componentes da classe de estratégias de encaminhamento, como o `CCNCheck`, possuem duas funções especiais, uma que notifica o componente sobre a chegada de um interesse e outra que o notifica sobre a chegada de um pacote de dados.

Ainda na Figura 4.1, é possível identificar mais dois componentes conectados ao núcleo do NDNSim, as *faces* `AppFace` e `NetDeviceFace`. No contexto do NDNSim, as *faces* são abstrações utilizadas para realizar a comunicação entre a camada de rede da CCN (`L3Protocol`) e as outras camadas da pilha de protocolos. A `AppFace` é utilizada para conectar as aplicações ao `L3Protocol`. Por exemplo, se um nó deseja ser um consumidor, ele deve implementar a lógica para tal na camada de aplicação e utilizar uma `AppFace` para enviar interesses e receber dados utilizando a camada de rede da CCN. Por outro lado, a `NetDeviceFace` é utilizada para conectar o `L3Protocol` a um protocolo de camada de enlace utilizado em uma determinada interface. Dessa forma, os dados enviados por um nó através da CCN são inseridos diretamente em um *frame* e transmitidos de acordo com o protocolo de camada de enlace utilizado.

Analisando este modelo de comunicação, é possível perceber que, utilizando-se *faces*

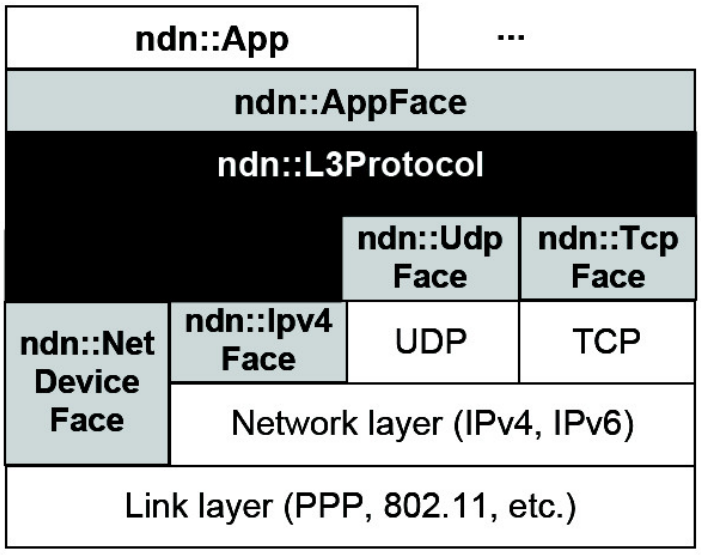

Figura 4.2: Comunicação entre a camada de rede da CCN e as demais camadas da pilha de protocolos [4].

adequadas, a CCN pode ser implementada em cima de qualquer protocolo existente na pilha TCP/IP. A Figura 4.2 ilustra este fato. Além da `AppFace` e da `NetDeviceFace`, o NDNSim atualmente implementa a `TCPFace` e a `UDPFace`. Tais *faces* permitem que a CCN utilize os protocolos TCP e UDP, respectivamente, para enviar interesses e receber dados.

Nas simulações, as aplicações referentes ao consumidor, produtor legítimo e produtor malicioso se comunicam com a camada de rede da CCN através de uma `AppFace`. Por outro lado, cada interface física está associada a uma `NetDeviceFace`, o que permite que pacotes de interesse e pacotes de dados sejam trocados entre dois nós remotos diretamente através do protocolo de camada de enlace.

4.2 Topologias de Rede Utilizadas

Topologia em Grade: A topologia em grade, ilustrada na Figura 4.3, possui 21 linhas e 21 colunas, totalizando 441 nós e 840 ligações entre eles. As linhas e colunas da grade são numeradas de 1 a 21. Assim, o publicador malicioso ocupa a posição $(1, 10)$, o publicador legítimo ocupa a posição $(1, 12)$ e o consumidor ocupa a posição $(21, 11)$. Todos os enlaces operam com banda de $1Mbps$ e atraso de $10ms$. Para introduzir um grau de variabilidade na rede, quando um pacote de dados é recebido por um roteador, ele não é imediatamente processado. Ao invés disso, o processamento do pacote de dados é agendado para ser executado entre 1 e 2

milissegundos no futuro. Para tanto, é sorteado aleatoriamente um valor a partir do intervalo $[1, 2]$, onde a probabilidade de qualquer valor ser sorteado é distribuída uniformemente no intervalo. Consequentemente, em média, o processamento dos pacotes de dados é atrasado por $1,5$ milissegundos.

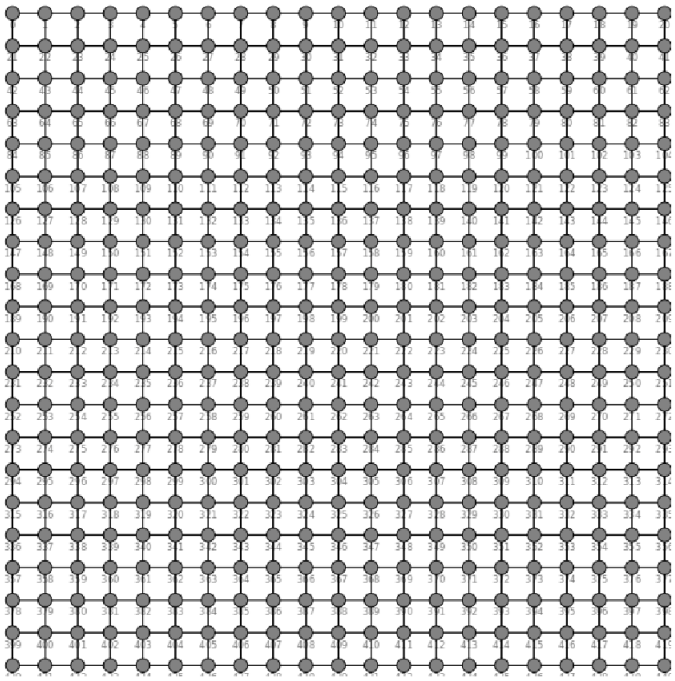

Figura 4.3: Topologia em grade com 21 linhas e 21 colunas.

Topologia Rocketfuel: A topologia *rocketfuel*, ilustrada na Figura 4.4, é baseada na topologia do ISP Exodus, obtida através do mapeador de topologias Rocketfuel [24]. A rede conta com 192 nós, onde 95 são nós folha, 58 são *gateways* [1] e os outros 39 compõem o *backbone* da rede. Esses nós estão conectados entre si através de 422 ligações. Diferentemente da topologia em grade, nas simulações utilizando a topologia *rocketfuel* não foram fixadas posições para o consumidor e para os produtores (legítimo e malicioso). Ao contrário, tais posições foram escolhidas aleatoriamente durante as simulações.

4.3 Comportamento dos Nós da Rede

Os cenários de simulação avaliados neste trabalho podem ser descritos de acordo com o comportamento dos nós que compõem a rede. No contexto da CCN existem três tipos

[1] *Gateways* são nós que estão conectados somente a nós folha e aos roteadores do núcleo da rede.

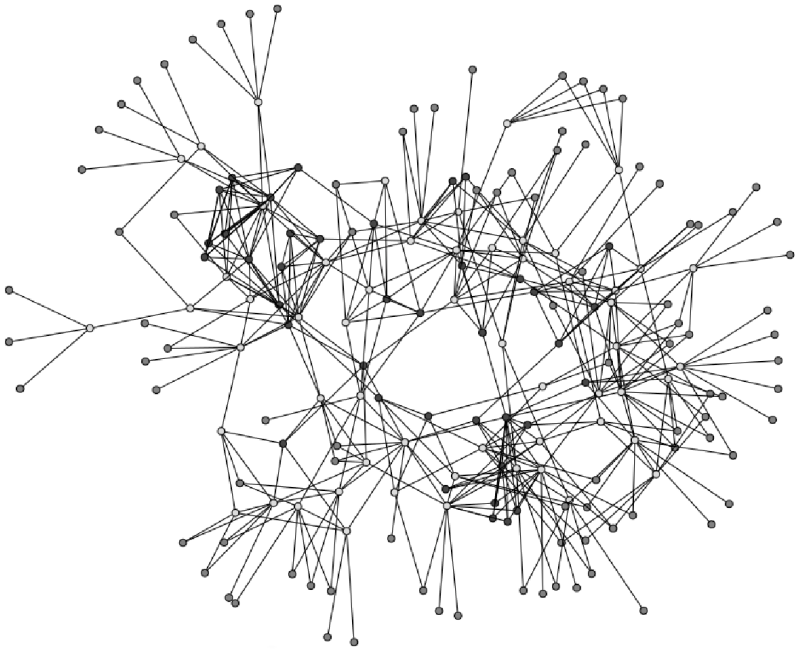

Figura 4.4: Topologia rocketfuel.

diferentes de nós: publicadores de conteúdo, consumidores e roteadores.

4.3.1 Publicadores de Conteúdo

Inicialmente, foi preciso definir de que maneira seria implementada a publicação de conteúdos. A CCN não prevê um padrão para tal e por isso é possível escolher a maneira que melhor se encaixa em cada caso. O NDNSim possui três opções para este propósito:

1. as entradas na `FIB` de cada roteador devem ser inseridas manualmente, sendo equivalente ao uso de rotas estáticas;

2. uma entidade global é responsável por calcular as rotas e popular as `FIBs` de todos os roteadores da rede, atuando como um protocolo de roteamento centralizado;

3. os interesses inundam a rede em busca de um nó que possua o conteúdo requisitado.

Claramente a primeira opção não é nem prática nem escalável, o que limitaria bastante a densidade das topologias utilizadas nas simulações. A segunda opção seria interessante, porém ela é implementada de forma a não permitir que um interesse seja encaminhado por mais de uma interface em um roteador. Dessa maneira, ou o interesse seria sempre encaminhado até um publicador legítimo ou até um malicioso, o que não seria adequado

à análise da poluição de conteúdos na CCN. Dessa forma, optou-se por utilizar a terceira opção, ou seja, a inundação de pacotes de interesse.

No modelo de simulação utilizado, os publicadores legítimos somente respondem a interesses cujo nome do conteúdo requisitado esteja dentro de seu prefixo de atuação. Por exemplo, suponha que o prefixo `/br.uff.ic` tenha sido definido para o publicador legítimo. Consequentemente, ele responderia com um pacote de dados para `/br.uff.ic/videos/1`, mas não para `/br.ufrj.if/videos/1`. Como resposta, os publicadores legítimos geram conteúdos com 1024 bytes de tamanho e com o mesmo nome indicado no pacote de interesse correspondente.

Por outro lado, o objetivo dos publicadores maliciosos é gerar a maior quantidade de conteúdos poluídos possível. Por essa razão, seu prefixo de atuação é definido como `/`, um prefixo presente em todos os nomes de conteúdo da CCN. O efeito desta configuração é fazer com que os publicadores maliciosos gerem pacotes de dados como resposta para qualquer interesse recebido. As respostas geradas pelos publicadores maliciosos também são conteúdos com 1024 bytes de tamanho, mas, para identificar o conteúdo como poluído, o cabeçalho do pacote de dados associado possui uma *flag* de poluição ligada. Dessa maneira, os publicadores maliciosos realizam um ataque de poluição de conteúdos genérico, que pode ser enquadrado em qualquer umas das três categorias de ataque descritas na Seção 2.3.

4.3.2 Consumidor

Em todas as simulações realizadas, o consumidor está sempre interessado em 20 conteúdos diferentes e os requisita com a taxa de `10 interesses/s`. Para que o consumidor não aguarde indefinidamente por uma resposta, para cada interesse enviado é mantido um temporizador. Inicialmente, o valor do temporizador é definido como $50ms$. Conforme os interesses vão sendo atendidos, o consumidor calcula um valor estimado para o RTT e o utiliza como parâmetro para definir o novo valor para o temporizador. É importante ressaltar que, para o consumidor, interesses somente são considerados atendidos caso o conteúdo recebido seja legítimo. Caso contrário, o conteúdo é simplesmente descartado e não é considerado no cálculo da estimativa do RTT. Quando o temporizador se esgota, o consumidor pode requisitar novamente o mesmo conteúdo ou desistir. Para as simulações, o consumidor foi configurado para realizar 10 tentativas de recuperar um conteúdo antes de desistir.

Ao receber um conteúdo, o consumidor verifica sua assinatura e o descarta caso este

seja uma versão poluída. O descarte do conteúdo poluído é realizado antes que qualquer estrutura de dados do nó seja alterada. Assim, do ponto de vista do consumidor, a recepção de um conteúdo poluído é equivalente a não receber conteúdo algum e o consumidor continua esperando receber uma versão legítima até o esgotamento do temporizador.

4.3.3 Roteadores

Nas simulações, todos os roteadores possuem o módulo do `CCNCheck` instalado. Entretanto, seu comportamento é dependente da versão do `CCNCheck` utilizada. Caso sejam configurados com o `CCNCheck v.1`, os roteadores verificam com probabilidade p a assinatura dos conteúdos que recebem, descartando as versões poluídas. Por outro lado, se os roteadores forem configurados com o `CCNCheck v.2`, então os conteúdos oriundos do *cache* também são verificados com a mesma probabilidade p antes de serem encaminhados. Por fim, se o `CCNCheck v.3` for utilizado, além das funcionalidades executados pelo `CCNCheck v.2`, também é realizada, de um em um segundo, uma verificação de todos os conteúdos armazenados no `CS` do roteador.

4.4 Conclusão do Capítulo

Este capítulo apresentou uma descrição do modelo de simulação utilizado para avaliar o `CCNCheck`. Inicialmente foi abordado, resumidamente, o funcionamento do módulo NDNSim, utilizado para implementar a pilha da CCN no simulador NS3. Em seguida, foram apresentadas as topologias de rede utilizadas durante as simulações. A topologia em grade é sintética, possui caminhos entre o consumidor e os publicadores com um grande número de saltos (≥ 21) e tem uma estrutura simétrica. Por outro lado, a topologia *rocketfuel*, gerada a partir da topologia de um ISP real, possui caminhos com um número de saltos reduzido e uma estrutura assimétrica. Por fim, foi apresentado o comportamento dos publicadores, do consumidor e dos roteadores que compõem as topologias utilizadas durante as simulações.

Apesar do valor da probabilidade de verificação de conteúdos (p) ter que ser pequeno para manter o *overhead* de verificação de assinaturas baixo, a probabilidade do consumidor receber um conteúdo poluído depende fortemente da topologia utilizada. Por essa razão, o próximo capítulo apresenta uma análise matemática da influência da topologia na eficiência do `CCNCheck`.

Capítulo 5

Análise do Impacto da Topologia no CCNcheck

Para evitar que o *overhead* gerado pela verificação de assinaturas seja excessivo para os roteadores, a probabilidade de verificação empregada deve assumir valores pequenos. Entretanto, a probabilidade do consumidor receber conteúdos poluídos depende da topologia utilizada. Neste capítulo, tal dependência será analisada sob a perspectiva do número de saltos entre o consumidor e o publicador malicioso, e do número de ramificações conectando o consumidor ao resto da rede.

No restante deste capítulo, considere que o consumidor solicita o conteúdo desejado somente uma vez. Se um conteúdo poluído for obtido como resposta, o consumidor o descarta e desiste de obter tal conteúdo.

5.1 Influência do Número de Saltos

Suponha que o consumidor esteja conectado a um publicador malicioso através de apenas um roteador. Essa configuração é mostrada na Figura 5.1. Nesse caso, apenas o roteador `R` poderá verificar a assinatura dos conteúdos, evento que ocorre com probabilidade p. Assim, o consumidor só receberá um conteúdo poluído no caso de `R` não verificar sua assinatura. Logo, a probabilidade do consumidor receber um conteúdo poluído ($P_r[CO_P]$) será igual a $(1-p)$.

Como o valor de p precisa ser pequeno, $(1-p)$ será um valor grande e o consumidor receberá muitos conteúdos poluídos, mesmo com a utilização do CCNcheck. Suponha agora que o roteador da Figura 5.1 foi substituído por dois roteadores conectados em série. Consequentemente, o consumidor só receberá um conteúdo poluído se ambos os roteadores

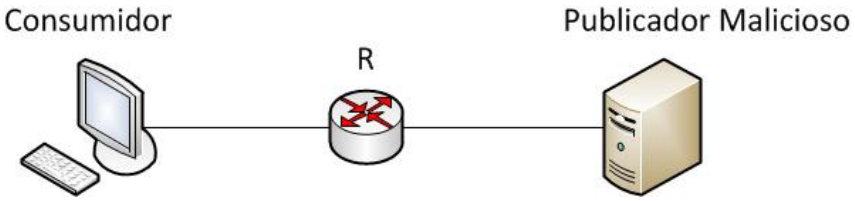

Figura 5.1: Topologia com apenas um roteador ligando o consumidor e o publicador malicioso.

não verificarem sua assinatura. Neste caso, o consumidor receberá um conteúdo poluído com probabilidade $P_r[CO_P] = (1-p)^2$. Logo, a probabilidade do consumidor receber conteúdos poluídos reduziu com a adição de mais um roteador conectando-o ao publicador malicioso.

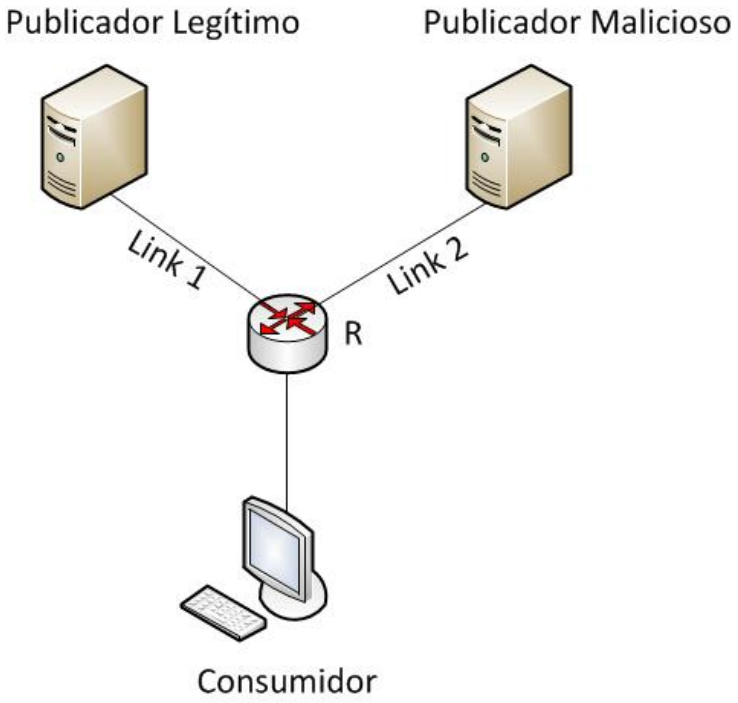

Figura 5.2: Topologia com apenas um roteador ligando o consumidor aos publicadores.

Considere agora que além do publicador malicioso, também exista um publicador legítimo. A Figura 5.2 ilustra este cenário. Neste caso, ao receber um interesse, o roteador R o encaminha para ambos os publicadores conectados a ele. Como consequência, para cada interesse do consumidor, R recebe dois pacotes de dados como resposta, um legítimo e o outro poluído. Como somente o primeiro destes pacotes de dados irá consumir a entrada na PIT gerada pelo interesse, somente tal pacote de dados será efetivamente recebido por R. Suponha que os atrasos $d1$ e $d2$ nos *links* 1 e 2, respectivamente, sejam valores aleatórios oriundos de uma mesma distribuição de probabilidades. Nesse caso, para cada sorteio de $d1$ e $d2$, três situações podem ocorrer:

1. $d1 < d2$: o roteador R recebe o conteúdo legítimo e descarta o poluído;
2. $d1 > d2$: o roteador R recebe o conteúdo poluído e descarta o legítimo;
3. $d1 = d2$: ambos os conteúdos chegam ao mesmo tempo em R e este usa algum critério determinístico para escolher qual conteúdo será aceito e qual será descartado.

Se a probabilidade de ambos os atrasos sorteados serem iguais for muito pequena, a situação 3 pode ser desprezada. Assim, somente as situações 1 e 2 podem ocorrer. Como consequência, a probabilidade do roteador R efetivamente receber um conteúdo poluído ($P_r[R_P]$) é igual a probabilidade dele receber um conteúdo legítimo, que por sua vez é igual a $\frac{1}{2}$. No caso do roteador R verificar assinaturas com probabilidade $p = 0$, então 50% dos interesses enviados pelo consumidor serão atendidos por conteúdos poluídos. Por outro lado, se R verificar assinaturas com probabilidade $0 < p \leq 1$, então o consumidor só receberá um conteúdo poluído caso este seja recebido e não verificado pelo roteador R. Logo, a probabilidade do consumidor receber um conteúdo poluído é dada pela Equação 5.1.

$$P_r[CO_P] = \frac{1}{2} \cdot (1 - p) \tag{5.1}$$

Considere agora a Figura 5.3. Na topologia ilustrada, existem $n + 1$ saltos entre o consumidor e qualquer um dos dois publicadores. Seguindo a mesma linha de raciocínio das figuras anteriores, o consumidor só receberá um conteúdo poluído caso $R1$ receba efetivamente um conteúdo poluído e nenhum dos n roteadores verifique sua assinatura. Se todos os roteadores forem configurados com o mesmo valor de p para a verificação de assinaturas, então a probabilidade do consumidor receber um conteúdo poluído é dada pela Equação 5.2. Consequentemente, pode-se concluir que $P_r[CO_P]$ reduz exponencialmente em função do número de saltos entre o consumidor e o produtor malicioso.

$$P_r[CO_P] = P_r[R1_P] \cdot (1 - p)^n \tag{5.2}$$

Para validar a Equação 5.2, foi realizado um conjunto de simulações utilizando-se a topologia da Figura 5.3 variando-se o número de saltos de 2 a 20. Todos os roteadores foram configurados com o `CCNCheck V2` com probabilidade de verificação igual a 10%. É importante notar que, como existe apenas um consumidor e cada conteúdo é requisitado apenas uma vez, não faz diferença usar o `CCNCheck v.1` ou o `CCNCheck v.2`, pois nenhum conteúdo é recuperado do *cache*.

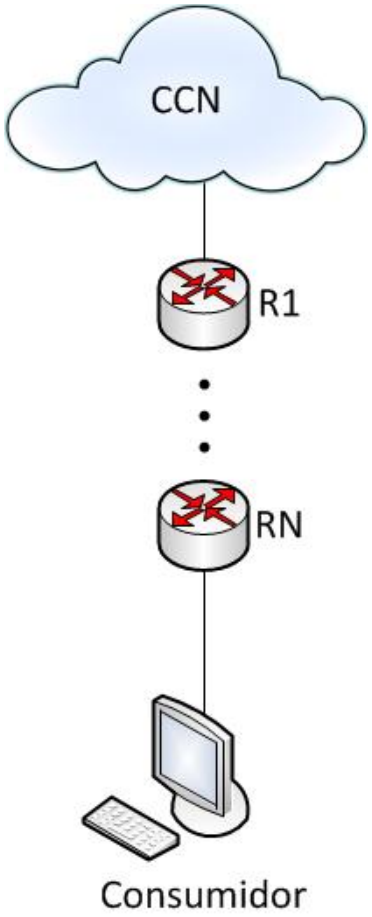

Figura 5.3: Topologia com n roteadores ligando o consumidor aos publicadores.

O roteador `R1` foi conectado a um publicador malicioso e a um publicador legítimo, assim como na Figura 5.2. Os atrasos dos *links* ligando o roteador `R1` aos publicadores foram escolhidos a cada transmissão de pacote de dados, de acordo com a distribuição uniforme. Logo, a probabilidade do roteador `R1` receber um conteúdo poluído ($P_r[R1_P]$) é igual a $\frac{1}{2}$. Para cada valor de n, foram realizadas 500 rodadas de simulação. Em cada rodada, a probabilidade do consumidor receber um conteúdo poluído ($P_r[CO_P]$) foi calculada dividindo-se o total de conteúdos poluídos recuperados pelo total de conteúdos requisitados. Este cálculo merece uma explicação mais detalhada.

Ao enviar um interesse, o consumidor pode obter como resposta um conteúdo legítimo ou um conteúdo malicioso, ou ainda, simplesmente não receber resposta alguma. Devido ao cenário de simulação utilizado, na ausência de publicadores maliciosos, os consumidores sempre conseguem receber todos os conteúdos requisitados sem a necessidade de retransmissões. Portanto, se o consumidor não recebe conteúdo algum como resposta para determinado interesse, isso significa que, no caminho até o consumidor algum roteador verificou a assinatura do conteúdo, detectou a poluição e descartou o pacote de

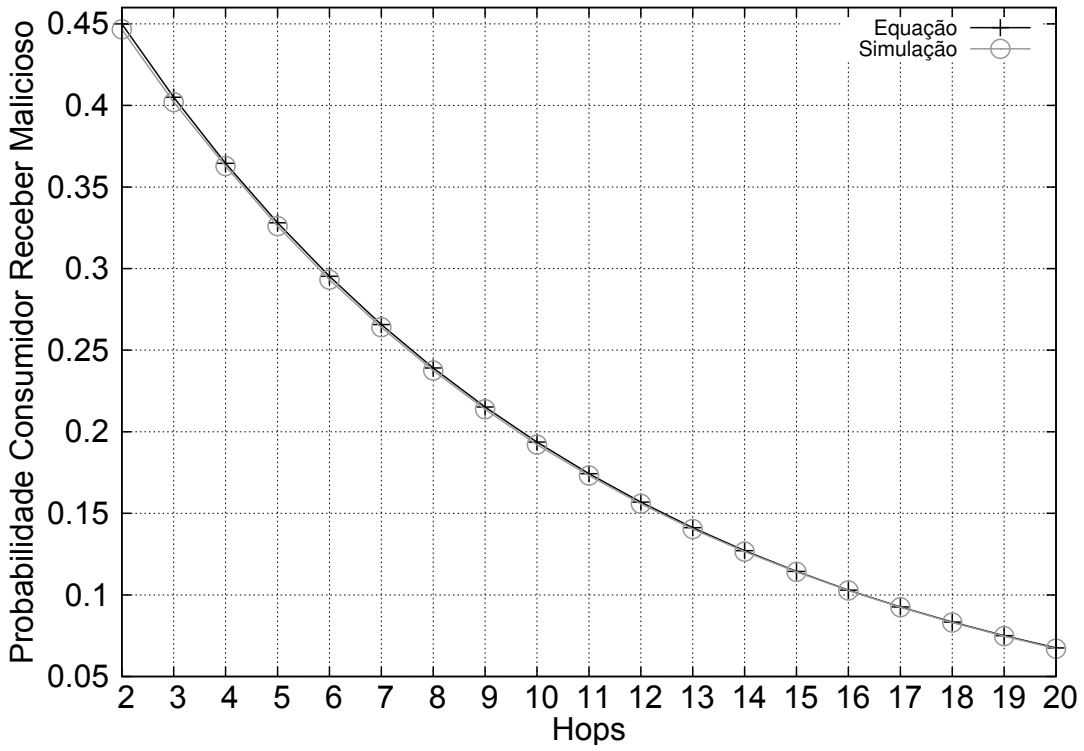

Figura 5.4: Probabilidade do consumidor receber um conteúdo poluído variando-se o número de saltos de 2 à n.

dados. Portanto, a ausência de resposta para um interesse pode ser interpretada como um terceiro tipo de conteúdo, além dos legítimos e poluídos. Assim, a probabilidade do consumidor receber um conteúdo poluído é dada pela frequência relativa dos conteúdos poluídos recebidos, que pode ser calculada como $P_r[CO_P] = \frac{P}{(P+L+A)}$, onde P é o total de conteúdos poluídos recebidos, L o total de conteúdos legítimos recebidos e A o total de ausências de resposta. Como todos os interesses enviados pelo consumidor retornam um conteúdo legítimo, poluído ou a ausência de conteúdo, então $(P + L + A)$ é igual ao total de conteúdos requisitados. Consequentemente, $P_r[CO_P]$ pode ser calculada como a razão do total de conteúdos poluídos recebidos pelo total de conteúdos requisitados.

Por fim, foram calculados a média e o desvio padrão de $P_r[CO_P]$ nas 500 rodadas de simulação para cada valor de n. A Figura 5.4 mostra um gráfico que compara os valores para $P_r[CO_P]$ obtidos a partir da Equação 5.2 com aqueles obtidos a partir das simulações. Pode-se perceber que as curvas praticamente se sobrepõem, o que corrobora com a validade da Equação 5.2. Na realidade, os valores gerados pela simulação possuem uma pequena variação, mas devido à escala do gráfico, se plotada, a barra de erro gerada seria quase que invisível. Assim, os valores usados para plotar o gráfico, incluindo o desvio padrão, são mostrados na Tabela 5.1.

Tabela 5.1: Média e desvio padrão de $P_r[CO_P]$ na topologia da Figura 5.4, calculados pela Equação 5.2 e através de simulações.

# de Saltos	Equação 5.2	Simulação	
	Média	Média	Desvio Padrão
2	0,45	0,44661	0,00066
3	0,405	0,40204	0,00066
4	0,3645	0,36282	0,00069
5	0,32805	0,32604	0,00064
6	0,29524	0,29328	0,00063
7	0,26572	0,26403	0,00061
8	0,23914	0,23754	0,00059
9	0,21523	0,21378	0,00056
10	0,19371	0,19206	0,00054
11	0,17433	0,17317	0,00052
12	0,15690	0,15590	0,00048
13	0,14121	0,14035	0,00046
14	0,12709	0,12655	0,00044
15	0,11438	0,11413	0,00042
16	0,10294	0,10279	0,00039
17	0,09265	0,09247	0,00039
18	0,08338	0,08310	0,00037
19	0,07504	0,07478	0,00034
20	0,06754	0,06708	0,00031

5.2 Influência do Número de Ramos

Além do número de saltos, o número de ramos conectados a um nó também influencia na sua probabilidade de receber conteúdos poluídos. Na topologia da Figura 5.2, o consumidor está conectado aos publicadores por apenas um ramo. Agora, considere o efeito de se adicionar mais um ramo ao consumidor, como mostrado na Figura 5.5. Da mesma forma que no roteador `R` da Figura 5.5, nos roteadores `R1` e `R2` metade dos conteúdos recebidos serão legítimos e a outra metade serão poluídos. O consumidor também receberá metade dos conteúdos encaminhados por $R1$ e a outra metade por `R2`.

Os dois ramos ligados ao consumidor são simétricos, ou seja, ambos possuem apenas um caminho com dois saltos entre o consumidor e os publicadores. Além disso, as probabilidades de `R1` e `R2` receberem conteúdos poluídos são iguais e ambos os roteadores verificam assinaturas com a mesma probabilidade p. Por essa razão, a probabilidade de um conteúdo poluído ser transmitido pelo *link 1* é igual a probabilidade de um conteúdo poluído ser transmitido pelo *link 2*, que por sua vez é dada por $\frac{1}{2} \cdot (1 - p)$. Como o consumidor só recebe metade dos conteúdos enviados por cada *link* e ele pode receber um

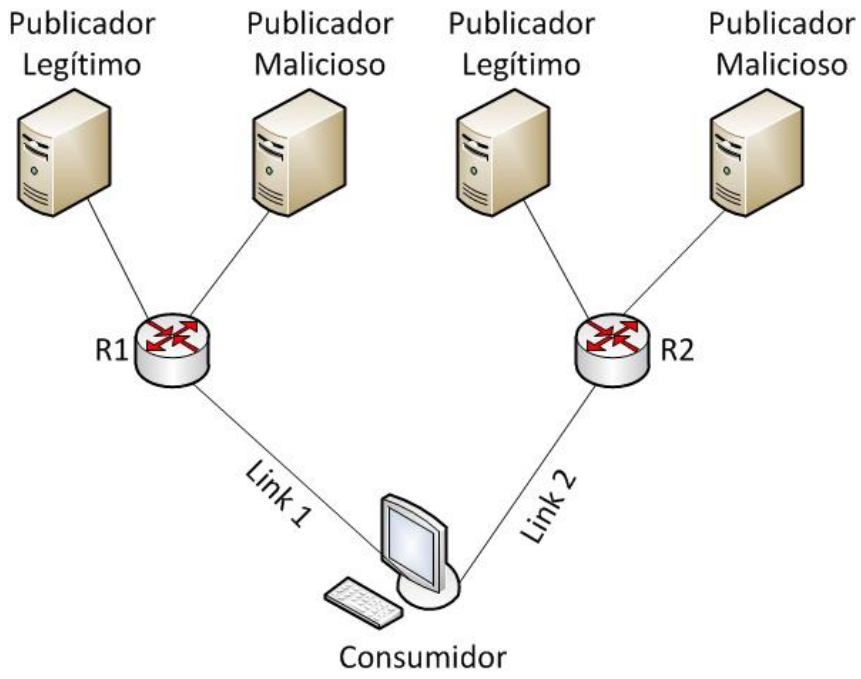

Figura 5.5: Topologia com dois ramos simétricos ligando o consumidor aos publicadores.

conteúdo poluído pelo *link 1* ou pelo *link 2*, sua probabilidade de receber um conteúdo poluído é dada pela Equação 5.3. Pode-se perceber então que quando há simetria, a adição de novos ramos não influencia o valor de $P_r[CO_P]$.

$$P_r[CO_P] = \frac{1}{4} \cdot (1-p) + \frac{1}{4} \cdot (1-p) = \frac{1}{2} \cdot (1-p) \tag{5.3}$$

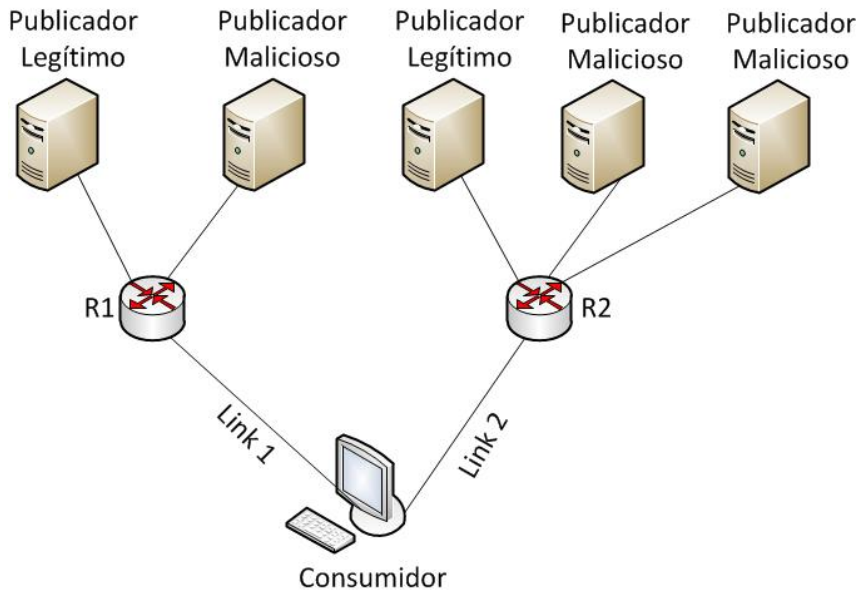

Figura 5.6: Topologia com dois ramos assimétricos ligando o consumidor aos publicadores.

Considere agora que os ramos da Figura 5.5 sejam assimétricos. Um exemplo é mos-

trado na Figura 5.6. Neste caso, a probabilidade do roteador $R2$ receber um conteúdo poluído passa a ser $\frac{2}{3}$. Assim, o valor de $P_r[CO_P]$ passa a ser dado pela Equação 5.4.

$$P_r[CO_P] = \frac{1}{4} \cdot (1-p) + \frac{1}{3} \cdot (1-p) = \frac{7}{12} \cdot (1-p) \tag{5.4}$$

Neste caso, o valor de $P_r[CO_P]$ aumentou em relação a topologia de ramo único da Figura 5.2, pois a quantidade de conteúdos poluídos que atravessam o novo ramo é maior do que no ramo original. Se os dois publicadores maliciosos conectados ao roteador $R2$ na Figura 5.6 fossem removidos, então a probabilidade do roteador $R2$ receber um conteúdo poluído seria igual a 0. Consequentemente, o valor de $P_r[CO_P]$ passaria a ser obtido a partir da Equação 5.5. Neste caso, ao adicionar um novo ramo, o valor de $P_r[CO_P]$ cai pela metade.

$$P_r[CO_P] = \frac{1}{4} \cdot (1-p) + 0 \cdot (1-p) = \frac{1}{4} \cdot (1-p) \tag{5.5}$$

Pode-se generalizar a topologia da Figura 5.5 para a mostrada na Figura 5.7. Agora, n ramos ligam o consumidor ao resto da rede. A probabilidade dos roteadores `Ri` receberem conteúdos poluídos depende do resto da topologia ligada a eles. Neste cenário, considerando que todos os roteadores tem a mesma probabilidade de verificar assinaturas, então a probabilidade do consumidor receber um conteúdo poluído pode ser dada pela Equação 5.6.

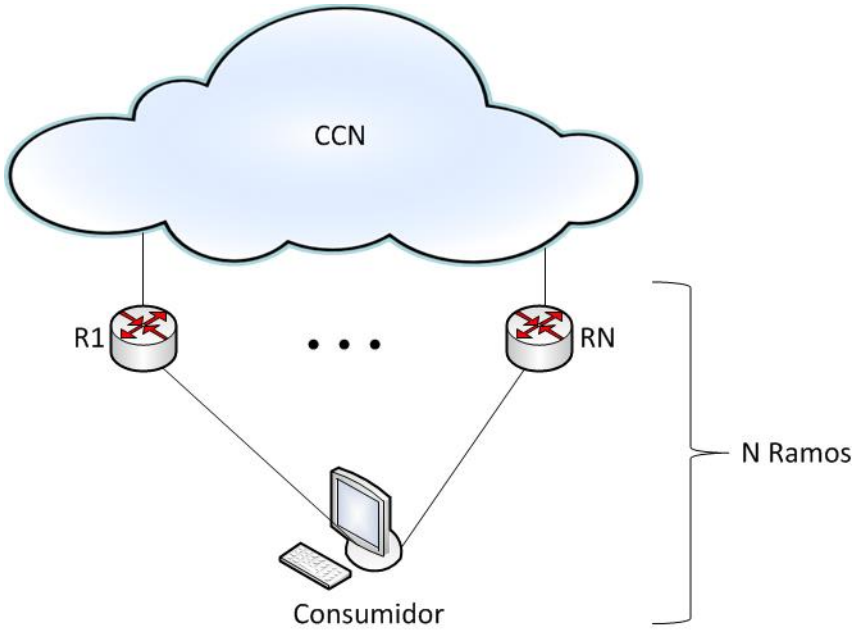

Figura 5.7: Topologia com dois ramos simétricos ligando o consumidor aos publicadores.

$$P_r[CO] = \sum_{i=1}^{n} P_r[V_i] \cdot (1-p) \cdot \frac{1}{n} \tag{5.6}$$

Para validar tal equação foi realizado um conjunto de simulações variando-se o número de ramos de 1 a n, como mostrado na Figura 5.7. Cada roteador `Ri` foi conectado a um publicador legítimo e um malicioso. Para que as topologias fossem criadas automaticamente, mas sem gerar somente ramos simétricos, os publicadores maliciosos foram configurados para responder a interesses de acordo com uma probabilidade escolhida aleatoriamente. Além disso, quando o publicador malicioso envia um conteúdo juntamente com o publicador legítimo do mesmo ramo, o roteador `Ri` conectado a eles recebe, impreterivelmente, o conteúdo malicioso. Por essa razão, é possível dizer que a probabilidade do roteador `Ri` receber um conteúdo poluído é igual a probabilidade do publicador malicioso conectado a ele enviar tal conteúdo. Foram executadas 500 rodadas de simulação para cada número de ramos avaliado n.

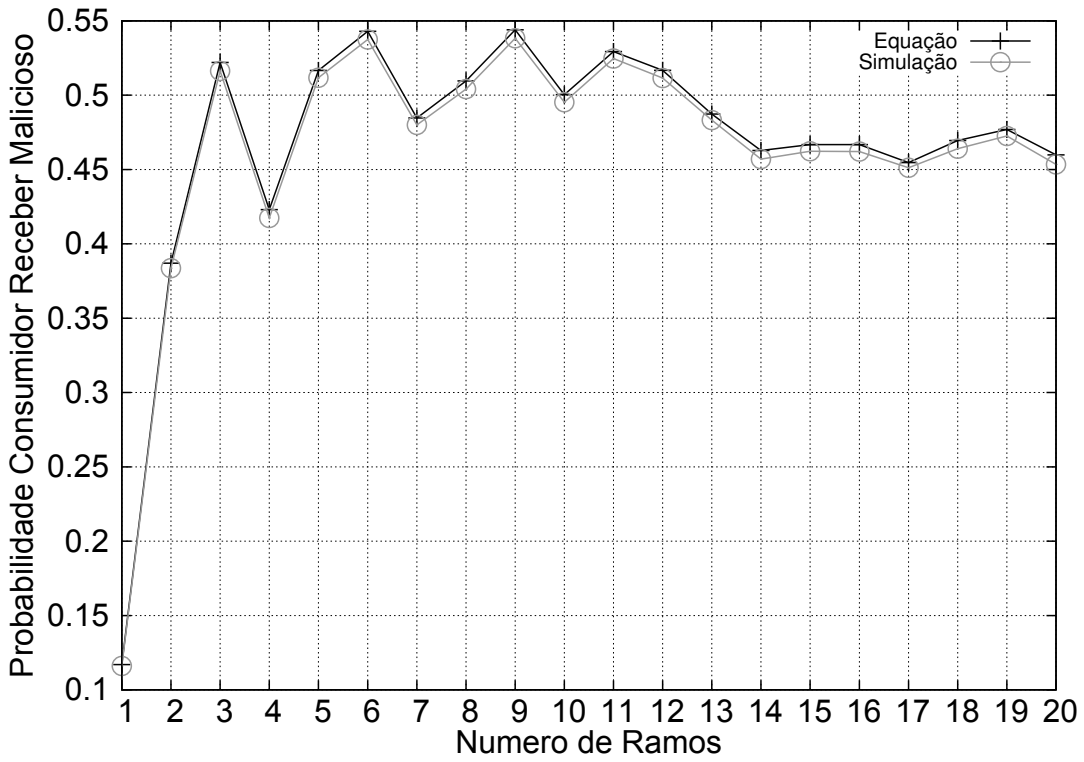

Figura 5.8: Probabilidade do consumidor receber um conteúdo poluído variando-se o número de ramos de 1 a n.

Em cada rodada, foi calculada a probabilidade do consumidor receber um conteúdo poluído. Para tanto, foi utilizada a mesma abordagem que no caso da variação do número de saltos. Por fim, foram calculados a média e o desvio padrão de $P_r[CO_P]$ para cada valor de n. Os resultados foram plotados no gráfico da Figura 5.8. A Tabela 5.2 mostra

a probabilidade dos roteadores `R1` a `R20` receberem um conteúdo poluído. A Tabela 5.3 mostra a média e desvio padrão da probabilidade do consumidor receber conteúdo poluído para o número de ramos variando de 1 a 20. É importante ressaltar que fossem outros os valores da Tabela 5.2, o comportamento do gráfico na Figura 5.8 também seria diferente.

Tabela 5.2: Probabilidade dos roteadores `Ri`.

Roteador	Probabilidade
V1	0,13
V2	0,73
V3	0,88
V4	0,14
V5	0,99
V6	0,75
V7	0,15
V8	0,76
V9	0,91
V10	0,12
V11	0,91
V12	0,42
V13	0,15
V14	0,16
V15	0,58
V16	0,52
V17	0,29
V18	0,8
V19	0,68
V20	0,15

5.3 Conclusão do Capítulo

O objetivo deste capítulo foi analisar a influência das características da topologia na eficiência do `CCNCheck`. Foi possível concluir que, desconsiderando-se os efeitos causados pelos *caches* da rede, a probabilidade do consumidor receber conteúdos poluídos ($P_r[CO_P]$) decresce exponencialmente com o número de saltos existentes no caminho percorrido por um pacote de dados do publicador malicioso até o consumidor. Por outro lado, foi mostrado que não é possível afirmar se a probabilidade do consumidor receber um conteúdo poluído aumenta ou diminui com o número de ramos conectando o consumidor ao resto da rede. Se no novo ramo adicionado ao consumidor o número de conteúdos poluídos propagados for maior do que no ramo original, então $P_r[CO_P]$ irá aumentar. Por outro lado, se o número de conteúdos poluídos propagados no novo ramo for menor do

Tabela 5.3: Média e desvio padrão de $P_r[CO_P]$ na topologia da Figura 5.7, calculados pela Equação 5.6 e através de simulações.

# de Ramos	Equação 5.6	Simulação	
	Média	Média	Desvio Padrão
1	0,11700	0,11605	0,00042
2	0,38700	0,38362	0,00070
3	0,52200	0,51635	0,00067
4	0,42300	0,41744	0,00067
5	0,51660	0,51179	0,00069
6	0,54300	0,53745	0,00066
7	0,48468	0,48005	0,00070
8	0,23914	0,50415	0,00067
9	0,54400	0,53827	0,00066
10	0,50040	0,49527	0,00070
11	0,52932	0,52466	0,00067
12	0,51675	0,51157	0,00067
13	0,48733	0,48322	0,00069
14	0,46281	0,45704	0,00070
15	0,46680	0,46229	0,00069
16	0,46681	0,46207	0,00070
17	0,45469	0,45119	0,00070
18	0,46950	0,46401	0,00069
19	0,47691	0,47261	0,00069
20	0,45990	0,45340	0,00067

que no ramo original, então $P_r[CO_P]$ irá diminuir. Por fim, se o número de conteúdos poluídos propagados no novo ramo for igual ao do ramo original, então $P_r[CO_P]$ não será alterada.

Uma vez apresentada a descrição do modelo utilizado nas simulações (Capítulo 4) e a avaliação da influência da topologia na eficiência do `CCNCheck`, neste capítulo, o próximo passo é avaliar, através de simulações, as abordagens de utilização do `CCNCheck` propostas no Capítulo 3.

Capítulo 6

Avaliação dos Resultados de Simulação

Neste capítulo são avaliadas, através de simulações, as abordagens de utilização do `CCNCheck` propostas no Capítulo 3. Todas as três versões do mecanismo são consideradas, identificando-se os cenários onde elas são mais, ou menos, indicadas. Para a melhor compreensão das análises, primeiramente são discutidas a metodologia e as métricas utilizadas.

6.1 Métricas Avaliadas

A eficiência do `CCNCheck` é interpretada de maneira diferente pelo consumidor e pela rede. Para o consumidor, o que importa é que o conteúdo requisitado seja recuperado. Se uma versão poluída for obtida, o consumidor a descarta e tenta novamente obter o conteúdo. Somente quando uma versão legítima é recebida o conteúdo é considerado como recuperado. Se o consumidor tiver que tentar obter o mesmo conteúdo por diversas vezes até que finalmente logre êxito, não importa quantas versões poluídas serão obtidas neste processo (elas são simplesmente descartadas), mas sim que uma versão legítima seja recuperada. Portanto, para o consumidor, o principal indicador de desempenho do `CCNCheck` é a fração de conteúdos requisitados que foram efetivamente recuperados.

Por outro lado, a rede é afetada pela poluição de conteúdos por ter seus recursos desperdiçados durante o encaminhamento de dados poluídos. Ao verificar probabilisticamente os conteúdos, parte desse desperdício pode ser evitada, pois os conteúdos poluídos que são verificados não são encaminhados. Consequentemente, do ponto de vista da rede, a eficiência do `CCNCheck` é medida pela sua capacidade de reduzir o total de pacotes com dados poluídos trocados.

Em suma, as métricas utilizadas nas avaliações deste capítulo são as seguintes:

Fração de Conteúdos Recuperados (FCR): representa fração dos conteúdos requisitados que foram efetivamente recebidos pelo consumidor. Para o seu cálculo, divide-se o total de conteúdos legítimos recebidos pelo total de conteúdos requisitados pelo consumidor.

Total de Pacotes com dados Poluídos (TPP): mostra quantos pacotes com dados poluídos foram trocados na rede.

6.2 Metodologia

Foram utilizadas metodologias diferentes para as topologias em grade (Figura 4.3) e *rocketfuel* (Figura 4.4). Na topologia em grade, para cada valor da probabilidade de verificação de assinaturas (p) avaliado, foram realizadas 500 rodadas de simulação. Em cada rodada, foram calculadas as métricas FCR e TPP. Por fim, foi calculada a média e o desvio padrão de cada métrica considerando as 500 rodadas de simulação realizadas.

Na topologia *rocketfuel*, foram escolhidos aleatoriamente 50 conjuntos contendo 3 nós folha da rede, um representando o consumidor, outro o publicador legítimo e o último o publicador malicioso. Para cada um dos 50 conjuntos, foram realizadas 500 rodadas de simulação. Da mesma forma que na topologia em grade, em cada rodada de simulação foram calculados os valores para todas as métricas. Por fim, foram calculados a média e o desvio padrão para todas as métricas, considerando todas as rodadas de simulação executadas, ou seja, 500 rodadas para cada um dos 50 conjuntos de posições, totalizando 25000 rodadas de simulação. Esse processo foi repetido para cada valor de p avaliado.

6.3 Cenário 1: Probabilidades Estáticas e Iguais na Topologia em Grade

Neste cenário, todos os roteadores da rede foram configurados com o mesmo valor para a probabilidade de verificação de assinaturas (p). Para reduzir o *overhead* gerado, somente foram considerados valores de p menores que 10%.

6.3.1 Resultados Para a Métrica FCR

O gráfico ilustrado na Figura 6.1 mostra como a fração de conteúdos recuperados varia para diferentes valores de p. O primeiro resultado importante que pode ser extraído é que, para $p = 0$, o `CCNCheck v.3` permitiu que o consumidor recuperasse 100% dos conteúdos requisitados. Comparando este resultado com o comportamento padrão da CCN[1], o `CCNCheck v.3` permitiu um aumento de 48% na métrica `FCR`. Como o `CCNCheck v.3` utiliza um processador adicional para realizar a verificação periódica do *cache*, este procedimento não interfere no processamento dos pacotes trocados na rede[2]. Por essa razão, é possível dizer que o `CCNCheck v.3` maximiza a métrica `FCR`, sem introduzir *overhead* ao processamento dos pacotes recebidos e encaminhados pelo roteador. Por outro lado, com as versões 1 e 2 do `CCNCheck`, se todos os roteadores verificarem 10% do tráfego de dados ($p = 0,1$), serão recuperados aproximadamente 36% mais conteúdos do que na CCN com seu comportamento padrão.

O motivo da eficiência expressiva do `CCNCheck v.3` é a verificação periódica dos *caches* da rede. Quando um consumidor requisita um conteúdo, o interesse inunda a rede na busca pelo mesmo. Se essa requisição representa a primeira tentativa do usuário, então, como só existe um consumidor, obrigatoriamente o interesse será encaminhado até ambos os publicadores, legítimo e malicioso. Consequentemente, ambos os publicadores responderão ao interesse enviando um conteúdo através das interfaces por onde o interesse foi recebido. Se a versão recebida pelo consumidor será legítima ou poluída, depende de características da rede, como atraso nos enlaces, distância entre o consumidor e os publicadores, etc.

Como pode ser observado na Figura 6.1, quando a verificação de conteúdos não é realizada ($p = 0$) e, portanto, a CCN apresenta seu comportamento padrão, aproximadamente 52% dos conteúdos requisitados são recuperados. Os outros 48% representam a fração de conteúdos requisitados que retornaram versões poluídas para o consumidor. Para cada um desses conteúdos poluídos, o consumidor realiza 10 tentativas de recuperar uma versão legítima. Porém, como $p = 0$, uma vez que um conteúdo poluído tenha sido armazenado em *cache*, ele permanece lá indefinidamente. Consequentemente, cada novo interesse enviado, representando uma nova tentativa de recuperar uma versão legítima

[1] Quando as versões 1 e 2 do `CCNCheck` são utilizados com $p = 0$, nenhuma verificação de conteúdos é realizada e, portanto, esta configuração pode ser considerada análoga ao comportamento padrão da CCN.

[2] Na realidade podem haver interferências devido ao compartilhamento de barramentos e outras estruturas do roteador. Entretanto, tais interferências dependem da maneira como o *hardware* foi configurado e, para os fins deste trabalho, podem ser ignoradas.

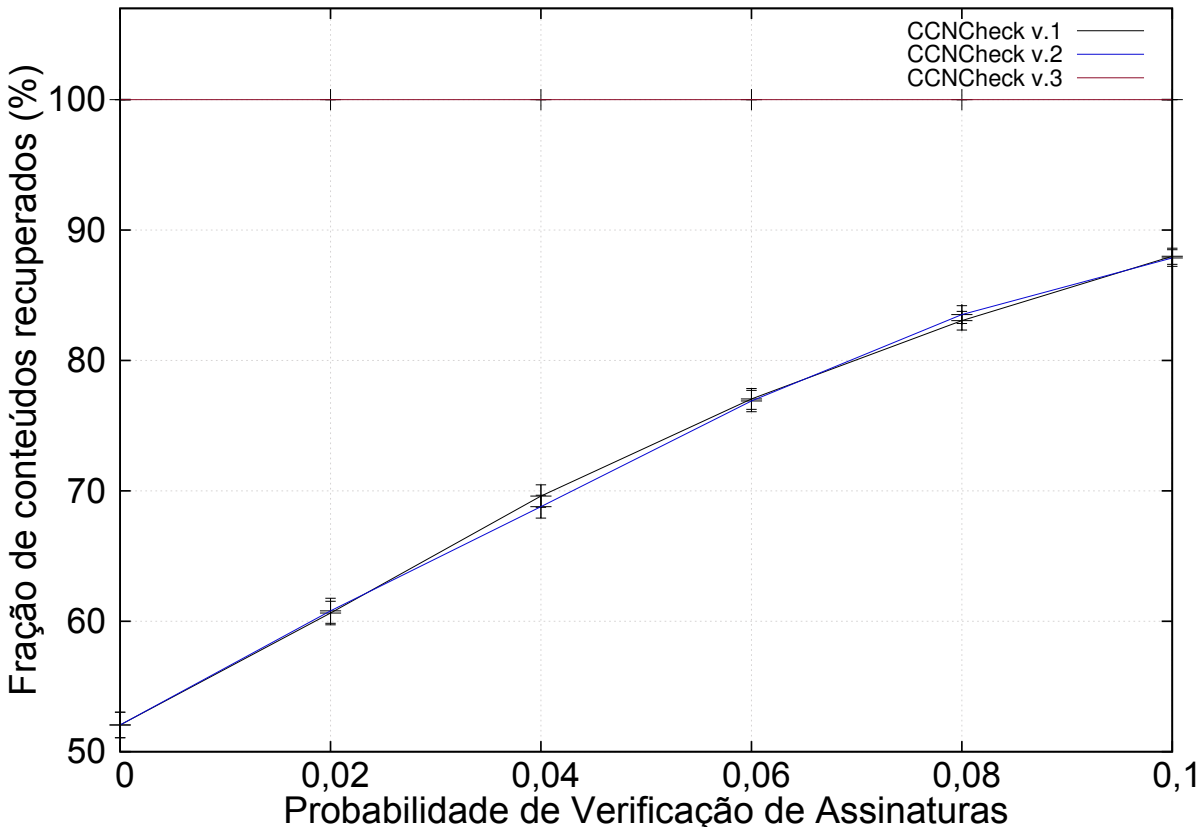

Figura 6.1: Fração de conteúdos recuperados no Cenário 1 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

do conteúdo, é atendida com uma versão poluída diretamente pelo *cache* do roteador de borda do consumidor. Portanto, é possível concluir que a probabilidade do consumidor recuperar um conteúdo legítimo na primeira tentativa é 52%. Já a probabilidade do consumidor recuperar um conteúdo legítimo após a primeira tentativa é igual a 0.

Com a utilização do `CCNCheck v.3`, as entradas poluídas dos *caches* da rede são removidas. Com isso, cria-se uma oportunidade para que, mesmo após receber um conteúdo poluído, o consumidor seja capaz de obter uma versão legítima em uma nova tentativa. Agora, conteúdos poluídos só podem ser obtidos diretamente do publicador malicioso e não mais dos *caches* da rede. Por outro lado, mesmo quando o consumidor recebe um conteúdo poluído, a versão legítima também se propaga por alguns roteadores da rede. Como o `CCNCheck v.3` só remove as entradas poluídas do *cache*, as entradas legítimas são mantidas. Portanto, um conteúdo legítimo pode ser obtido tanto diretamente do publicador legítimo quanto de algum *cache* da rede. Na prática, o efeito causado pelo `CCNCheck v.3` é fazer com que cada nova tentativa de obter uma versão legítima de um conteúdo seja equivalente a primeira tentativa, mas em um cenário onde o número de publicadores

legítimos é maior do que o de publicadores maliciosos (cada *cache* contendo uma versão legítima do conteúdo requisitado pode ser visto como mais um publicador legítimo). Assim, em cada tentativa subsequente a primeira, a probabilidade do consumidor receber um conteúdo legítimo é maior que 52%. Por essa razão, independentemente do valor de p, mesmo antes do consumidor esgotar suas 10 tentativas ele sempre recebe uma versão legítima do conteúdo.

Para facilitar a compreensão deste comportamento, considere o seguinte exemplo: suponha que o consumidor tenha interesse em recuperar 20 conteúdos diferentes. Para simplificar os cálculos, a probabilidade do consumidor receber um conteúdo legítimo na primeira tentativa será arredondada para 50%. Além disso, será desconsiderado o efeito causado pelos *caches* na probabilidade do consumidor obter um conteúdo legítimo em tentativas subsequentes a primeira. Como resultado, em cada nova tentativa de recuperar uma versão legítima de um conteúdo, a probabilidade de sucesso será sempre 50%. A Tabela 6.1 mostra quantos conteúdos são requisitados, quantas dessas requisições retornam conteúdos legítimos e quantas retornam conteúdos maliciosos em cada tentativa realizada pelo consumidor.

Tabela 6.1: Para cada tentativa, são mostrados o total de conteúdos requisitados, quantas dessas requisições retornam conteúdos legítimos e quantas retornam conteúdos maliciosos.

Tentativa	Conteúdos Requisitados	Legítimos Recebidos	Maliciosos Recebidos
1	20	10	10
2	10	5	5
3	5	2	3
4	3	1	2
5	2	1	1
6	1	0	1
7	1	1	0

Inicialmente, o consumidor requisita 20 conteúdos. Como a probabilidade de receber uma versão legítima é 50%, metade dos conteúdos recebidos são legítimos e a outra metade são conteúdos poluídos. Assim, o consumidor faz uma nova tentativa, mas agora somente os 10 conteúdos que foram recebidos poluídos anteriormente são requisitados. Mais uma vez, somente metade das requisições retornam conteúdos legítimos, logo restam ainda 5 conteúdos poluídos a serem recuperados. Na terceira tentativa, esse 5 conteúdos são requisitados. No exemplo, duas dessas requisições retornam conteúdos legítimos, enquanto 3 dos conteúdos são recebidos poluídos. Na quarta tentativa esses três conteúdos são requisitados e dois deles são recebidos poluídos. Na sexta tentativa, resta apenas 1 conteúdo a

ser recuperado. Entretanto, esse conteúdo tem 50% de chance de ser recebido poluído, e, de acordo com a Tabela 6.1 é o que ocorre. Portanto, o consumidor realiza a 7ª tentativa de recuperar o conteúdo remanescente e, finalmente, obtém a versão legítima.

6.3.2 Resultados Para a Métrica TPP

Além de proporcionar um aumento na recuperação de conteúdos, o `CCNCheck` também reduz o desperdício de recursos da rede através da redução do total de pacotes com dados poluídos (`TPP`) trocados, como mostra o gráfico da Figura 6.2. Para $p = 0,1$, a utilização das versões 1 e 2 do `CCNCheck` propiciou uma redução de aproximadamente 14% no valor de `TPP`. Enquanto isso, para o mesmo valor de p, a utilização do `CCNCheck v.3` permitiu uma redução de 13% no total de pacotes com dados poluídos trocados na rede.

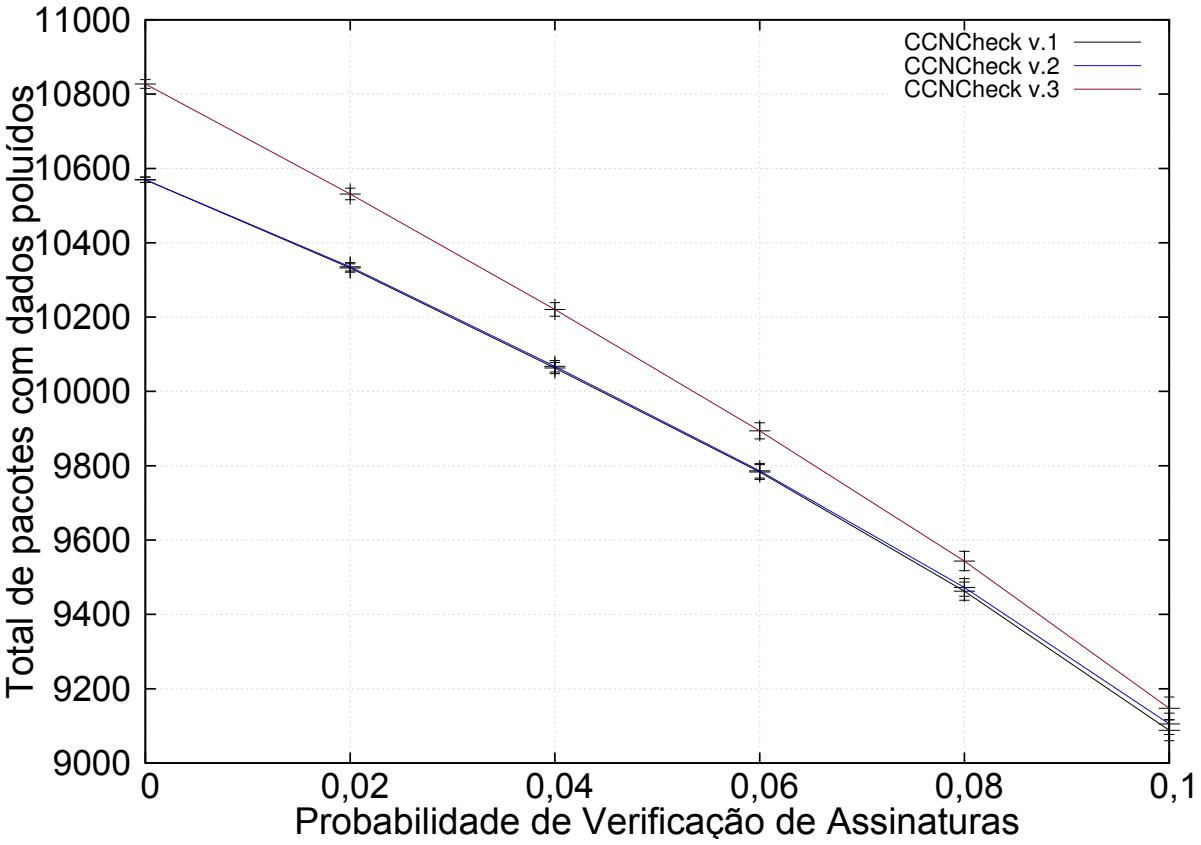

Figura 6.2: Total de pacotes com dados poluídos transmitidos no Cenário 1 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

Tanto o `CCNCheck v.2` quanto o `CCNCheck v.3` realizam a verificação de conteúdos em *cache*, mas a maneira e a frequência com que tais verificações são realizadas são diferentes nas duas versões. O `CCNCheck v.2` verifica entradas em *cache* sob demanda, ou

seja, somente o conteúdo em *cache* utilizado para responder um interesse recebido é verificado. Além disso, essa verificação ocorre com a mesma probabilidade que os conteúdos recebidos pelo roteador são verificados. Como essa probabilidade deve ser baixa, apenas uma pequena fração de entradas do *cache* são verificadas.

Considere o cenário ilustrado na Figura 6.3, onde todos os roteadores verificam conteúdos com probabilidade $p = 0,1$ e todos eles foram configurados com o `CCNCheck v.2`. Suponha que o consumidor requisite um conteúdo e obtenha uma versão poluída como resposta. Neste caso, o *cache* de R_1 já contém a versão poluída. Assim, quando o consumidor requisitar novamente o conteúdo, devido ao `CCNCheck v.2`, o roteador R_1 tem 10% de probabilidade de verificar o conteúdo em *cache* antes de utilizá-lo para responder ao consumidor. Suponha então que o conteúdo foi verificado pelo roteador. Consequentemente, a versão poluída em *cache* será descartada e o interesse recebido do consumidor será encaminhado para R_2. Apesar do `CCNCheck v.2` ter permitido que o interesse fosse encaminhado por mais um salto, o que não teria ocorrido se os roteadores estivessem utilizando o `CCNCheck v.1`, ou estivessem adotando o padrão da CCN, esse esforço terá sido em vão se a entrada poluída no *cache* de R_2 não for verificada e removida também. Mais que isso, se o interesse for respondido por R_2 com uma versão poluída, então o `CCNCheck` terá contribuído para aumentar `TPP`, sem aumentar `FCR`. A probabilidade de que R_1 e R_2 verifiquem a entrada poluída em *cache* antes de envia-la ao consumidor é de apenas 1%.

Por essa razão, o benefício proporcionado pelo `CCNCheck v.2` em relação aquele proporcionado pelo `CCNCheck v.1` é, em geral, pequeno. Além disso, a Figura 6.2 mostra que, para valores pequenos de p, as versões 1 e 2 do `CCNCheck` apresentam resultados sobrepostos. Isso significa que, por conta dos valores baixos de p, poucos conteúdos poluídos são verificados e removidos no roteador de borda. Entretanto, conforme p aumenta, as curvas dessas duas versões começam a se distanciar. Para $p = 0,1$, o `CCNCheck v.2` provê uma redução de $13,8\%$ em `TPP`, enquanto o `CCNCheck v.1` provê uma redução de 14% para esta métrica. Apesar da diferença de desempenho ser pequena, ela mostra que quando o valor de p aumenta, aumentam as verificações das entradas poluídas no *cache* dos roteadores e, por isso, o total de pacotes com dados poluídos trocados na rede é ligeiramente maior no `CCNCheck v.2` do que na versão 1.

Diferentemente do `CCNCheck v.2`, o `CCNCheck v.3` realiza verificação de conteúdos em *cache* periodicamente. Além disso, todas as entradas do *cache* são verificadas. Consequentemente, somente conteúdos legítimos são mantidos no *cache*. Assim, quando acontece de um conteúdo poluído ser recebido pelo consumidor, isso significa que ele foi re-

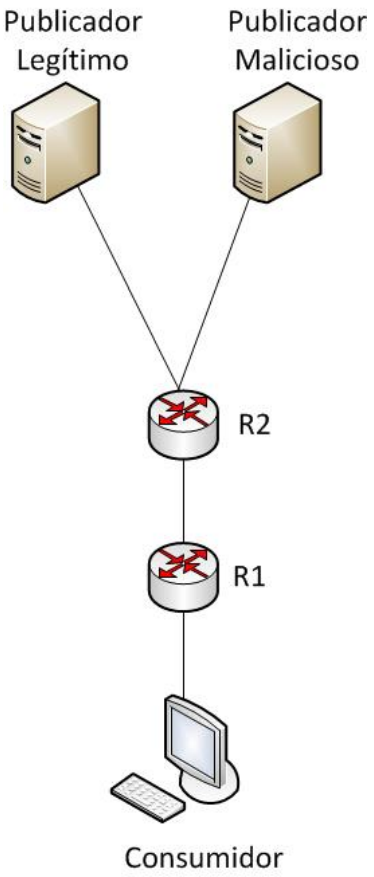

Figura 6.3: Exemplo de uma topologia onde o consumidor está ligado ao roteador R_1 que, por sua vez, está ligado ao roteador R_2 que, por fim, está ligado a um publicador legítimo e a um publicador malicioso.

cuperado a partir do publicador malicioso, e que, portanto, percorreu todo o caminho até o consumidor. Como na topologia em grade todos os caminhos entre o consumidor e os publicadores possuem no mínimo 21 saltos, então cada conteúdo poluído recebido representa um acréscimo de, no mínimo, 21 pacotes com dados poluídos à `TPP`. Adicionalmente, muitos conteúdos precisarem ser requisitados por diversas vezes antes que uma versão legítima seja obtida, acarretando em um aumento significativo para `TPP`, como pode ser notado na Figura 6.2.

6.3.3 Conclusão do Cenário

Com relação à métrica `FCR`, o `CCNCheck v.3` teve, incontestavelmente, o melhor desempenho. Com ele, foi possível recuperar 100% dos conteúdos requisitados, mesmo com $p = 0$, onde o *ovehead* causado ao processamento de pacotes é praticamente nulo. Por outro lado, as versões 1 e 2 do mecanismo apresentaram essencialmente o mesmo desempenho. Para $p = 0,1$, ambas permitiram que aproximadamente 88% dos conteúdos fossem

recuperados. Considerando agora a métrica `TPP`, o `CCNCheck v.3` foi o mecanismo que permitiu a maior redução do total de pacotes com dados poluídos trocados na rede. Entretanto, para cada valor de p avaliado, essa versão apresentou um valor para `TPP` maior do que nas outras duas versões. Os resultados gerados para as versões 1 e 2 do `CCNCheck` foram muito próximos, tendo ambas proporcionado uma redução de aproximadamente 14% para `TPP`. Assim, é possível concluir que o alto desempenho do `CCNCheck v.3` na métrica `FCR` compensa o maior desperdício de recursos da rede identificados pela métrica `TPP` e, por essa razão, essa versão é a mais indicada neste cenário.

6.4 Cenário 2: Probabilidades Estáticas e Iguais na Topologia Rocketfuel

Da mesma maneira que no Cenário 1, todos os roteadores da rede verificam conteúdos com a mesma probabilidade. Entretanto, neste cenário, ao invés de utilizar a topologia em grade, é utilizada a topologia *rocketfuel*. Nesta topologia, o número de saltos percorridos por um pacote de dados é consideravelmente menor do que na topologia em grade. Consequentemente, a eficiência do `CCNCheck` neste cenário é menor do que no Cenário 1.

6.4.1 Resultados Para a Métrica FCR

A Figura 6.4 mostra, para este cenário, como a fração de conteúdos recuperados varia para diferentes valores de p. Primeiramente, é preciso entender que o comportamento das versões do `CCNCheck` não muda por conta da alteração do cenário. Sendo assim, a diferença de desempenho observada neste cenário, em relação ao Cenário 1, é causada pela estrutura da topologia e pela metodologia utilizada nas simulações. A primeira diferença importante é que, para $p = 0$, o `CCNCheck v.3` permite que aproximadamente 84% dos conteúdos requisitados sejam recuperados. Isso representa uma redução de 16% em relação a mesma métrica no Cenário 1.

Para entender essa queda de desempenho, lembre-se de que na topologia *rocketfuel* as posições dos publicadores e do consumidor não são fixas. Na verdade, são formados 50 conjuntos contendo, cada um, três nós folha diferentes, escolhidos aleatoriamente. Esses três nós folha representam, durante uma rodada de simulação, o consumidor, o publicador legítimo e o publicador malicioso. Assim, dependendo da escolha desses três nós folha, a fração de conteúdos recebidos pelo consumidor pode variar drasticamente. De fato, com a utilização das versões 1 e 2 do `CCNCheck` e para $p = 0$, em aproximadamente metade

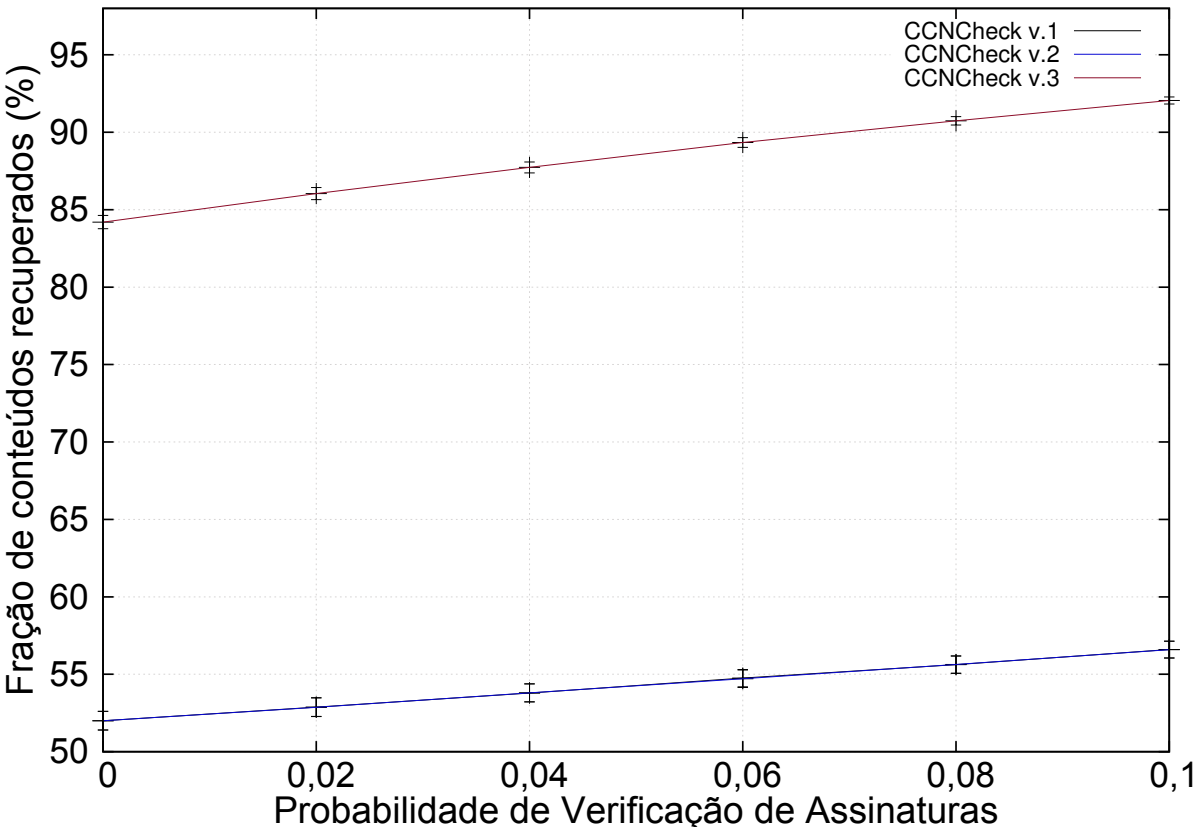

Figura 6.4: Fração de conteúdos recuperados no Cenário 2 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

dos 50 conjuntos o consumidor foi capaz de recuperar 100% dos conteúdos requisitados e, na outra metade, nenhum conteúdo foi recuperado pelo consumidor. Por essa razão, para $p = 0$, o gráfico da Figura 6.4 mostra que as versões 1 e 2 do `CCNCheck` permitem que pouco mais de 50% dos conteúdos sejam recuperados.

Quando o `CCNCheck v.3` é utilizado com $p = 0$, somente as simulações que utilizam um conjunto de nós folha onde nenhum conteúdo é recuperado têm a métrica `FCR` afetada. Entretanto, devido ao alto nível de poluição ao qual o consumidor é submetido nesses conjuntos, nem todos os conteúdos são recuperados. Como resultado, a fração de conteúdos recuperados passa de 52% para aproximadamente 84%, o que representa um aumento de 32%. Aumentando-se o valor de p, o valor de `FCR` para as três versões do `CCNCheck` também aumenta. Entretanto, o aumento proporcionado pela versão 3 é aproximadamente o dobro do proporcionado pelas versões 1 e 2.

6.4.2 Resultados Para a Métrica TPP

Como pode ser observado no gráfico da Figura 6.5, os resultados obtidos para a métrica `TPP` neste cenário são bastante distintos daqueles obtidos no Cenário 1. Claramente, a utilização do `CCNCheck v.3` faz com que o total de pacotes com dados poluídos trocados na rede seja bem maior do que nas versões 1 e 2 do `CCNCheck`. Como já observado na Seção 6.4.1, para $p = 0$ as versões 1 e 2 permitem a recuperação de 52% dos conteúdos, enquanto o `CCNCheck v.3` permite que 84% dos conteúdos sejam recuperados. Essa diferença é proporcionada pelas retransmissões realizadas pelo consumidor que, apesar de ineficazes nas versões 1 e 2 do mecanismo, são aproveitadas pela versão 3. Entretanto, quando uma retransmissão não é bem sucedida, o conteúdo poluído resultante inunda a rede e causa uma elevação de `TPP`. Como o `CCNCheck v.3` não consegue prover a recuperação de todos os conteúdos, alguns deles serão requisitados por até 10 vezes, antes do consumidor desistir. Consequentemente, a utilização do `CCNCheck v.3` faz com que um grande número de conteúdos poluídos se propaguem na rede. Nas versões 1 e 2, a maior parte desses conteúdos poluídos seriam retornados do *cache* do roteador de borda, resultando em um menor número de pacotes com dados poluídos sendo trocados na rede.

6.4.3 Conclusão do Cenário

As versões 1 e 2 do `CCNCheck` apresentaram resultados muito próximos para ambas as métricas avaliadas. Por outro lado, a diferença dos resultados obtidos com a utilização do `CCNCheck v.3` foram bastante significativas, tanto para `FCR` quanto para `TPP`. Para $p = 0,1$, o `CCNCheck v.3` permite que 92% dos conteúdos requisitados sejam recuperados, enquanto que para as outras duas versões essa fração é de apenas 56%. Entretanto, ao proporcionar esse benefício, o `CCNCheck v.3` paga o preço de gerar um maior desperdício dos recursos da rede, devido aos altos valores de `TPP` obtidos. Como exemplo, para $p = 0,02$, enquanto com o `CCNCheck v.2`, na média, foram trocados 4701.25 pacotes com dados poluídos, com a utilização do `CCNCheck v.3` esse valor subiu para 10276.85, uma diferença de aproximadamente 54%. Por essa razão, a decisão sobre qual versão do `CCNCheck` deve ser utilizada neste cenário depende do que se deseja priorizar. Se a satisfação do usuário for mais importante do que o custo gerado pelo desperdício dos recursos da rede, então o `CCNCheck v.3` é a versão indicada. Caso contrário, a preferível utilizar o `CCNCheck v.1` ou o `CCNCheck v.2`.

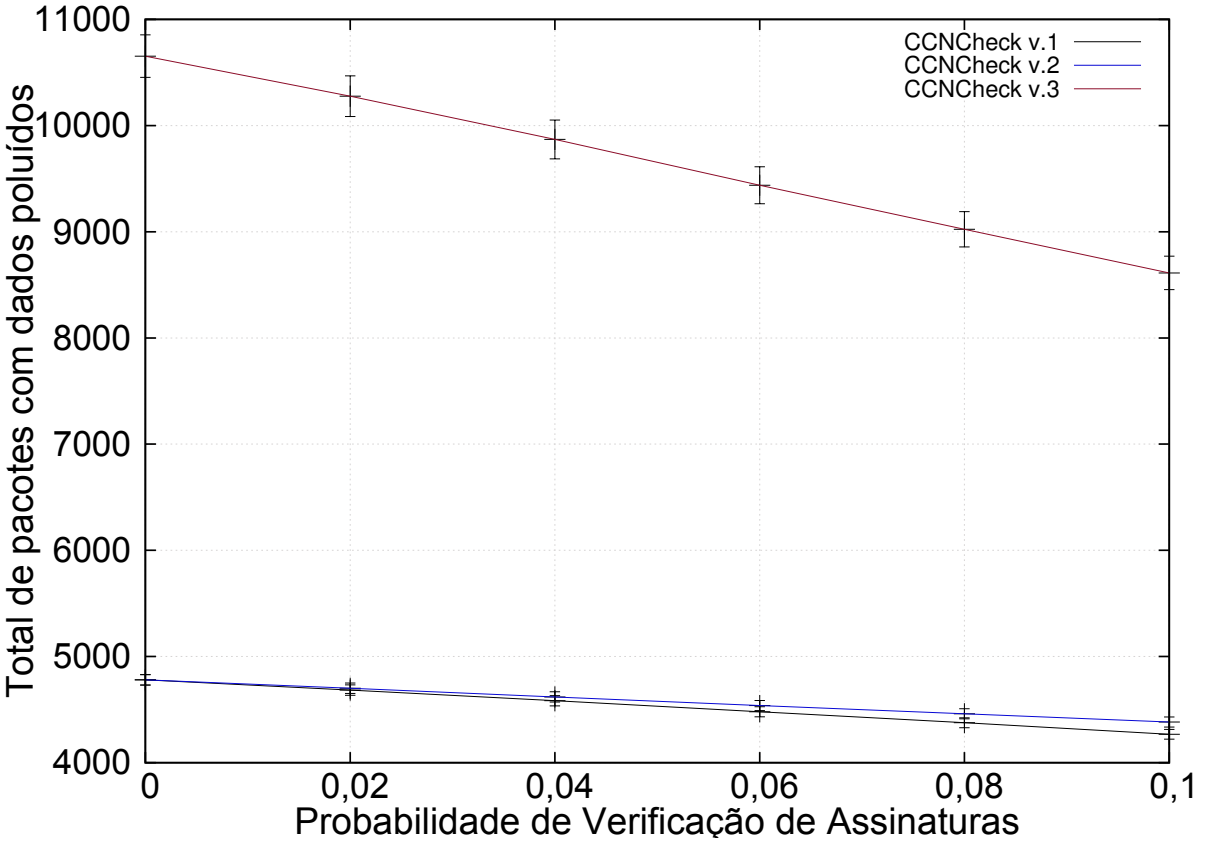

Figura 6.5: Total de pacotes com dados poluídos transmitidos no Cenário 2 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

6.5 Cenário 3: Probabilidade Estática e Mais Alta na Borda na Topologia Rocketfuel

Neste cenário, todos os roteadores de borda verificam conteúdos com a mesma probabilidade p_{borda} que, durante as simulações, variou de $0,5$ a $0,9$. Já os roteadores do núcleo da rede verificam conteúdos com probabilidade $p = 0,01$.

6.5.1 Resultados Para a Métrica FCR

A Figura 6.6 mostra o comportamento da fração de conteúdos recuperados para diferentes valores de p_{borda}. Com a utilização do `CCNCheck v.3`, aproximadamente 100% dos conteúdos são recuperados para todos os valores de p_{borda} avaliados. Como a probabilidade de verificação de conteúdos na borda da rede é alta, uma quantidade menor de conteúdos poluídos é admitida na rede. Consequentemente, já na primeira tentativa, é possível recuperar uma fração maior de conteúdos, como evidenciado pelo resultado obtido pelo `CCNCheck v.1` quando $p = 0,5$ (77% dos conteúdos são recuperados). Os

33% de conteúdos requisitados que foram recebidos poluídos na primeira tentativa são recuperados pelo consumidor através de tentativas subsequentes.

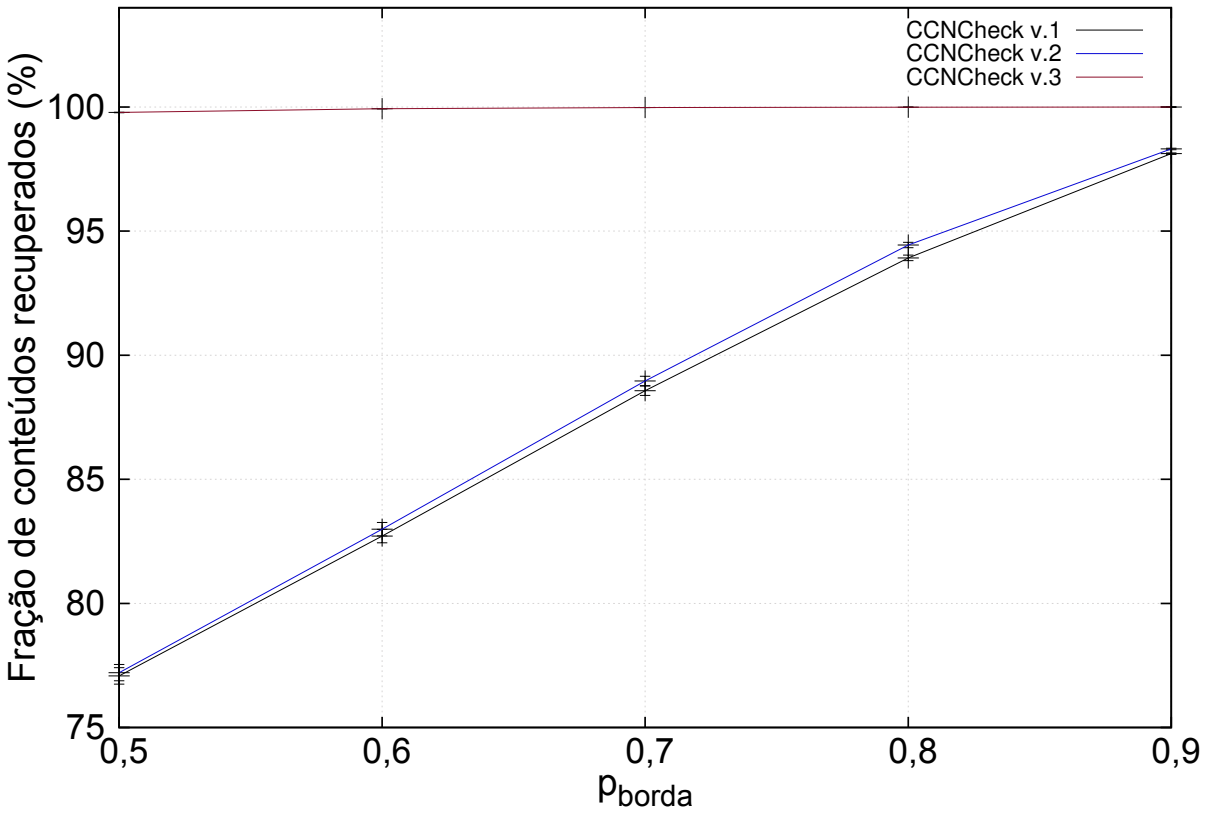

Figura 6.6: Fração de conteúdos recuperados no Cenário 3 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

O comportamento das curvas representando as versões 1 e 2 do `CCNCheck` são muito próximas, mas é possível perceber ligeiras diferenças. Para valores de p_{borda} entre $0,5$ e $0,8$, as duas curvas tendem a se afastar, mostrando que a versão 2 é mais eficiente que a versão 1. Entretanto, variando p_{borda} de $0,8$ até $0,9$, essa característica muda e as curvas passam a se aproximar. O afastamento das curvas pode ser explicado considerando-se que o `CCNCheck v.2` cria algumas oportunidades para que versões legítimas possam ser recuperadas, ao invés de versões poluídas oriundas dos `caches` da rede. A utilização de valores altos para p_{borda} faz com que mais conteúdos legítimos se propaguem na rede. Como resultado, um número maior de oportunidades criadas pelo `CCNCheck v.2` é aproveitado, fazendo com que a fração de conteúdos recebidos com esta versão seja maior do que com a versão 1. Conforme p_{borda} aumenta, menos conteúdos poluídos são admitidos na rede e, consequentemente, os roteadores terão uma quantidade reduzida de conteúdos poluídos armazenados em *cache*. Assim, o benefício adicional proporcionado pelo `CCNCheck v.2`

perde seu efeito para valores realmente altos de p_{borda}. Por essa razão, as curvas das versões 1 e 2 se aproximam para valores de p_{borda} entre $0,8$ e $0,9$.

6.5.2 Resultados Para a Métrica TPP

Como pode ser visto no gráfico da Figura 6.7, para $p_{borda} = 0,5$, o `CCNCheck v.3` acarreta numa quantidade de pacotes com dados poluídos significativamente maior que nas outras duas versões do `CCNCheck`. Entretanto, conforme p_{borda} aumenta, menos conteúdos poluídos são admitidos na rede e, como consequência, mais conteúdos legítimos se propagam no lugar de suas versões poluídas. Quando $p_{borda} = 0,9$, as curvas de todas as três versões do *CCNCheck* se aproximam muito e, na prática, a diferença de `TPP` entre elas pode ser desprezada.

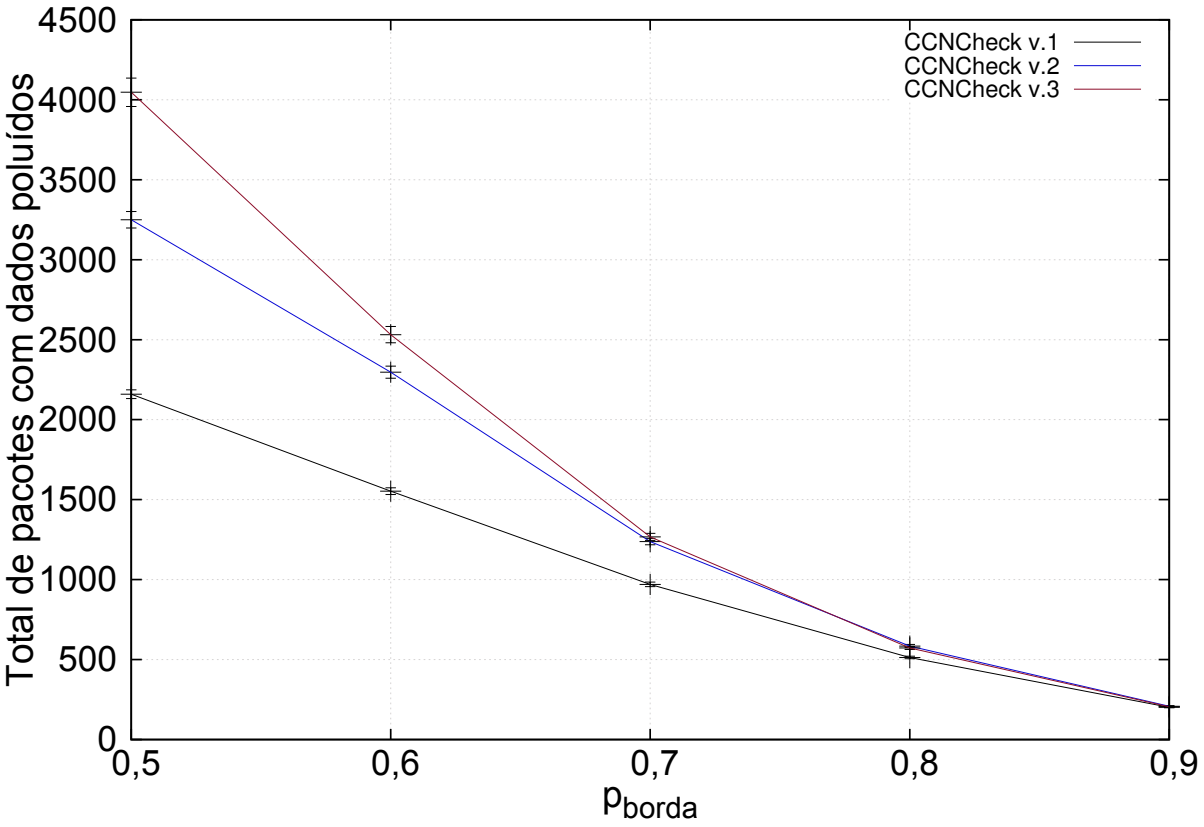

Figura 6.7: Total de pacotes com dados poluídos transmitidos no Cenário 3 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

6.5.3 Conclusão do Cenário

Com a utilização do `CCNCheck v.3`, para $p_{borda} = 0,5$ praticamente todos os conteúdos requisitados são recuperados. Por outro lado, para o mesmo valor de p_{borda}, as versões 1 e 2 do `CCNCheck` permitiram que aproximadamente 77% dos conteúdos fossem recuperados. Também para $p_{borda} = 0,5$, o `CCNCheck v.3` acarretou em aproximadamente 47% mais pacotes com dados poluídos trocados na rede que o `CCNCheck v.1`. Já para $p_{borda} = 0,9$, as versões 1 e 2 do `CCNCheck` produziram resultados muito próximos aos obtidos pelo `CCNCheck v.3`, mas com a vantagem de não precisarem de um processador extra para desempenharem sua função.

6.6 Cenário 4: Probabilidade Dinâmica na Borda na Topologia Rocketfuel

No Cenário 3, considerando as versões 1 e 2 do `CCNCheck`, para que quase todos os conteúdos requisitados sejam recuperados, é necessário que os roteadores de borda verifiquem os conteúdos com probabilidade $p_{borda} = 0,9$. Entretanto, na topologia *rocketfuel* existe apenas um publicador malicioso que está conectado a um pequeno conjunto de roteadores de borda (*gateways*). Para tais roteadores, boa parte dos 90% de conteúdos verificados serão realmente versões poluídas. Por outro lado, para os demais roteadores de borda, a maior parte dos conteúdos verificados serão versões legítimas. Uma vez que verificar conteúdos legítimos pode ser visto como uma penalidade, pois recursos são gastos em uma atividade que não traz nenhum benefício, seria interessante que os roteadores de borda mais afetados pela poluição verificassem conteúdos com maior probabilidade que os menos afetados.

Como já discutido no Capítulo 3, este trabalho propõe um procedimento simples e empírico para o cálculo de p_{borda}. Inicialmente, todos os roteadores de borda são configurados com $p_{borda} = p_{min}$, onde p_{min} é a menor probabilidade com que o roteador verifica conteúdos. Quando um conteúdo é verificado, o valor de p_{borda} é atualizado segundo a Equação 3.5. Os valores de k_1 e k_2 determinam o quanto o valor de p_{borda} é afetado pela recepção de um conteúdo poluído e de um conteúdo legítimo, respectivamente. Por exemplo, se $k_1 = 0,01$ e $k_2 = 0,001$, então seria necessário receber 10 conteúdos poluídos, sem verificar um conteúdo legítimo para que p_{borda} aumentasse em 10%. Por outro lado, para reduzir p_{borda} em 10% seria necessário receber 100 conteúdos legítimos sem verificar um conteúdo poluído. Uma vez que a recepção de um conteúdo poluído representa um evento

mais importante do que a recepção de um conteúdo legítimo, k_2 deve ser razoavelmente menor que k_1.

$$p_{borda} = \begin{cases} \max(1, (p + k_1)) & \text{para c poluído} \\ \max(p_{min}, (p - k_2)) & \text{para c legítimo} \end{cases} \tag{6.1}$$

6.6.1 Resultados Para a Métrica FCR

A Figura 6.8 mostra o comportamento da fração de conteúdos recuperados para diferentes valores de p_{min}. Para $p_{min} = 0,1$, o `CCNCheck v.3` permitiu que todos os conteúdos requisitados fossem recuperados. Enquanto isso, as outras duas versões do `CCNCheck` acarretaram em um valor entre 92 e 94% para `FCR`.

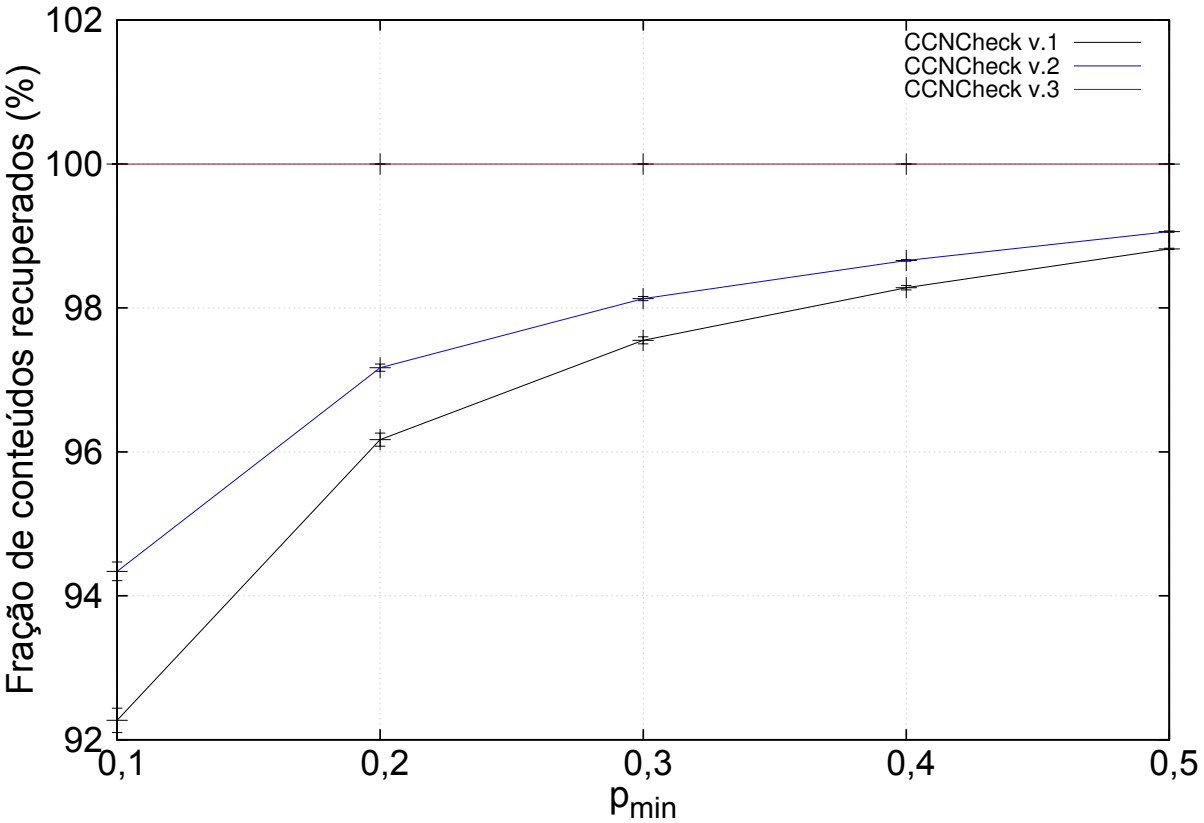

Figura 6.8: Fração de conteúdos recuperados no Cenário 4 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

Quando $p_{min} = 0,5$, o `CCNCheck v.2` permite que aproximadamente 99% dos conteúdos sejam recuperados. Por outro lado, no Cenário 3, para obter um resultado próximo a esse valor, foi preciso fazer com que todos os roteadores de borda verificassem conteúdos com probabilidade $p_{borda} = 0,9$. Porém, quando $p_{min} = 0,5$, isso não significa que

os roteadores de borda irão verificar conteúdos com essa probabilidade. Ao invés disso, fazer $p_{min} = 0,5$ implica que nenhum roteador de borda irá verificar conteúdos com uma probabilidade menor que $0,5$. No pior caso, todos os roteadores de borda poderiam, na verdade, passar a verificar conteúdos com probabilidade igual a 100%. Entretanto, como mostra o gráfico da Figura 6.9, para $p_{min} = 0,5$, aproximadamente 80% dos roteadores de borda se mantém verificando conteúdos com probabilidade mínima. Portanto, a utilização da Equação 6.1 para o cálculo de p_{borda} permite que a carga de verificação seja concentrada nos roteadores de borda mais afetados pela poluição, enquanto os demais verificam conteúdos com a probabilidade mínima.

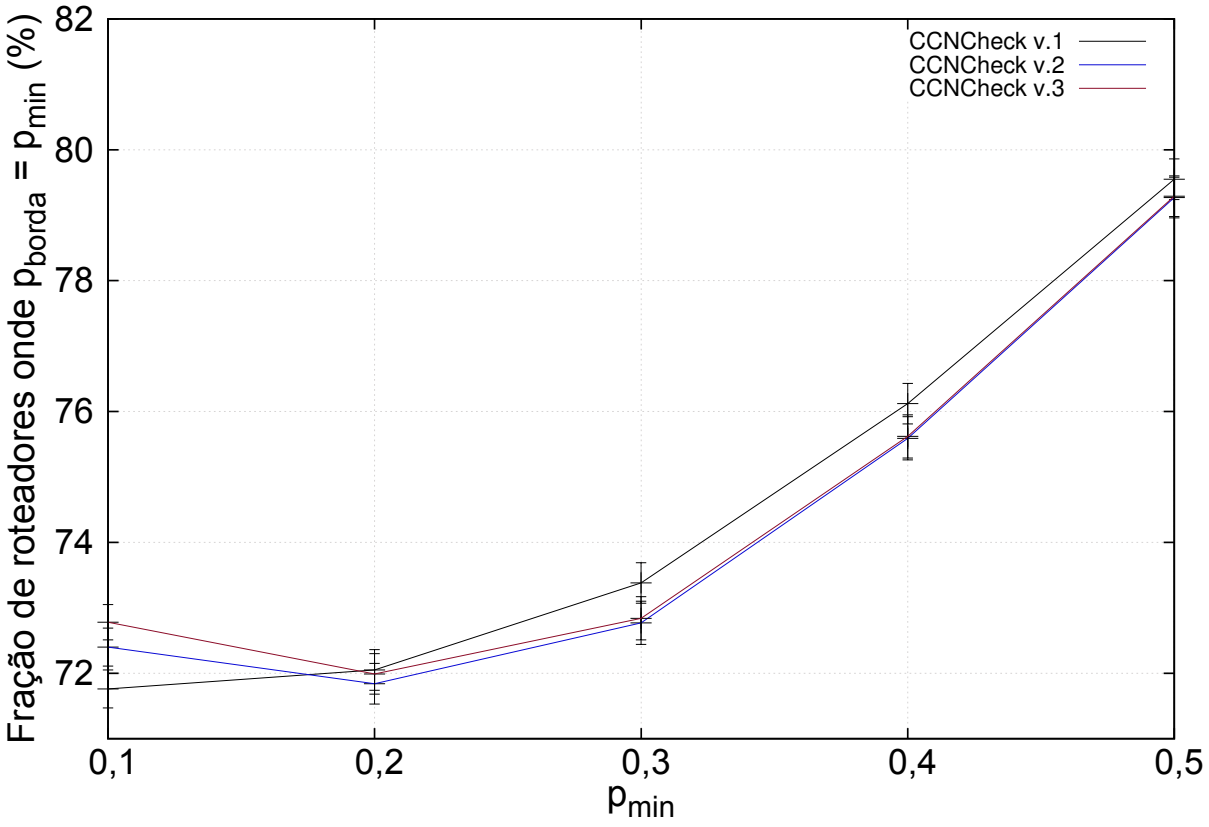

Figura 6.9: Fração de roteadores verificando conteúdos com probabilidade mínima.

6.6.2 Resultados Para a Métrica TPP

Com o auxílio do gráfico mostrado na Figura 6.10, é possível perceber que, neste cenário, o comportamento da métrica `TPP` para as três versões do `CCNCheck` é invertido em relação aos outros cenários previamente analisados. Agora, é o `CCNCheck v.1` que acarreta em uma maior quantidade de pacotes com dados poluídos sendo trocados na rede. Essa inversão ocorre, pois, com o cálculo adaptativo de p_{borda}, uma quantidade

muito pequena de conteúdos poluídos consegue entrar na rede. Por essa razão, a remoção de entradas poluídas dos *caches* da rede, realizadas pelas versões 2 e 3 do `CCNCheck`, permite que novas tentativas de recuperar um conteúdo legítimo sejam bem sucedidas. Enquanto isso, se o `CCNCheck v.1` for utilizado, após receber um conteúdo poluído, todas as novas tentativas de receber uma versão legítima serão respondidas pela versão poluída armazenada no *cache* do roteador de borda.

Apesar disso, a tendência das curvas representando as três versões do `CCNCheck` na Figura 6.10 é bastante parecida com a do Cenário 3 (Figura 6.7). Conforme P_{min} aumenta, mais conteúdos são verificados pelos roteadores de borda e, com isso, p_{borda} reage mais rapidamente a poluição, evitando que uma fração maior de conteúdos poluídos sejam admitidos na rede. Consequentemente, mais conteúdos legítimos são recuperados pelo consumidor já na primeira tentativa, o que faz com que a curva representando o `CNCheck v.1` tenha uma tendência decrescente. Como mais conteúdos são recuperados na primeira tentativa, menor é a influência da verificação de conteúdos em *cache* provida pelas versões 2 e 3 do `CCNCheck`. As três curvas tendem a se encontrar conforme mais e mais conteúdos vão sendo recuperados na primeira tentativa. No limite, todos os conteúdos são recuperados na primeira tentativa e as três curvas estarão completamente sobrepostas.

6.6.3 Conclusão do Cenário

Neste cenário, é simples ver que o `CCNCheck v.3` é a versão mais indicada. Com relação a métrica `FCR`, quando $p_{min} = 0,1$ todos os conteúdos requisitados são recuperados. Para esse mesmo valor de p_{min}, aproximadamente 73% dos roteadores de borda se mantém verificando apenas 10% dos conteúdos. Também com $p_{min} = 0,1$, o `CCNCheck v.3` acarretou na menor quantidade de pacotes com dados poluídos trocados na rede, considerando todas as outras duas versões do `CCNCheck`.

Se, por acaso, o `CCNCheck v.3` não puder ser implementado, devido a necessidade do segundo processador, então a versão mais indicada neste cenário é o `CCNCheck v.2`. Para $p_{min} = 0,1$, o `CCNCheck v.2` permitiu a recuperação de 2% mais conteúdos do que a versão 1. Apesar dessa diferença ser pequena, é na métrica `TPP` que o `CCNCheck v.2` se destaca em relação a versão 1. Com a versão 2, foram trocados na rede aproximadamente 23% menos pacotes com dados poluídos do que na versão 1.

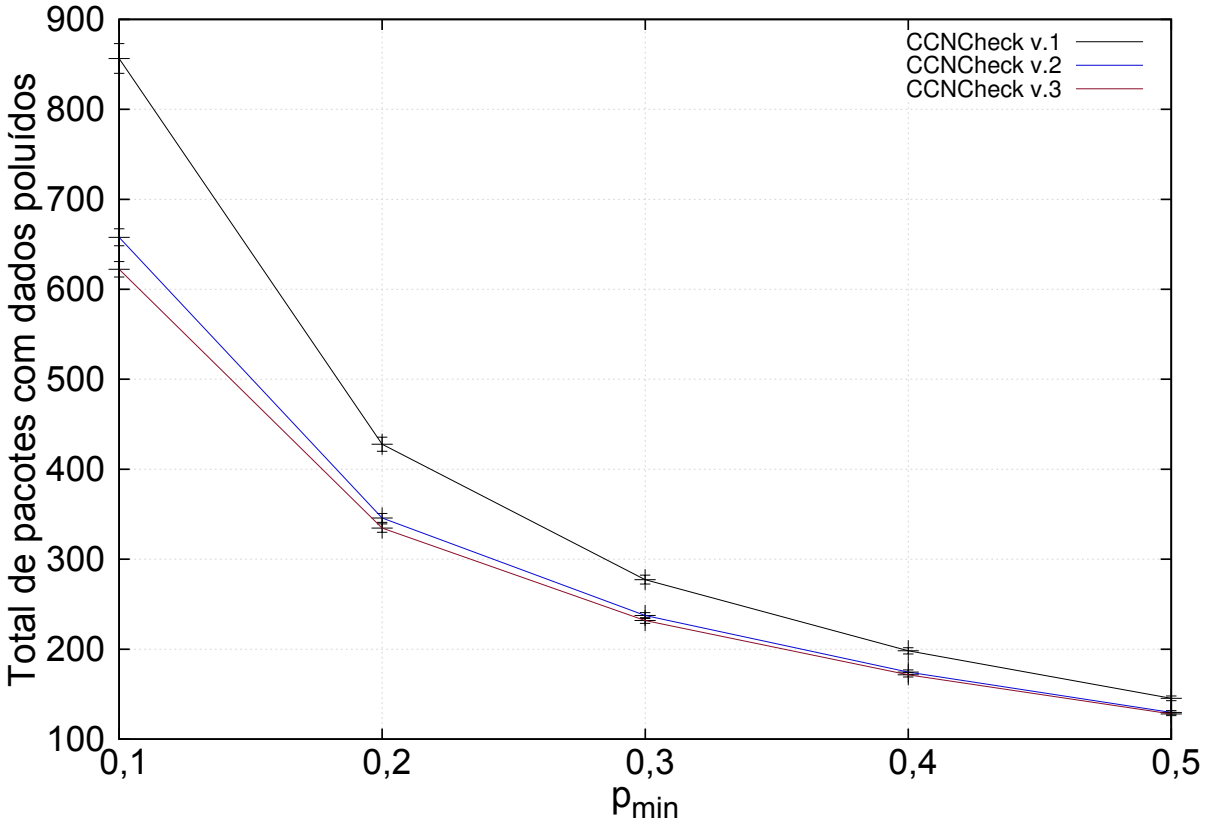

Figura 6.10: Total de pacotes com dados poluídos transmitidos no Cenário 4 para as três versões do `CCNCheck`. As barras de erro foram construídas a partir de um intervalo com 95% de confiança.

6.7 Conclusão do Capítulo

O objetivo deste capítulo foi avaliar as três versões propostas para o `CCNCheck` com as três abordagens de utilização definidas no Capítulo 3. Na abordagem 1, todos os roteadores foram configurados para verificar conteúdos com a mesma probabilidade. Consequentemente, a carga de verificação de conteúdos é distribuída uniformemente por todos os roteadores da rede. Devido ao *overhead* gerado pela verificação de assinaturas, os valores de p considerados nos Cenários 1 e 2, que utilizam essa abordagem, tiveram que ser necessariamente pequenos. Consequentemente, como ilustrado pela diferença de desempenho de todas as versões do `CCNCheck` nos Cenários 1 e 2, a eficiência na mitigação da poluição utilizando esta abordagem depende fortemente da topologia empregada. Além disso, esta abordagem não provê justiça, pois considera que todos os roteadores da rede possuem os mesmos recursos e a mesma carga de tráfego. Esta situação é análoga a um governo que cobra o mesmo valor k de imposto de rendo para ricos e pobres. Se o valor de k for muito alto, os ricos ainda terão facilidade de pagar, mas os pobres se endividarão. Da

mesma maneira, nem todos os roteadores da rede possuem os mesmos recursos e a mesma carga de tráfego. Dessa forma, dependendo do valor de p, alguns roteadores podem não suportar a carga de verificação de assinaturas, enquanto outros poderão executar essa tarefa facilmente.

A abordagem 2, onde os roteadores de borda verificam conteúdos com uma probabilidade p_{borda}, maior que a dos roteadores do núcleo da rede, tenta tornar a distribuição da carga de verificação de conteúdos mais justa. Aqueles roteadores com menor carga de tráfego de dados (roteadores de borda) podem suportar uma probabilidade de verificação mais alta que os demais. Essa abordagem é menos dependente da topologia, pois conteúdos poluídos somente serão aceitos na rede com probabilidade $1 - p_{borda}$. Apesar de ser melhor que a abordagem 1, a abordagem 2 ainda peca na justiça. Nem todos os roteadores de borda estão submetidos ao mesmo nível de poluição. Por essa razão, o valor de p_{borda} deveria se adaptar a situação de cada roteador. Se um roteador de borda está recebendo muitos conteúdos poluídos, então ele deve verificar assinaturas com uma valor de p_{borda} mais alto. Por outro lado, se o roteador de borda só recebe conteúdos legítimos, ele deve verificar assinaturas com uma probabilidade mínima. Como resultado, a terceira abordagem de utilização do `CCNCheck` foi proposta e avaliada no Cenário 4. Através dos resultados obtidos, foi possível perceber que, para as duas métricas avaliadas, o desempenho de todas as versões do `CCNCheck` foi maximizada nesse cenário.

Com relação ao desempenho das versões do `CCNCheck`, a versão 3 do mecanismo foi superior para a métrica `FCR` em todos os cenários avaliados. No Cenário 4, o `CCNCheck v.3` também apresentou os melhores resultados para a métrica `TPP`. Para os outros cenários, a eficiência do `CCNCheck v.3` com relação a métrica `TPP` foi inferior a das outras duas versões. Em especial, no Cenário 2, a diferença de performance entre a versão 3 e as demais quanto a esta métrica foi bastante significativa. Assim, é possível concluir que se a prioridade do mecanismo for permitir a maior fração possível de conteúdos recuperados pelos consumidores, então o `CCNCheck v.3` deve ser utilizado. Por outro lado, se a prioridade for a redução do desperdício de recursos da rede, então cada um dos cenários precisa ser analisado individualmente.

Com o cálculo de p_{borda} sendo realizado de maneira adaptativa, de acordo o nível de polução ao qual os roteadores de borda estão submetidos, a carga de verificação de conteúdos é concentrada naqueles roteadores mais afetados pela poluição. O Cenário 1, por outro lado, adota uma política oposta, onde a carga de verificação de conteúdos é distribuída uniformemente por todos os nós da rede. Neste caso, a eficiência dessa

abordagem de utilização do `CCNCheck` passa a depender fortemente da topologia utilizada.

Capítulo 7

Conclusão e Perspectivas Futuras

Este trabalho teve por objetivo avaliar a verificação probabilística de assinaturas como uma proposta de mitigação para ataques de poluição de conteúdos na CCN. Para tanto, foram desenvolvidas três versões do mecanismo batizado como `CCNCheck`. O `CCNCheck v.1` realiza a verificação probabilística de conteúdos quando estes são recebidos pelo roteador. Caso o conteúdo seja poluído, ele é simplesmente descartado, mas o interesse para o conteúdo continua pendente na `PIT` do roteador. O `CCNCheck v.2` realiza as mesmas operações que a primeira versão. Adicionalmente, quando um interesse é recebido e o roteador possui o conteúdo requisitado em *cache*, antes de enviá-lo como resposta, o `CCNCheck v.2` realiza a verificação probabilística de sua assinatura. Se o conteúdo for verificado como sendo poluído, o conteúdo não é encaminhado e a entrada do *cache* correspondente é eliminada. O `CCNCheck v.3` possui as mesmas funcionalidades que o `CCNCheck v.2`, mas também realiza a verificação de todo o `CS` do roteador periodicamente. Para que a verificação periódica do *cache* não interfira com o processamento de pacotes realizado pelo roteador, é sugerido que tal verificação seja executada em um processador exclusivo para esse fim.

Além do mecanismo em si, também foram propostas e avaliadas três abordagens de utilização para o `CCNCheck`. Na primeira, todos os roteadores da rede foram configurados com o mesmo valor para a probabilidade de verificação de assinaturas (p). Foram realizadas simulações para avaliar o desempenho do `CCNCheck` quanto ao aumento da fração de conteúdos recebidos pelo consumidor e a redução do total de pacotes com dados poluídos trocados na rede. Foram utilizadas duas topologias com esta abordagem, uma topologia em grade com 21 linhas e 21 colunas (Cenário 1) e a topologia *rocketfuel* (Cenário 2). Conforme mostraram os resultados, no Cenário 1, mesmo quando $p = 0$, o `CCNCheck v.3` foi capaz de recuperar 100% dos conteúdos requisitados, representando um aumento de

aproximadamente 48% em relação ao comportamento padrão da CCN. Por outro lado, o desempenho do `CCNCheck v.3` na métrica `TPP` foi ligeiramente menor do que a das outras duas versões. Para $p = 0,1$, a versão 3 do `CCNCheck` proporcionou uma redução de 13% para `TPP` em relação ao comportamento padrão da CCN . Por outro lado, a redução proporcionada pelas versões 1 e 2 foi de aproximadamente 14%.

No Cenário 2, o `CCNCheck v.3` manteve o melhor resultado para a métrica `FCR`, permitindo que 40% mais conteúdos fossem recuperados quando $p = 0,1$, se comparado ao comportamento padrão da CCN. As versões 1 e 2 obtiveram resultados semelhantes e pouco expressivos, permitindo um aumento de apenas 4% para `FCR`. Por outro lado, o desempenho do `CCNCheck v.3` quanto a métrica `TPP` não foi bom. Para $p = 0,1$, o `CCNCheck v.3` proporcionou um aumento de aproximadamente 44% em `TPP`. Por outro lado, para o mesmo valor de p, as versões 1 e 2 proporcionaram uma redução de aproximadamente 11% em `TPP`. Assim, é preciso priorizar a métrica mais importante para decidir qual versão co `CCNCheck` utilizar neste cenário.

O terceiro cenário avaliado emprega a topologia *rocketfuel* e a segunda abordagem de utilização do `CCNCheck` proposta no Capítulo 3. Nele, os roteadores de borda verificam conteúdos com uma probabilidade mais alta (p_{borda}) do que aquela utilizada pelos roteadores do núcleo da rede. Neste cenário, mais uma vez o desempenho do `CCNCheck v.3` foi muito superior ao das demais versões no que diz respeito a métrica `FCR`. Em particular, para $p_{borda} = 0,5$, aproximadamente 100% dos conteúdos foram recuperados. Se comparado ao comportamento tradicional da CCN, o `CCNCheck v.3` provê um aumento de aproximadamente 48% para a fração de conteúdos recuperados. Já as versões 1 e 2 proporcionam um aumento de aproximadamente 25% para `FCR`. Com relação à métrica `TPP`, para $p_{borda} = 0,5$, a utilização `CCNCheck v.3` gera aproximadamente 47% mais pacotes com dados poluídos que o `CCNCheck v.1` e mais 20% que o `CCNCheck v.2`. Conforme p_{borda} aumenta, a diferença de desempenho entre as três versões diminui. Quando $p_{borda} = 0,9$, as três versões do `CCNCheck` apresentam uma redução de aproximadamente 96% em `TPP`, se comparados ao comportamento tradicional da CCN. No Cenário 2, para obter o melhor desempenho para a métrica `FCR`, proporcionado pelo `CCNCheck v.3`, era necessário sacrificar bastante a métrica `TPP`, e vice versa. Já no Cenário 3, é possível otimizar o desempenho para `FCR` sem causar um impacto negativo na métrica `TPP`. Para tanto, basta utilizar o `CCNCheck v.3` com $p_{borda} = 0,9$. Entretanto, ao fazer isso, aumenta-se o *overhead* causado pela verificação de conteúdos.

No quarto cenário avaliado, a probabilidade p_{borda} é calculada dinamicamente, de

acordo com o nível de poluição ao qual o roteador de borda está submetido. Para tanto, foi proposto um processo simples onde a recepção de 5 conteúdos poluídos aumenta em 10% o valor de p_{borda} e a recepção de 50 conteúdos legítimos reduz em 10% o valor de p_{borda}. Inicialmente, $p_{borda} = p_{min}$, onde p_{min} é a menor probabilidade de verificação utilizada. Neste cenário, todos as versões do `CCNCheck` obtiveram um bom desempenho tanto para `FCR` quanto para `TPP`. Entretanto, o `CCNCheck v.3` foi superior, não somente para a métrica `FCR`, como ocorreu nos cenários 1, 2 e 3, mas também para a métrica `TPP`. Para $p_{min} = 0,5$, o `CCNCheck v.3` propiciou a recuperação de 100% dos conteúdos e reduziu em 97% o total de pacotes com dados poluídos trocados na rede. Além disso, para esse valor de p_{min}, aproximadamente 80% dos roteadores de borda se mantiveram verificando conteúdos com a probabilidade mínima. Assim, é possível concluir que o cálculo dinâmico de p_{borda} permite que a carga de verificação seja concentrada nos roteadores mais afetados pela poluição de conteúdos, reduzindo assim a quantidade de verificações desnecessárias.

Adicionalmente, foi avaliado de que maneira diferentes topologias de rede podem influenciar no desempenho do CCNcheck. Esta análise foi dividida em duas partes. Na primeira, procurou-se entender como o número de saltos entre o consumidor e um publicador malicioso interfere na probabilidade do consumidor receber um conteúdo poluído. Como resultado, concluiu-se que tal probabilidade diminui exponencialmente com o número de saltos. A segunda análise realizada foi quanto ao número de ramos. Neste caso não é possível afirmar se a probabilidade do consumidor receber um conteúdo poluído aumenta ou diminui com o número de ramos. Se o número de conteúdos poluídos que chegam pelo ramo adicionado é maior que no ramo original, então a probabilidade aumenta. Por outro lado, se o número de conteúdos poluídos que chegam pelo novo ramo for menor do que no ramo original, então a probabilidade diminui. Por fim, se ambos os ramos forem atravessados pela mesma quantidade de conteúdos poluídos, então a probabilidade do consumidor receber um conteúdo poluído permanece inalterada.

As contribuições geradas a partir deste trabalho podem ser sumarizadas da seguinte maneira:

i) Estudo do estado da arte sobre as questões de segurança a respeito da CCN. Como resultado, foi produzido o minicurso intitulado `Segurança em Redes Centradas em Conteúdo: Vulnerabilidades, Ataques e Contramedidas`, apresentado no XII Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais (SBSeg) no ano de 2012.

ii) Além do minicurso, foi submetido o artigo `CCNcheck: um mecanismo de mitigação`

`para poluição de conteúdos em Redes Centradas em Conteúdo` para o XIII Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais (SBSeg) no ano de 2013. Atualmente o aceite está sendo aguardado.

iii) Projeto e implementação do `CCNCheck v.1`, `CCNCheck v.2` e `CCNCheck v.3`;

iv) Análise da influência da topologia na eficiência do `CCNCheck`;

v) Proposta de três abordagens distintas para utilização do `CCNCheck`.

vi) Proposta de um processo simples e empírico para o cálculo da probabilidade de verificação de assinaturas de maneira adaptativa.

Este trabalho representou um primeiro passo na proposta do `CCNCheck`. Por essa razão, existem ainda muitas lacunas a serem exploradas. Todas as simulações realizadas consideraram sempre a existência de um único consumidor que requisita conteúdos a uma determinada frequência. Seria interessante incluir mais consumidores requisitando conteúdos com base em distribuições mais próximas da realidade ou de acordo com algum *trace*. Adicionalmente, poderiam ser incluídos mais publicadores, tanto legítimos quanto maliciosos. Dessa forma, seria possível avaliar como o `CCNCheck` se comporta em diferentes cenários de ataque, incluindo aqueles onde publicadores maliciosos se unem a consumidores maliciosos para ampliar a disseminação de conteúdos poluídos na rede.

Com relação ao funcionamento do `CCNCheck`, uma melhoria interessante seria fazer com que quando um conteúdo poluído fosse verificado, o roteador enviasse um alerta a seus vizinhos informando o *hash* do conteúdo poluído identificado. Ao receber o alerta o roteador poderia tomar duas atitudes. Na primeira ele verifica se existe algum conteúdo em seu `CS` cujo *hash* é igual aquele informado no alerta. Caso seja encontrado, o conteúdo é removido. Adicionalmente, o roteador pode armazenar o *hash* informado no alerta em uma lista negra. Assim, quando um conteúdo é recebido, seu *hash* é calculado e comparado com aqueles presentes na lista negra. Se uma correspondência for encontrada, o conteúdo é descartado. Caso contrário, sua assinatura será verificada de maneira probabilística.

Uma vez que os consumidores sempre verificam conteúdos, eles também poderiam enviar alertas para a rede avisando sobre um conteúdo poluído. A utilização de alertas pode aumentar a eficiência do `CCNCheck`, mas sua adoção pode trazer outros riscos à segurança. Seria necessário criar uma relação de confiança entre os roteadores, mas mais importante, entre os roteadores e os consumidores.

Por fim, dos cenários avaliados, aquele onde os valores de p_{borda} são calculados dinamicamente se mostrou o mais promissor. Entretanto, o procedimento proposto neste trabalho para calcular dinamicamente os valores de p_{borda}, apesar de ter apresentado bons resultados nas simulações realizadas, é muito simples e empírico. Na realidade os incrementos e decrementos realizados quando são recebidos conteúdos poluídos e legítimos, respectivamente, poderiam ser projetados de diversas outras maneiras. Por essa razão, é necessário realizar uma análise mais aprofundada para que os valores de p_{borda} sejam calculados de maneira mais consistente. Além disso, os roteadores do núcleo da rede também poderiam ser configurados para verificar assinaturas com valores de p calculados dinamicamente. Em tal cálculo, seria interessante considerar questões relativas a topologia da rede, como por exemplo a distância do roteador até o consumidor, ou a centralidade do roteador.

Referências

[1] CCN Interest Message. Online: http://www.ccnx.org/releases/latest/doc/technical/InterestMessage.html, (2013).

[2] NS-3 Simulator. Online: http://www.nsnam.org, (2013).

[3] Abadi, M. On SDSI's Linked Local Name Spaces. Em *Journal of Computer Security* (1998), vol. 6, páginas 6–1.

[4] Afanasyev, A.; Moiseenko, I.; Zhang, L. ndnSIM: NDN simulator for NS-3. Relatório Técnico NDN-0005, (2012).

[5] Arianfar, S.; Koponen, T.; Raghavan, B.; Shenker, S. On preserving privacy in content-oriented networks. Em *Proceedings of the ACM SIGCOMM workshop on Information-centric networking* (2011), ICN '11, ACM, páginas 19–24.

[6] DiBenedetto, S.; Gasti, P.; Tsudik, G.; Uzun, E. ANDaNA: Anonymous Named Data Networking Application. Em *NDSS 2012* (2012).

[7] Dingledine, R.; Mathewsonn, N.; Syverson, P. Tor: The second-generation onion router. Em *The 13th USENIX Security Symposium* (2004), páginas 21–21.

[8] Ellison, C. M.; Inc, C.; Frantz, B.; Lampson, B.; Rivest, R.; Thomas, B. M.; Ylonen, T. SPKI certificate theory. IETF Network Working Group RFC 2693, (1999).

[9] Gasti, P.; Tsudik, G.; Uzun, E.; Zhang, L. DoS & DDoS in Named-Data Networking. Em *Proceedings of International Conference on Computer Communications and Networks (ICCCN 2013)* (2013).

[10] Ioannidis, J.; Bellovin, S. M. Implementing Pushback: Router-Based Defense Against DDoS Attacks. Em *Network and Distributed System Security Symposium - NDSS* (2002).

[11] Jacobson, V.; Smetters, D. K.; Thornton, J. D.; Plass, M.; Briggs, N.; Braynard, R. Networking named content. Em *Commun. ACM* (2012), vol. 55, ACM, páginas 117–124.

[12] Jacobson, V.; Smetters, D. K.; Thornton, J. D.; Plass, M. F.; Briggs, N. H.; Braynard, R. L. Networking named content. Em *Proceedings of the 5th international conference on Emerging networking experiments and technologies* (2009), CoNEXT '09, ACM, páginas 1–12.

[13] Lee, U.; Choi, M.; Cho, J.; Sanadidi, M. Y.; Gerla, M. Understanding pollution dynamics in p2p file sharing. Em *Proceedings of the 5th International Workshop on Peer-toPeer Systems (IPTPS'06* (2006).

[14] Leiner, B. M.; Cerf, V. G.; Clark, D. D.; Kahn, R. E.; Kleinrock, L.; Lynch, D. C.; Postel, J.; Roberts, L. G.; Wolff, S. A brief history of the internet. Em *SIGCOMM Comput. Commun. Rev.* (2009), vol. 39, ACM, páginas 22–31.

[15] Liang, J.; Kumar, R.; Xi, Y.; Ross, K. Pollution in p2p file sharing systems. Em *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE* (2005), vol. 2, páginas 1174–1185 vol. 2.

[16] Lou, X.; Hwang, K. Collusive piracy prevention in p2p content delivery networks. Em *IEEE Transactions on Computers* (2009), vol. 58, páginas 970–983.

[17] Moreira, M. D. D.; Fernandes, N. C.; Costa, L. H. M. K.; Duarte, O. C. M. B. Internet do Futuro: Um Novo Horizonte. Em *Simpósio Brasileiro de Redes de Computadores - SBRC'2009* (2009), páginas 1–59.

[18] Myers, M.; Ankney, R.; Malpani, A.; Galperin, S.; Adams, C. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. IETF Network Working Group RFC 2560, (1999).

[19] Passarella, A. Review: A survey on content-centric technologies for the current internet: Cdn and p2p solutions. Em *Comput. Commun.* (2012), vol. 35, Elsevier Science Publishers B. V., páginas 1–32.

[20] Ribeiro, I. C. G.; Guimarães, F. Q.; Kazienko, J.; Rocha, A. A. A.; Velloso, P. B.; Moraes, I. M.; De Albuquerque, C. V. Segurança em Redes Centradas em Conteúdo: Vulnerabilidades, Ataques e Contramedidas. Em *Minicurso - Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg* (2012), páginas 101–150.

[21] Rodriguez, P.; Tan, S.-M.; Gkantsidis, C. On the feasibility of commercial, legal p2p content distribution. Em *SIGCOMM Comput. Commun. Rev.* (2006), vol. 36, ACM, páginas 75–78.

[22] Schwetzingen, T. L. Security & Scalability of Content-Centric Networking. Master's thesis, Technische Universitat Darmstadt, (2010).

[23] Smetters, D.; Jacobson, V. Securing Network Content. Tech. Rep. TR-2009-1, Xerox Palo Alto Research Center - PARC, (2009).

[24] Spring, N.; Mahajan, R.; Wetherall, D. Measuring ISP topologies with rocketfuel. Em *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications* (2002), ACM SIGCOMM, páginas 133–145.

[25] Wählisch, M.; Schmidt, T. C.; Vahlenkamp, M. Backscatter from the data plane — threats to stability and security in information-centric networking. Em *ArXiv* (2012), vol. abs/1205.4778.

[26] Zimmermann, P. R. *The official PGP user's guide.* MIT Press, (1995).