UNIVERSIDADE FEDERAL FLUMINENSE

**ANDRÉ LUIZ NASSERALA PIRES**

# NEGAÇÃO DE SERVIÇO POR CONLUIO PRODUTOR-CONSUMIDOR EM REDES ORIENTADAS A CONTEÚDO

NITERÓI

2017

UNIVERSIDADE FEDERAL FLUMINENSE

**ANDRÉ LUIZ NASSERALA PIRES**

# NEGAÇÃO DE SERVIÇO POR CONLUIO PRODUTOR-CONSUMIDOR EM REDES ORIENTADAS A CONTEÚDO

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

Orientador:
Igor Monteiro Moraes

NITERÓI

2017

**Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF**

P667 Pires, André Luiz Nasserala
Negação de serviço por conluio produtor-consumidor em redes orientadas a conteúdo / André Luiz Nasserala Pires. – Niterói, RJ : [s.n.], 2017.
64 f.

Tese (Doutorado em Computação) - Universidade Federal Fluminense, 2017.
Orientadores: Igor Monteiro Moraes

1. Rede de computadores. 2. Sistema de computador. 3. Segurança da informação. I. Título.

CDD 004.6

ANDRÉ LUIZ NASSERALA PIRES

# NEGAÇÃO DE SERVIÇO POR CONLUIO PRODUTOR-CONSUMIDOR EM REDES ORIENTADAS A CONTEÚDO

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

Aprovada em julho de 2017.

BANCA EXAMINADORA

---

Prof. Igor Monteiro Moraes - Orientador, UFF

---

Prof. Célio Vinicius Neves de Albuquerque, UFF

---

Prof. Débora Christina Muchaluat Saade, UFF

---

Prof. Marcelo Gonçalves Rubinstein, UERJ

---

Prof. Luís Henrique Maciel Kosmalski Costa, UFRJ

Niterói

2017

*Dedico esse trabalho aos meus pais Assis e Nasserina, a minha esposa Camila, aos meus filhos Luiz André e Nassere José, e em especial ao meu avô Nassere Nasserala, cada letra desse trabalho tem um pouco de cada um.*

# Agradecimentos

Agradeço a todos que de qualquer maneira ajudaram no desenvolvimento desse trabalho. Em especial aos meus pais Assis Dantas e Nasserina Nasserala, ao meu avô Nassere Nasserala, que é meu principal motivador nessa tarefa. Agradeço aos professores Luís Henrique Maciel Kosmalski Costa, Célio Vinícius Neves de Albuquerque, Débora Christina Muchaluat Saade e Marcelo Gonçalves Rubinstein por terem aceitado participar dessa defesa de tese. Em especial agradeço ao meu orientador Professor Igor Monteiro Moraes pela paciência e dedicação. Por último, agradeço ao Dinter UFF/UFAC, Laboratório MídiaCom, CNPq e CAPES.

# Resumo

As Redes Orientadas a Conteúdo (ROC) são um novo paradigma de comunicação para a Internet. As ROCs propõem um novo modelo de comunicação que, diferentemente da arquitetura TCP/IP centrada na interconexão entre sistemas finais, busca recuperar o conteúdo requisitado por um usuário independente da localização física ou lógica desse conteúdo. Existem diversas propostas de arquiteturas para esse novo paradigma. Uma dessas propostas, e que vem tendo destaque na literatura é a arquitetura *Content Centric Networking* (CCN).

Uma das principais características da CCN é que tanto roteadores, quanto sistemas finais podem armazenar conteúdos recebidos em *cache* para aumentar sua disponibilidade. Essa característica torna a arquitetura mais robusta a ataques de negação de serviço (*Denial of Service* - DoS) tradicionais. Isso ocorre porque nós intermediários podem atender as solicitações de conteúdo sem que as solicitações cheguem diretamente à fonte do conteúdo. Então não existem garantias que o ataque afetará o nó vítima. Porém, com essa nova arquitetura novos tipos de ataques de negação de serviço surgiram explorando características presentes na CCN.

Essa tese propõe e avalia uma contramedida para comedir o ataque de negação de serviço por conluio produtor-consumidor na arquitetura CCN chamada de Cache nFace. Nesse ataque, consumidores e produtores maliciosos agem em conluio, gerando, disponibilizando e manipulando a popularidade de conteúdos. Assim, aumenta-se a probabilidade de um consumidor legítimo recuperar o conteúdo diretamente do produtor, aumentando o seu tempo de recuperação. A contramedida proposta comede este ataque, dividindo o *cache* de um nó em sub-*caches*. Cada sub-*cache* armazena somente os conteúdos solicitados através de uma interface de rede específica. Isso aumenta a robustez do *cache* sob ataque. Sem a contramedida, se um determinado nó encontra-se no caminho entre o consumidor e o produtor malicioso, seu *cache* inteiro é comprometido sob ataque. Isso ocorre porque o consumidor malicioso solicita dados produzidos pelo produtor malicioso numa taxa elevada ao ponto de remover conteúdos legítimos do *cache* dos nós intermediários. Porém, a probabilidade de todas as interfaces de rede de um nó receberem tráfego de ataque ao mesmo tempo é pequena. Essa é a premissa para o bom funcionamento do Cache nFace.

A análise é feita em duas topologias, uma em árvore com 32 nós, e outra em malha, com 192 nós. Variam-se a posição e a quantidade de atacantes para ambas topologias. As métricas utilizadas são tempo médio de recuperação de conteúdos legítimos, a ocupação maliciosa média do *cache* dos roteadores, a taxa média de erros de *cache* dos conteúdos legítimos e o percentual de conteúdos legítimos recuperados do produtor legítimo. Os resultados mostram que o cache nFace reduz em até 50% a eficiência do ataque e supera a outra proposta encontrada na literatura em todos os cenários analisados.

**Palavras-chave:** Internet do Futuro, CCN, Segurança, Negação de Serviço.

# Abstract

Information-Centric Networking (ICN) is a new communication paradigm to the Internet. Different from the TCP/IP architecture focused on the interconnection between end systems, ICN introduces a new communication model that aims at retrieving a content requested by a user regardeless of the logical or physical location of this content. There are several proposed architectures to this new paradigm. One of these proposals found in the literature is the Content Centric Networking (CCN).

One of the main features of the CCN is that both routers and end systems can store in cache previously received contents to increase its availability. In-network caching increases the CCN robustness against traditional denial of service attacks (DoS). With CCN, intermediate nodes can satisfy content requests before these messages reach the content source. Thus, there is no guarantee that the attack will impact the victim node. Nevertheless, new types of DoS attacks have emerged in order to explore several CCN features.

This thesis proposes and evaluates a countermeasure to mitigate the producer-consumer collusion attack in the CCN architecture called Cache nFace. This is a kind of DoS attack where malicious consumers and producers act in collusion, generating, publishing, and manipulating content popularity. The goal of the attack is to increase the probability of a legitimate consumer retrieves contents directly from the producer, increasing the content retrieval time. The proposed countermeasure mitigates this attack by dividing the cache of a node in sub-caches. Each sub-cache stores only the contents requested through a specific network interface. The goal is to increase the robustness of the cache under attack. Without the countermeasure, if a particular node is in the path between the consumer and the malicious producer, its entire cache is compromised under attack. In this case, the malicious consumer requests contents produced by the malicious producer at high rate in order to remove legitimate content from the cache of intermediate nodes. The probability that all network interfaces of a node are under attack simultaneously, however, is small. This is the key assumption of Cache nFace.

In the analysis, we consider two network topologies: a tree-topology with 32 nodes and a mesh-topology with 192 nodes. The position and the number of attackers are varied for both topologies. We consider the following metrics: the average retrieval time for legitimate contents, the malicious cache occupation of routers, the average cache miss rate for legitimate contents, and the percentage of legitimate contents retrieved from the legitimate producer. Results show that the Cache nFace reduces the attack efficiency by up to 50% and surpasses the other proposal found in the literature in all scenarios analyzed.

**Keywords:** Future Internet, CCN, Security, Denial of Service.

# Lista de Figuras


1.1 Cronologia da evolução das aplicações da Internet. . . . . 2

1.2 Mudança do paradigma de comunicação [22]. . . . . 3

2.1 Exemplo de recuperação de conteúdos WEB na Internet atual. . . . . 8

2.2 Exemplo de recuperação de conteúdos na CCN. . . . . 9

2.3 Funcionamento básico de um Nó CCN. . . . . 10

2.4 Exemplo de PIT. . . . . 11

2.5 Exemplo de FIB. . . . . 11

2.6 Processo de encaminhamento dos pacotes de interesse. . . . . 12

2.7 Processo de recuperação de conteúdos - dados. . . . . 12

2.8 Exemplo de CS. . . . . 13

2.9 Estrutura de um nome hierárquico. . . . . 14

3.1 Tipos de ataques em CCN [1]. . . . . 21

3.2 Exemplo de funcionamento do CacheShield. . . . . 29

4.1 O ataque em conluio produtor-consumidor: nós legítimos e maliciosos em ação. . . . . 31

4.2 Rede sob ataque em conluio produtor-consumidor. . . . . 32

5.1 Um exemplo de uso do mecanismo Cache nFace. . . . . 36

5.2 Cache nFace em ação. . . . . 37

6.1 Topologias utilizadas nas simulações. . . . . 41

6.2 O tempo de recuperação de conteúdos legítimos (LRU x LFU), árvore. . . 43

6.3 Percentual de carga do produtor (LRU x LFU), árvore. . . . . 44

6.4 Métricas em relação ao tamanho do CS, topologia em árvore. . . . . 45

6.5 O tempo de recuperação de conteúdos legítimos na topologia em árvore. . . 47

6.6 O tempo de recuperação de conteúdos legítimos na topologia em malha. . . 48

6.7 O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos para as contramedidas avaliadas na topologia em árvore. . . . . 50

6.8 O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos para as contramedidas avaliadas na topologia em malha. . . . . 51

6.9 A taxa de erros de *cache* para os conteúdos legítimos para as contramedidas avaliadas na topologia em árvore. . . . . 53

6.10 A taxa de erros de *cache* para os conteúdos legítimos para as contramedidas avaliadas na topologia em malha. . . . . 54

6.11 Percentual de carga do produtor para as contramedidas avaliadas na topologia em árvore. . . . . 55

6.12 Percentual de carga do produtor para as contramedidas avaliadas na topologia em malha. . . . . 56

# Lista de Abreviaturas e Siglas

| | | |
|---|---|---|
| ARPA | : | *Advanced Research Projects Agency*; |
| AS | : | *Autonomous System*; |
| CCN | : | *Content Centric Networking*; |
| CDN | : | *Content Distribution Network*; |
| CS | : | *Content Store*; |
| DNS | : | *Domain Name System*; |
| DoS | : | *Denial of Service*; |
| FIB | : | *Forwarding Information Base*; |
| GDSF | : | *Greedy Dual-Size Frequency*; |
| HTTP | : | *HyperText Transfer Protocol*; |
| ICN | : | *Information-Centric Networking*; |
| IP | : | *Internet Protocol*; |
| LFU | : | *Least Frequently Used*; |
| LRU | : | *Least Recently Used*; |
| MPC | : | *Most Popular Content*; |
| OSPF | : | *Open Shortest Path First*; |
| P2P | : | *Peer-to-Peer*; |
| PIT | : | *Pending Interest Table*; |
| ROC | : | Redes Orientadas a Conteúdo; |
| SVM | : | *Support Vector Machines*; |
| TCP | : | *Transmission Control Protocol*; |
| URI | : | *Uniform Resource Identifier*; |
| WWW | : | *World Wide Web*. |

# Sumário

**1 Introdução** **1**

1.1 Objetivos . . . . . 5

1.2 Organização do Texto . . . . . 6

**2 A Arquitetura CCN** **7**

2.1 Princípios de Funcionamento da CCN . . . . . 7

2.2 Funcionamento da CCN . . . . . 10

2.3 Nomeação de Conteúdos . . . . . 13

2.4 Roteamento e Encaminhamento de Conteúdos . . . . . 15

2.5 Armazenamento de Conteúdos . . . . . 16

**3 Segurança em CCN e Trabalhos Relacionados** **18**

3.1 Segurança em CCN . . . . . 18

3.2 DoS em CCN x DoS em Redes TCP/IP . . . . . 19

3.3 Novos Ataques em CCN . . . . . 20

3.3.1 Ataques de Nomeação . . . . . 21

3.3.2 Envenenamento do Cache . . . . . 22

3.3.3 Pedidos Aleatórios e Impopulares . . . . . 22

3.4 Negação de Serviço . . . . . 23

3.5 Trabalhos Relacionados . . . . . 24

3.6 O Mecanismo CacheShield . . . . . 27

**4 Ataque de Negação de Serviço por Conluio Consumidor-Produtor** **30**

4.1 Definição do Ataque . . . . . 30

**5 O Mecanismo Cache nFace** **34**

5.1 Funcionamento do Cache nFace . . . . . 34

5.2 Encaminhamento e Armazenamento com o Cache nFace . . . . . 35

**6 Avaliação do Mecanismo Cache nFace** **39**

6.1 Cenários de Avaliação . . . . . 39

6.2 Resultados . . . . . 42

6.2.1 Avaliação do Ataque . . . . . 42

6.2.1.1 Políticas de Descarte do Cache . . . . . 42

6.2.1.2 Tamanho do Cache . . . . . 44

6.2.2 Avaliação das Contramedidas . . . . . 46

6.2.2.1 Tempo Médio de Recuperação de Conteúdos Legítimos . . 46

6.2.2.2 Ocupação Maliciosa dos Caches e Taxa de Erro do Cache . 49

6.2.2.3 Conteúdos Recuperados do Produtor Legítimo . . . . . 52

**7 Conclusão** **58**

7.1 Resultados Obtidos . . . . . 58

7.2 Limitações da Proposta . . . . . 59

7.3 Contribuições . . . . . 59

7.4 Trabalhos Futuros . . . . . 60

**Referências** **61**

# Capítulo 1

# Introdução

A Internet é uma rede que começou a ser projetada na década de 1960 como um projeto da ARPA (*Advanced Research Projects Agency*), um órgão ligado ao Departamento de Defesa dos EUA. No início, o objetivo era desenvolver uma rede que se mantivesse funcionando mesmo que alguns de seus pontos fossem destruídos por algum tipo de ataque militar, e as principais aplicações eram o compartilhamento de recursos de *hardware*, como impressoras, arquivos e a troca de mensagens eletrônicas (e-mails). Desde a criação até os dias atuais, a Internet segue o princípio fim-a-fim que deixa a inteligência da rede com a aplicação usada, tornado o núcleo da rede simples no sentido de apenas encaminhar pacotes.

Com a popularização do acesso à Internet e com o advento da *World Wide Web* (WWW), no início dos anos 90, o conteúdo multimídia passou a ser consumido por usuários domésticos com intensidade maior, assim redes de compartilhamento de arquivos baseadas em serviços par-a-par (*peer-to-peer* - P2P) como Emule ou Kazaa, passaram a ser usadas com maior frequência. Deste modo, observa-se que a distribuição de conteúdo na Internet atravessou um processo evolutivo, afastando-se da definição de sistema de informação textual em direção à definição de um sistema de informação multimídia, no qual dados, serviços e aplicações são consumidos como conteúdos [34, 8]. De acordo com um levantamento da Sandvine[1] em dezembro de 2015, somente o Netflix[2], empresa que vende acesso a conteúdo multimídia, é responsável por 35,2% do tráfego da Internet na América do Norte. No total do continente, 71% do tráfego da rede é constituído por pessoas assistindo a vídeos em *streaming*. Já na América Latina, o Netflix não opera em todos os países, na contagem geral, ele representa apenas 6,6% do consumo da Internet re-

[1] http://www.sandvine.com/
[2] http://www.netflix.com/

sidencial. Ainda de acordo com o mesmo instituto, no Brasil o líder de geração de tráfego é o YouTube[3], com 36,8%. Na América do Norte, o YouTube é o segundo colocado, com cerca de 18,7%. A Figura 1.1 mostra a mudança do interesse nos conteúdos, de aplicações simples do início da Internet, para as necessidades atuais com aplicações mais complexas, destacando a WWW como foco para início de consumo multimídia em massa na rede [22].

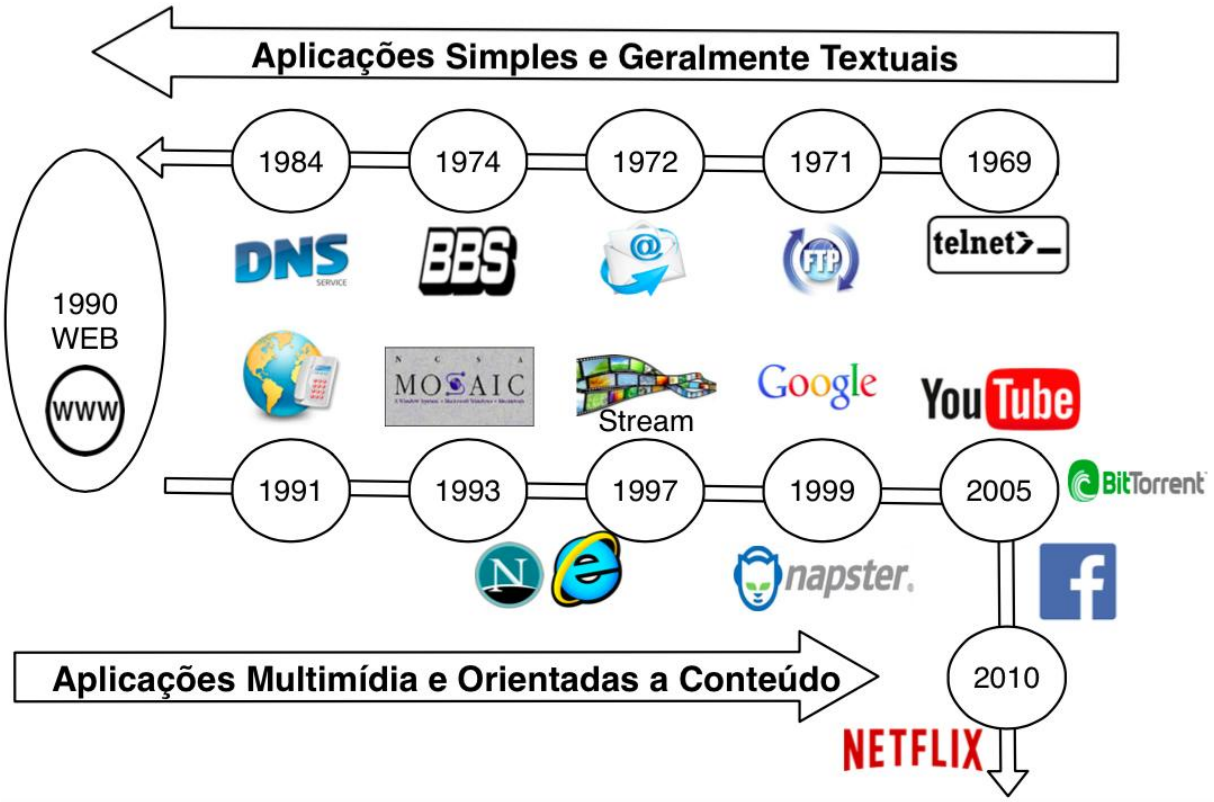



Figura 1.1: Cronologia da evolução das aplicações da Internet.

Apesar da mudança no tipo de conteúdo consumido pelos usuários, não houve grandes mudanças no núcleo da rede, ou seja, a rede ainda atende as solicitações de consumo de conteúdo baseada no princípio fim-a-fim. Algumas técnicas que utilizam P2P ou as CDNs (*Content Distribution Networks*), atendem parcialmente essa necessidade. Porém, a arquitetura atual da Internet remete a problemas de persistência, disponibilidade e segurança dos conteúdos, uma vez que tais aplicações fazem uso de soluções específicas e/ou proprietárias [8]. Na arquitetura vigente o tempo total de entrega do conteúdo é aumentado, pois existem, por exemplo, redirecionamentos HTTP (*HyperText Transfer Protocol*) e DNS (*Domain Name System*) dinâmico em CDNs, onde não é garantida a persistência das informações [22, 23]. Como toda comunicação acontece entre sistemas finais, o modelo de segurança IP tenta assegurar o canal de comunicação e não o conteúdo

[3]http://www.youtube.com

em si. Portanto, faz-se necessária uma mudança estrutural no núcleo de rede, isso remete a uma nova arquitetura de Internet.

As Redes Orientadas a Conteúdo são um novo paradigma de comunicação para Internet [8]. O principal objetivo dessas redes é a entrega de conteúdo para os usuários independentemente da localização desse conteúdo, ao contrário da arquitetura TCP/IP, cujo objetivo é a comunicação entre sistemas finais. Diversas arquiteturas foram propostas para esse novo paradigma de comunicação e uma das arquiteturas com maior destaque na literatura é a *Content Centric Networking* (CCN) [22, 38]. Entre as principais características da CCN estão o roteamento através de nomes de conteúdo, o armazenamento de conteúdo em nós intermediários da rede e a capacidade de auto-certificar o conteúdo, aplicando a segurança diretamente aos pacotes de dados [22].

Uma diferença importante entre a Internet atual e a CCN pode ser observada na Figura 1.2. Observa-se que a divisão entre *Links* Individuais e Aplicações individuais está centrada nos pedaços de conteúdo na CCN, enquanto na Internet atual todos os nós devem utilizar IP. Essa característica destaca a mudança do foco entre sistemas finais (IP) para o conteúdo, e ilustra o paradigma de orientação a conteúdos (pedaço do conteúdo) adotado pela CCN.

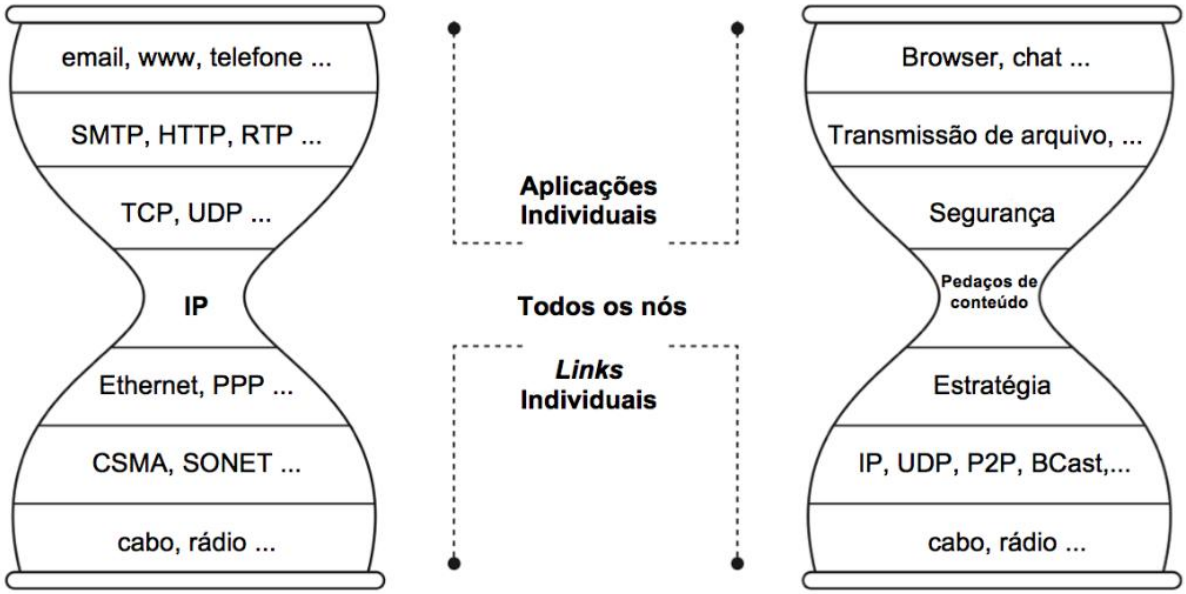



Figura 1.2: Mudança do paradigma de comunicação [22].

Uma das mais vantajosas características da CCN é o consumo indireto de conteúdo, ou seja, qualquer nó da rede pode armazenar conteúdos em *cache* e ao receber uma solicitação de conteúdo pode enviar tal conteúdo para o nó solicitante caso possua esse conteúdo em

*cache*. Na CCN, o nó que solicita o conteúdo é chamado de consumidor e o nó que é a fonte do conteúdo na rede é chamado de produtor. Quaisquer nós intermediários entre o consumidor e o produtor, chamados de roteadores de conteúdo, podem disponibilizar o conteúdo solicitado. Portanto, na CCN, é possível que um determinado nó, que esteja mais próximo do consumidor, consiga responder à solicitação de um conteúdo sem que o consumidor seja obrigado a recuperar esse conteúdo diretamente do produtor, que pode estar mais distante. Assim, o tempo de recuperação de conteúdos pode ser reduzido. Além disso, o armazenamento de conteúdo em *cache* aumenta a disponibilidade de conteúdos e pode reduzir o consumo de banda, uma vez que o conteúdo é encaminhado por menos saltos.

Outra característica da CCN é que a segurança é aplicada diretamente aos conteúdos, diferentemente da arquitetura TCP/IP, na qual a segurança é aplicada ao canal de comunicação entre os sistemas finais [38]. Um pacote de dados CCN é auto-certificado, isto é, ele contém a assinatura digital do conteúdo e do seu nome e a chave pública do produtor, ou informações para obtê-la [22]. Portanto, é possível verificar a integridade do pacote e se ele foi gerado pelo produtor que possui tal chave pública. Além disso, a CCN é mais robusta a ataques de negação de serviço (*Denial of Service* - DoS) comuns na Internet atual, como o de esgotamento de banda e o de reflexão, em virtude do uso de *cache* pelos nós intermediários e da agregação de solicitações de conteúdo [17]. Na CCN, como será visto na Seção 3.1, não há garantias de que um pacote seja encaminhado até um dado nó vítima.

Um ataque de negação de serviço particular, chamado de conluio produtor-consumidor, entretanto, pode ser efetivo porque os mecanismos nativos empregados pela CCN não são suficientes para inibi-lo [31, 33, 30]. Nesse ataque, consumidores maliciosos solicitam conteúdos que são disponibilizados apenas por produtores maliciosos a uma alta taxa. Isso aumenta o tempo de recuperação de conteúdos legítimos, em virtude do aumento da taxa de erro do *cache* (*cache miss*) para esses conteúdos e, consequentemente, da necessidade de nós legítimos terem que recuperar o conteúdo diretamente do seu produtor. Os mecanismos de segurança da CCN padrão são ineficazes na detecção do ataque em conluio, pois, do ponto de vista da rede, as solicitações e os conteúdos são legítimos. São enviados pacotes de interesse para conteúdos que existem e que são disponibilizados por produtores. O conteúdo é malicioso porque torna popular um conteúdo que não é de interesse de usuários legítimos. Como o produtor malicioso assina os conteúdos de acordo com a política definida pela CCN, os consumidores maliciosos podem solicitá-los sem risco de que esses conteúdos sejam descartados por mecanismos de verificação de assinaturas e

chaves.

Esse ataque é possível, porque a CCN emprega por padrão uma política de substituição do *cache* baseada, na popularidade dos conteúdos. Assim, se um determinado conteúdo não foi solicitado recentemente pelos consumidores, ele é considerado menos popular. Dessa forma, esse conteúdo terá prioridade de descarte quando houver necessidade de armazenar novos conteúdos. Vários nós maliciosos podem, então, solicitar um conjunto específico de conteúdos produzidos maliciosamente, e em taxas altas de envio de interesse para manipular a política de *cache*. Assim, dependendo da forma como os conteúdos maliciosos são solicitados, é possível até remover conteúdos legítimos do *cache*.

## 1.1 Objetivos

O objetivo desse trabalho é propor uma contramedida eficaz ao ataque de conluio produtor-consumidor. O primeiro passo, no entanto, é avaliar o ataque em diferentes cenários para entender melhor seu comportamento, bem como entender como as políticas de descarte do *cache* podem funcionar durante o ataque. Em seguida compara-se o mecanismo proposto mediante simulações, e em diferentes configurações e cenários com outro mecanismo conhecido na literatura como CacheShield [41].

A avaliação do ataque de conluio produtor-consumidor é feita através de simulações para diferentes configurações, nas quais se variam o número de consumidores e produtores em conluio, a taxa de pacotes de interesse e o padrão de solicitações de conteúdos maliciosos. As métricas empregadas são o tempo de recuperação de conteúdos legítimos, o percentual de ocupação maliciosa do *cache*, o percentual da taxa de erros de *cache* de conteúdos legítimos e o percentual de conteúdos legítimos recuperados do produtor legítimo. Os resultados mostram que o ataque compromete uma das maiores vantagens das CCN que é a redução do tempo de recuperação de conteúdos pelo uso do *cache* nos nós intermediários [32].

A princípio, observou-se nas simulações que as políticas de descarte baseadas no último uso LRU (*Least Recently Used*) são menos eficazes que as que observam a frequência do uso LFU (*Least Frequently Used*) na tentativa de reduzir o dano do ataque a rede.

O mecanismo proposto chama-se Cache nFace e consiste basicamente em dividir o *cache* de um nó em sub-*caches*. Cada um desses sub-*caches* é associado a uma interface de rede de um nó e o espaço de armazenamento de cada sub-*cache* é definido de acordo com taxa de transmissão da interface associada. Dessa forma, um sub-*cache* $s_i$ só pode

armazenar conteúdos, cujos pacotes de interesse tenham sido recebidos pela interface $i$. Isso aumenta a robustez do *cache* sob ataque devido à probabilidade de todas as interfaces serem atacadas ao mesmo tempo ser pequena. Caso exista apenas uma interface fora do alcance do ataque, essa pode continuar respondendo a interesses legítimos e aumentar a taxa de acerto do *cache*.

São usadas duas topologias de rede na avaliação do mecanismo proposto, uma em árvore de 32 nós e uma em malha de 192 nós. As métricas utilizadas são o tempo médio de recuperação de conteúdos legítimos, a ocupação maliciosa média do *cache* dos roteadores, a taxa média de erros de *cache* dos conteúdos legítimos e o percentual de conteúdos legítimos recuperados do produtor legítimo. Ainda é objetivo do trabalho comparar a proposta com o mecanismo CacheShield, proposto e analisado por Xie *et al.* [41]. Os resultados mostram que o mecanismo proposto é mais eficiente em comedir o ataque nas métricas avaliadas que o CacheShield em todos os cenários avaliados.

## 1.2 Organização do Texto

O trabalho está organizado da seguinte forma. No Capítulo 2 são apresentadas as características principais da CCN. No Capítulo 3 são apresentados aspectos de segurança e os trabalhos relacionados. Em seguida, no Capítulo 4, é apresentado o ataque de negação de serviço por conluio produtor-consumidor. No Capítulo 5, o mecanismo Cache nFace é descrito. No Capítulo 6 são definidos os cenários de avaliação e os resultados dos experimentos são analisados e discutidos. Por fim, no Capítulo 7 as conclusões são apresentadas.

# Capítulo 2

# A Arquitetura CCN

A CCN propõe conceitos inovadores. Como exemplo pode-se destacar um novo tipo de encaminhamento baseado em nomes de conteúdos. Esse novo encaminhamento contrasta com o modelo atual da Internet, no qual o encaminhamento é baseado no protocolo IP, cujo foco de decisão é o endereço de destino e não o conteúdo solicitado à rede. Isso acontece porque na arquitetura TCP/IP cabe à camada de rede tomar a decisão de roteamento, enquanto os conteúdos, são tratados somente nas camadas superiores dos modelos de referência. Isso deixa a inteligência com a borda da rede. Na cultura atual da Internet o interesse do usuário passa a ser a mídia buscada (som, imagem, vídeo, etc.), e é justamente isso que a CCN propõe: um foco para decisão de encaminhamento e roteamento no nome do conteúdo e não no endereço IP de destino. Esse novo modelo enfatiza o interesse pelo conteúdo, independente de sua localização física ou lógica. Para que seja efetivamente utilizada, a CCN exige mudanças físicas e lógicas nos roteadores que compõem a Internet. Essas mudanças permitem que roteadores respondam a requisições de conteúdo utilizando dados armazenado no seu *cache*, além de garantir uma resposta válida e íntegra da informação solicitada.

## 2.1 Princípios de Funcionamento da CCN

A Figura 2.1 mostra o processo de recuperação de um conteúdo WEB em uma rede TCP/IP. O Cliente solicita o conteúdo www.ufac.br e, então o DNS resolve o nome para o endereço IP 200.179.173.130. O *gateway* (R1) do Cliente é acionado, e através de sua tabela de roteamento utiliza a interface IF1. Os próximos saltos, R2 e R3, utilizam respectivamente as interfaces IF2 e IF3 para reencaminharem os pacotes e finalmente chegarem ao Servidor www.ufac.br. Na camada de transporte, é estabelecida uma conexão

TCP entre o programa do Cliente e o Servidor. O Cliente envia uma requisição HTTP para sua interface *socket*. O TCP transposrta essa mensagem para a interface *socket* do Servidor. O Servidor envia uma resposta HTTP para sua interface *socket*. Finalmente o TCP leva essa resposta para a interface *socket* do Cliente. Durante esse processo nenhum nó intermediário (R1, R2 e R3) pode fornecer os objetos solicitados pelos Cliente. O Cliente recupera todos os dados diretamente do Servidor.

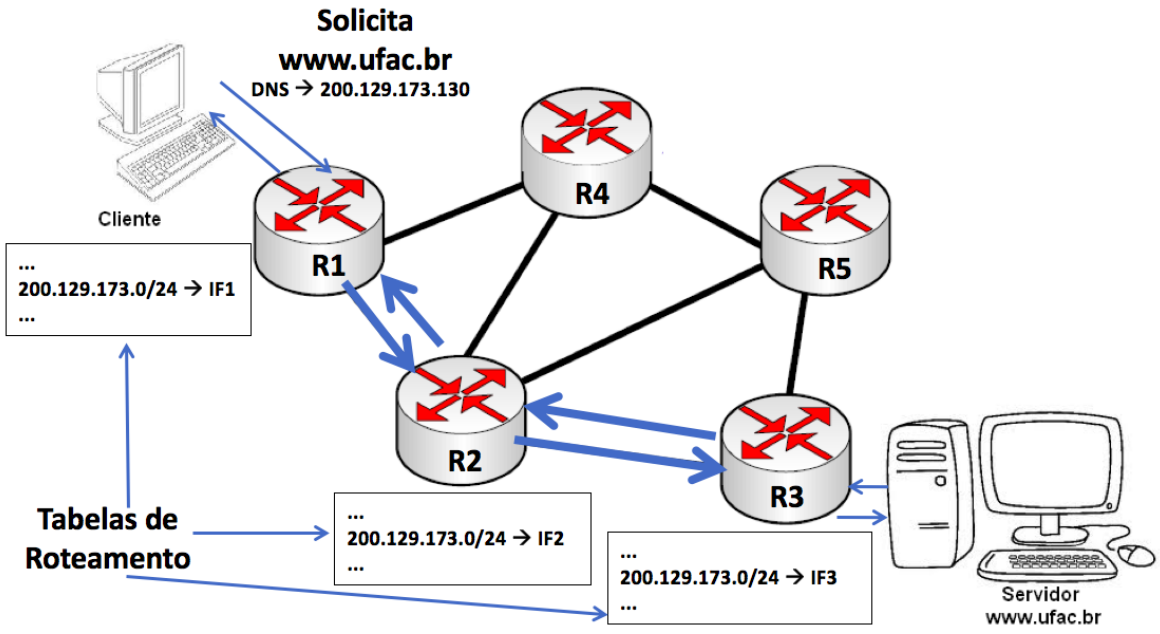



Figura 2.1: Exemplo de recuperação de conteúdos WEB na Internet atual.

No exemplo da rede TCP/IP (Figura 2.1) o conteúdo é sempre recuperado diretamente do servidor, pois não há emprego de *cache* pelos roteadores de núcleo. O caso do uso de *cache* em uma rede TCP/IP remete aos HTTP *proxies*, mas nesse caso não se trata de uma característica nativa da arquitetura, e sim a adição de um serviço opcional [26]. Na Figura 2.2 é ilustrado o mesmo processo de recuperação na rede CCN. O Cliente solicita um conteúdo www.ufac.br através de um pacote de interesse. Ao chegar no próximo salto, o roteador R1 verifica se www.ufac.br está em *cache*, caso positivo responde com pacote de dados. Não estando disponível em R1, o próximo passo é verificar se o conteúdo está pendente na PIT (*Pending Interest Table*). Caso positivo, R1 registra a IF1 como interface pendente para o conteúdo e aguarda. Caso o conteúdo não esteja na PIT, R1 verifica na FIB (*Forwarding Information Base*) quais interfaces correspondentes ao prefixo do conteúdo solicitado e encaminham pela(as) interface(es) um pacote de interesse. Caso não exista correspondência na FIB o interesse é descartado. No exemplo, R1 encaminha o interesse aos nós vizinhos R2 e R4 que repetem o processo. Quando um nó, no exemplo R4, tem o conteúdo, ele responde ao interesse com o pacote de dados

equivalente. Chegando em R1, esse pacote de dados é armazenado em *cache*, excluído da PIT e encaminhado pela(as) interface(es) do interesse pendente, chegando finalmente ao Cliente. Na CCN não existe a obrigação de que o processo de recuperação de conteúdos seja feito diretamente do Servidor www.ufac.br, qualquer nó intermediário (R4 no exemplo) pode atender as solicitações do Cliente, desde que tenha o conteúdo no seu *cache*. Apesar de exigir maior armazenamento nos nós, isso melhora a forma como os conteúdos são consumidos e disponibilizados na rede, trazendo diversas melhorias à Internet.

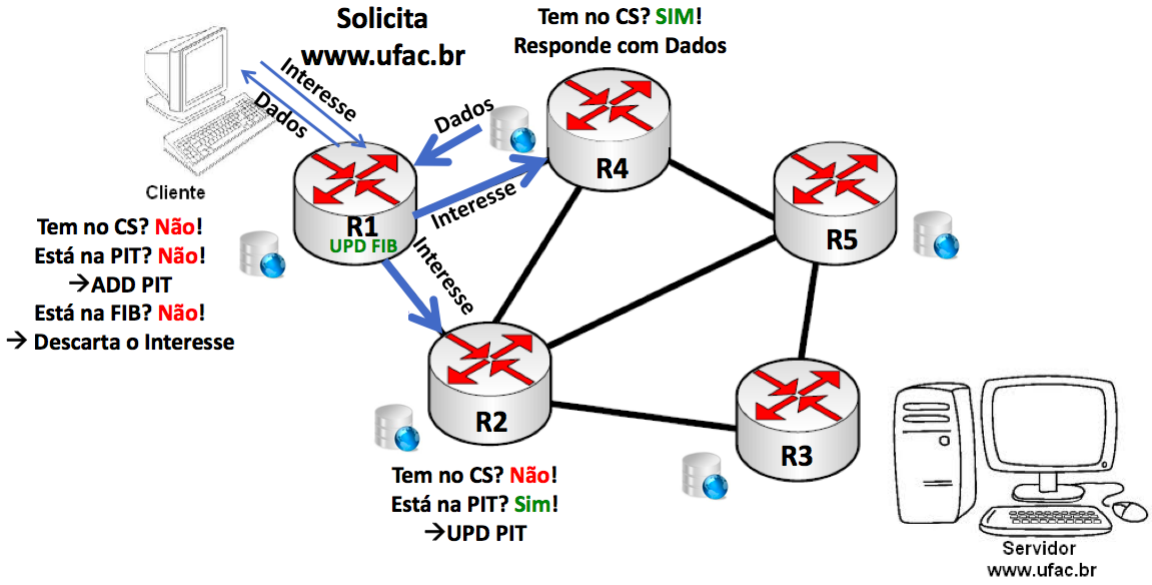



Figura 2.2: Exemplo de recuperação de conteúdos na CCN.

A arquitetura CCN tem como objetivos aumentar a disponibilidade e reduzir o tempo de recuperação de conteúdos. Na CCN, os nós da rede possuem um *cache* para armazenar conteúdos recebidos previamente. Consequentemente, qualquer nó pode responder a um pedido, se o conteúdo solicitado está disponível em seu *cache*, conhecido como *Content Store* (CS). Quanto mais nós armazenam um conteúdo na rede, maior a disponibilidade desse conteúdo e maior a probabilidade de consumidores recuperarem um conteúdo de um nó mais próximo. Essa é uma das vantagens da CCN em comparação com a arquitetura atual da Internet. Durante as seções seguintes são detalhados o funcionamento da CCN e seus principais elementos. Ao contrário dos *buffers* de pacotes em um roteador IP, o CS em um roteador CCN é essencialmente um *cache* [41]. Seu objetivo é maximizar o compartilhamento de objetos de conteúdo popular entre diferentes usuários e torná-lo o mais próximo possível das fontes dos pedidos.

## 2.2 Funcionamento da CCN

A CCN emprega dois tipos de pacotes: interesse e dados [22, 13]. Consumidores enviam pacotes de interesse para solicitar um conteúdo. Os produtores ou roteadores de conteúdo respondem aos interesses com pacotes de dados, que carregam o conteúdo em si ou pedaços de conteúdo, chamados de *chunks* [9]. Os nós encaminham tanto pacotes de interesse quanto pacotes de dados com base no próprio nome do conteúdo, ao invés do endereço de destino do nó que possui o conteúdo. Para realizar o encaminhamento de pacotes, cada nó CCN tem duas estruturas de dados: a *Pending Interest Table* (PIT) e a *Forwarding Information Base* (FIB). A Figura 2.3 mostra o princípio de funcionamento da CCN. Um nó recebe um pacote de interesse e responde com o pacote de dados correspondente, ou encaminha o interesse se não tiver o dado em seu *cache*.

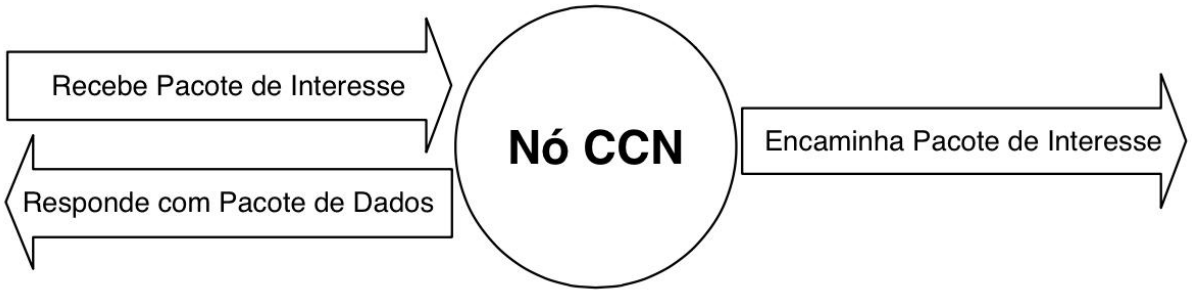



Figura 2.3: Funcionamento básico de um Nó CCN.

Como pode ser observado na Figura 2.4, a PIT guarda o estado de cada pacote de interesse encaminhado por um nó que ainda não recebeu uma resposta, ou seja, os interesses que esperam por um pacote de dados. Cada entrada da PIT também armazena as interfaces de recepção de um pacote de interesse. É importante ressaltar que o tamanho da PIT é limitado e, dessa forma, novos interesses que chegam enquanto a tabela está cheia não são encaminhados. Esse fato é explorado por usuários maliciosos, como detalhado no Capítulo 3. No exemplo os conteúdos br.uff/Musicas/GetLucky.mp3/v3/s1 e br.ufac/www/index.html/v1/s4 tiveram interfaces de chegada 2 e 4, e 1 e 3, respectivamente. Desde modo quando o pacote de dados equivalente aos conteúdos ingressar no nó eles serão encaminhados conforme o interesse pendente na PIT.

A FIB atua como uma tabela de encaminhamento para pacotes de interesse. Essa tabela contém uma lista de entradas, cada uma contendo o prefixo de um nome e uma lista de interfaces de saída para as quais os pacotes de interesse com nomes de mesmo prefixo devem ser encaminhados. Na Figura 2.5 tem-se um exemplo de FIB. Nesse exemplo se houver a necessidade de recuperar conteúdos cujo prefixo seja br.uff os pacotes de interesse

| PIT | |
|---|---|
| Nome do Pedaço de Conteúdo | Interfaces de Chegada |
| br.uff/Musicas/GetLucky.mp3/v3/s1 | 2,4 |
| br.ufac/www/index.html/v1/s4 | 1,3 |
| ... | ... |
| Prefixo_n/conteúdo/vn/sn | 1,2,3,... |

Figura 2.4: Exemplo de PIT.

para tais conteúdos são encaminhados pelas Interfaces 1 e 3. Caso o prefixo seja br.ufac, os pacotes de interesse são encaminhados pelas Interfaces 2 e 4, se o prefixo não possuir uma entrada correspondente na FIB o pacote de interesse é descartado.

| FIB | |
|---|---|
| Nome do Prefixo | Interfaces |
| br.uff | 1,3 |
| br.ufac | 2,4 |
| ... | ... |
| Prefixo n | 1,2,3,... |

Figura 2.5: Exemplo de FIB.

Quando um nó CCN recebe um pacote de interesse, ele verifica seu CS para encontrar uma cópia do conteúdo solicitado, cujo nome está no cabeçalho do pacote de interesse. Se o conteúdo está armazenado em *cache*, o nó envia um pacote de dados para o consumidor. Caso contrário, o nó verifica a sua PIT. Se houver uma entrada na PIT para o mesmo conteúdo, o nó atualiza a lista de interfaces de entrada e descarta o pacote de interesse. Esse procedimento é chamado de agregação de pacotes de interesse e torna a CCN mais robusta contra ataques de DoS, como discutido no Capítulo 3. Caso contrário, consulta a FIB para determinar a interface de saída para encaminhar o pacote de interesse, e então o nó cria uma nova entrada na PIT. Se não houver nenhuma entrada na FIB relacionada com o nome do conteúdo, o pacote de interesse é descartado. Os nós repetem este processo de encaminhamento para cada pacote de interesse recebido. A Figura 2.6 apresenta o fluxo gerado em um nó CCN quando um pacote de interesse é recebido.

Os pacotes de dados seguem o caminho reverso percorrido pelos pacotes de interesse porque a PIT armazena a lista de interfaces com interesses pendentes a serem atendi-

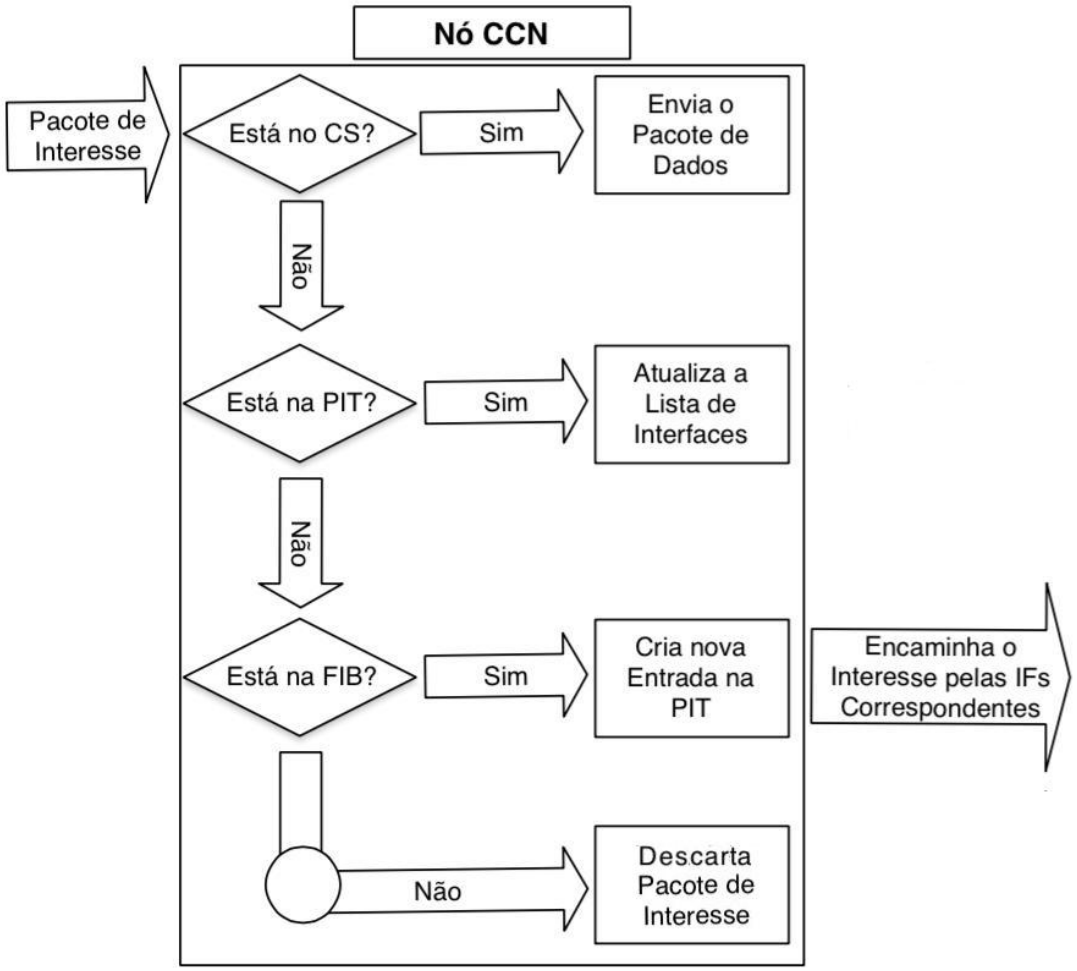



Figura 2.6: Processo de encaminhamento dos pacotes de interesse.

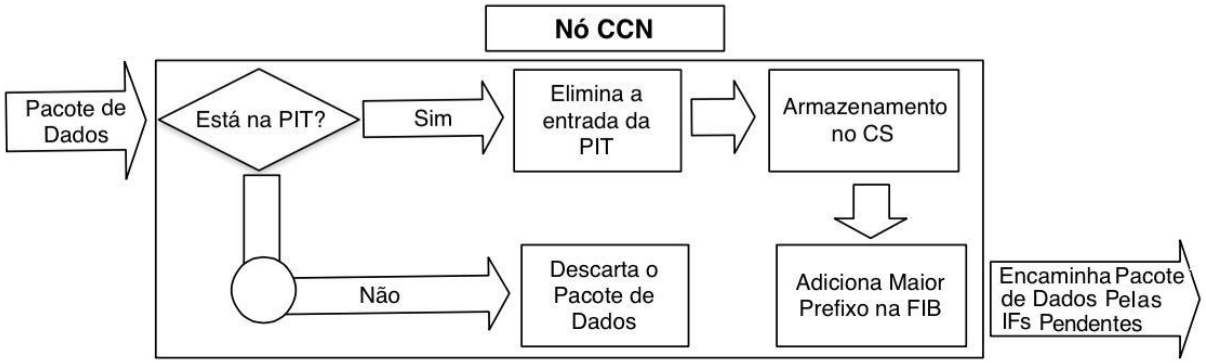



Figura 2.7: Processo de recuperação de conteúdos - dados.

dos [42]. A Figura 2.7 apresenta o fluxo gerado em um nó CCN quando recebe um pacote de dados. Um pacote de dados sempre vem em resposta a um pacote de interesse, portanto quando um pacote de dados ingressa em um nó CCN é verificado se existe entrada na PIT. Caso não exista, e isso pode ocorrer, por exemplo, por *timeout* do interesse, ele

será descartado. Se existir, a entrada na PIT correspondente ao interesse associado ao dado será eliminada e ele será enviado pela(as) interface(es) que aguardam a resposta. Em seguida o pacote de dados segue para o CS, que decidirá com base nas políticas de armazenamento se deve por em *cache* ou não o dado. A Figura 2.8 ilustra o CS de um nó CCN. Nessa estrutura são armazenados o nome do pedaço de conteúdo, como $br.uff/Musicas/GetLucky.mp3/v3/s1$ e $br.ufac/www/index.html/v1/s4$ e o dado associado a esse pedaço de conteúdo de forma criptografada.

| CS | |
|---|---|
| Nome do Pedaço de Conteúdo | Dado |
| br.uff/Musicas/GetLucky.mp3/v3/s1 | ... |
| br.ufac/www/index.html/v1/s4 | ... |
| ... | ... |
| Prefixo_n/conteúdo/vn/sn | ... |

Figura 2.8: Exemplo de CS.

Finalmente o maior prefixo correspondente ao nome do conteúdo e a interface são adicionados na FIB como política para futuros encaminhamentos de interesses.

## 2.3 Nomeação de Conteúdos

O processo de nomeação de conteúdo na CCN difere do sistema hierárquico de nomes e domínios DNS (*Domain Name System*). O DNS serve para mapear os nomes de estações para seus respectivos endereços de rede, ou seja, para associar um nome a um endereço IP (resolução de nomes). Esse processo também pode ser reverso, ou seja, resolver o IP em um nome, nesse caso chamado de resolução reversa. Na CCN não existe a necessidade de resolver um endereço num nome. Isso deve-se ao fato do próprio nome já ter as características necessárias para os processos de recuperação e localização do conteúdo. Quando fala-se em processo de nomeação de conteúdo, busca-se uma técnica para garantir a persistência, escalabilidade e segurança de conteúdos compartilhados na rede. Esse processo de nomeação deve estar associado a saber quais características que tornam possível identificar o produtor e o tipo de arquivo disponibilizado. Para tal, são utilizados esquemas de nomeação que permitem identificar o conteúdo e requisitar sua distribuição à infraestrutura de rede de forma eficiente, segura e com alta disponibilidade [8].

A CCN emprega por padrão uma técnica de nomeação conhecida como hierárquica [22]. Esse tipo de nomeação funciona através da concatenação de diferentes componentes hierárquicos de nome. Deste modo, identificadores únicos podem ser formados para atribuição a conteúdos. Os nomes hierárquicos possuem uma característica semântica, uma vez que suas estruturas e cada um dos componentes que os compõem refletem alguma informação a respeito da natureza do conteúdo: propriedade, versão, formato etc. Dessa forma, estruturas semelhantes a identificadores uniformes de recursos (*Uniform Resource Identifiers* - URIs) [28] podem ser utilizadas na representação de nomes hierárquicos [8].

A Figura 2.9 mostra a estrutura de um nome hierárquico que é adotado na CCN, no qual um usuário pode solicitar, por exemplo, o conteúdo Musicas/GetLucky.mp3 do publicador br.uff e receber um pedaço de conteúdo, específico, com a estrutura: br.uff/Musicas/GetLucky.mp3/1/1. Através do vetor *timestamp*, que é utilizado para versionamento do conteúdo, num momento posterior e utilizando o pedaço recebido, pode-se solicitar os outros pedaços desse conteúdo, do mesmo modo como foi efetuado com o primeiro recebido e na mesma versão (1 do *timestamp*). Essa segunda parte poderia ser, por exemplo, br.uff/Musicas/GetLucky.mp3/1/2, onde 1 representa a mesma versão do pedaço anterior, e 2 o número de sequência desse pedaço.

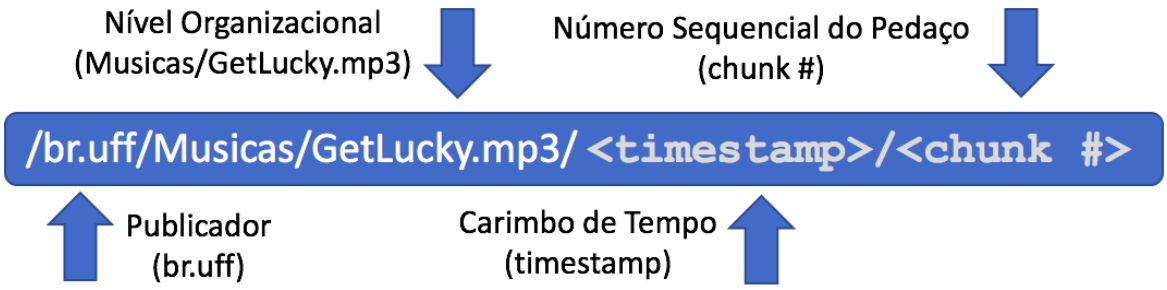



Figura 2.9: Estrutura de um nome hierárquico.

A nomeação hierárquica tem a vantagem de poder agregar nomes através de mapeamento de prefixos. Isso torna possível para um roteador CCN indicar que a recuperação de um conteúdo se dará por uma interface específica, visto que outros conteúdos de mesmo prefixo foram recuperados por essa interface. Isso é algo similar à agregação de rotas realizada pelos protocolos de roteamento da arquitetura TCP/IP.

A abordagem hierárquica para nomeação de conteúdos, por outro lado, pode trazer problemas no processo de recuperação de conteúdos. Uma distribuição de conteúdo segura requer que qualquer receptor seja capaz, de forma confiável, de avaliar três propriedades de cada parte do conteúdo recebido [38]:

- Sua integridade: se é uma cópia completa e não corrompida do conteúdo que o produtor gerou;
- Sua procedência: se o consumidor está disposto a confiar no conteúdo do produtor;
- Sua relevância: se o conteúdo é uma resposta para a pergunta do consumidor.

A melhor maneira de nomear os conteúdos de modo a garantir sua integridade, procedência e relevância ainda é alvo de grande debate na literatura [37]. No processo de nomeação, o conteúdo é assinado pelo publicador, mas existe um rótulo que apesar de escolhido pelo produtor, não é assinado. Esse rotulo é uma *string* legível por seres humanos e única dentro do conjunto de conteúdos publicados pelo publicador [4]. Deste modo, qualquer pedaço de conteúdo assinado por um publicador pode ser indetectável e substituído por qualquer outro com um rótulo diferente. Um usuário mal intencionado pode associar um novo rótulo de sua escolha com qualquer conteúdo publicado pelo publicador. Como de forma nativa somente o consumidor fará a verificação do conteúdo, se não houver roteadores verificando a integridade do pacote no caminho até o consumidor que solicitou, esse pacote pode ser adicionado num *cache* de um nó intermediário, poluindo esse *cache*. Isso será discutido no Capítulo 3. Apesar dos envolvidos diretamente no projeto da arquitetura CCN adotarem o modelo de nomeação hierárquico e afirmarem que este é o mais seguro [22, 38], outros argumentam a favor da utilização de nomes auto-certificados [18, 5].

## 2.4 Roteamento e Encaminhamento de Conteúdos

O roteamento baseado em nomes de conteúdos deve ser capaz de encontrar, independente da localização, conteúdos requisitados por um nó CCN. Para esse fim os nós CCN detêm informações sobre os dados armazenados em seu *cache* e informações sobre interesses pendentes de outros nós. Somente com essa informações os nós são capazes de construir uma tabela de informações de encaminhamento através da qual pode-se saber por qual interface de rede deve-se enviar o interesse do nó sendo que esse ainda não está disponível no *cache* local.

Para tal, os nós da CCN necessitam obter informações a respeito dos conteúdos existentes na rede a fim de encaminhar, da melhor forma possível, as requisições de conteúdo até cópias dos conteúdos requisitados. Evita-se assim o envio de um interesse por uma interface que não atenderá, provavelmente, tal solicitação.

O roteamento baseado em nomes de conteúdos deve apresentar características que garantam sua eficácia e eficiência, ou seja, deve ser capaz de entregar o conteúdo sem endereços de origem e destino, apenas com a informação da interface que solicitou. Além disso, rotas a base de nome devem ser capazes de se recuperar de falhas para evitar encaminhamento de dados a nós falhos em situações de descontinuidade. Outra característica importante é que numa rede CCN a eficiência de utilização dos *links* gera um baixo impacto na quantidade de tráfego. Por fim, esse tipo de roteamento deve ser capaz de realizar suas atividades em diferentes tipos de topologias, sendo essas de pequena ou larga escala.

Geralmente as técnicas de encaminhamento aplicados na CCN constroem tabelas baseadas no OSPF (*Open Shortest Path First*). O OSPF inunda toda a rede com atualizações de prefixos não agregados, atualizando assim a FIB dos nós da redes [40]. De forma semelhante a Internet atual, o uso de roteamento de caminho mais curto baseado no número de saltos ou nas métricas de nível de *link* entre um roteador e um produtor de conteúdo é considerado em CCN. No entanto, como podem exisatir roteadores de conteúdo no caminho entre o cosumidor e produtor, o roteamento de caminho mais curto pode não ser sempre otimizado [29]. Na CCN a estrutura responsável pelo encaminhamento é a FIB.

## 2.5 Armazenamento de Conteúdos

Com base na característica de acesso a conteúdos na Internet, na qual uma pequena parcela de conteúdos populares contribui com a maior parte do tráfego na rede [7], o espalhamento dos conteúdos e a disponibilização dos conteúdos em nós mais próximos dos consumidores pode reduzir o tráfego na rede e reduzir o tempo de recuperação de conteúdos.

Na CCN cabe ao nó, com políticas locais, a decisão de armazenar ou não o dado no seu *cache*. No caso do *cache* estar cheio, a política de descarte ou substituição utilizada entra em ação para determinar qual conteúdo novo substituirá o descartado. A política de adição padrão da CCN é armazenar em *cache* todo novo conteúdo recebido e que já não esteja no *cache*. Das várias políticas de descarte de *cache* existentes, destacam-se duas: uma baseada no conteúdo que está há mais tempo sem uso, chamada de LRU (*Least Recently Used*), e outra que descarta o conteúdo usado com menos frequência LFU (*Least Frequently Used*). O padrão da CCN é o LRU [3].

O armazenamento de dados feito pela CCN difere da forma adotada pelas soluções

tradicionais de CDNs. Nas CDNs existe um conjunto de pontos de presença, limitados e proprietários. Esse pontos dependem do serviço de DNS dinâmico para redirecionarem o tráfego, com base na quantidade de requisições, a um servidor. Esse servidor deve ter uma réplica do conteúdo e deve ser mais próximo geograficamente do consumidor para agilizar a recuperação do conteúdo. Na CCN qualquer nó pode atuar como fornecedor de conteúdo para outro nó, com isso pode-se dizer que a CCN estende as vantagens das CDNs generalizando o conceito de proximidade de consumo com políticas globais de armazenamento e distribuição de conteúdos.

# Capítulo 3

# Segurança em CCN e Trabalhos Relacionados

Como modelo de segurança, a CCN deve assegurar três propriedades em cada pedaço de conteúdo recebido: (1) se é válido, (2) se tem procedência, e (3) sua relevância [38]. Na CCN várias técnicas buscam garantir essas propriedades, porém ainda restam problemas de segurança a serem resolvidos e que afetam o bom funcionamento dessa arquitetura. Esse capítulo apresenta os mecanismos de segurança adotados nativamente pela CCN, explica porque a CCN é mais robusta do que a arquitetura TCP/IP contra ataques de negação de serviço tradicionais e apresenta novos ataques específicos para a CCN.

## 3.1 Segurança em CCN

Diferentemente da arquitetura TCP/IP da Internet atual, que deixa a responsabilidade pela segurança a cargo dos sistemas finais, a CCN auto-assegura os dados, exigindo que os produtores de conteúdo assinem criptograficamente todos os pacotes de dados [22]. A assinatura do produtor garante a integridade e permite determinar a procedência dos dados. Com isso, tem-se um modelo no qual a confiança do consumidor nos dados está desassociada do local onde o conteúdo foi obtido [38, 8]. Esse método também permite que qualquer nó verifique se um proprietário de chave pública é um produtor aceitável para um determinado pedaço de conteúdo.

Quando um publicador deseja publicar um certo conteúdo, ele deve fazê-lo utilizando três partes: o nome do conteúdo, o próprio conteúdo e a assinatura do publicador efetuada sob o mapeamento entre o nome e o conteúdo. Portanto, um usuário deve requisitar o conteúdo através de seu nome, obtendo o conteúdo em si e a assinatura do publicador sobre

o nome e o conteúdo. Com isso, o usuário precisa obter a chave pública do publicador para verificar a assinatura de forma a se certificar de que o conteúdo é íntegro e que ele foi realmente publicado pelo publicador detentor da chave pública [37].

## 3.2 DoS em CCN x DoS em Redes TCP/IP

Ataques de negação de serviço (DoS) são uma ameaça na Internet atual. A arquitetura CCN, entretanto, é mais robusta a esse tipo de ataque do que a pilha TCP/IP em virtude de duas características: o armazenamento de conteúdo pelos nós intermediários e a agregação de pacotes de interesse [17]. Ataques de esgotamento de banda e de reflexão, por exemplo, são pouco eficientes na CCN.

Ataques de esgotamento de banda inundam a vítima com requisições de serviço para esgotar seus recursos. Neste ataque, os pacotes devem chegar à vítima para que o ataque seja efetivo. Na CCN, no entanto, os pacotes não possuem o endereço de destino e os consumidores não podem garantir que os pacotes de interesse alcancem a origem do conteúdo, ou seja, o produtor e nesse a caso a vítima, porque qualquer nó pode responder ao interesse. Portanto, os consumidores maliciosos podem gerar uma quantidade enorme de pacotes de interesse para um dado conjunto de conteúdos, mas nenhum ou poucos desses pacotes alcançarão o produtor.

É importante ressaltar, também, que nós no caminho reverso entre o consumidor e o produtor armazenam conteúdos em *cache*. Assim, nós intermediários entre consumidor e produtor provavelmente irão satisfazer novos pacotes de interesse, que dificilmente alcançarão o produtor, dependendo da popularidade destes conteúdos. Além disso, a CCN reduz o número de pacotes de interesse transmitidos. Um nó só envia um pacote de interesse que não corresponde a uma entrada PIT. Caso contrário, o nó atualiza a lista de interfaces e descarta o pacote, como descrito na Seção 2.2.

Ataques de reflexão são baseados na técnica de falsificação de endereços IP (IP *spoofing*) e visam atacar vítimas diferentes simultaneamente. Na CCN, esses ataques são menos eficazes porque os pacotes de dados são sempre encaminhados para o consumidor através do caminho reverso percorrido pelo pacote de interesse. Consumidores também não podem garantir que os pacotes de interesse cheguem às vítimas intermediárias ou finais devido ao *cache* nos nós intermediários. Nós CCN, porém, podem enviar pacotes de interesse em todas as suas interfaces. Portanto, se o atacante e a vítima estão na mesma sub-rede do ataque, a reflexão pode ter certa eficácia [17]. Neste cenário, o atacante pode

enviar pacotes de interesse através de todas as suas interfaces com os endereços da camada MAC falsificados. Assim, múltiplas cópias do conteúdo são enviadas para a vítima. Apesar dos nós CCN não transmitirem o mesmo conteúdo mais de uma vez no mesmo domínio de difusão(*broadcast*), em topologias menores, e em posições privilegiadas, o ataque pode funcionar visto a pouca quantidade de saltos entre os consumidores e produtores [17].

Um tipo de ataque comum na Internet atual é o buraco negro (*Black-Hole*), nesse ataque um sistema autônomo (AS) mal configurado ou malicioso anuncia rotas inválidas de modo a motivar os outros ASes a transmitirem o seu tráfego para ele e assim enviá-los a um "buraco negro" e nunca responder ao que foi solicitado. A CCN é resistente ao buraco negro, pois roteadores têm acesso a mais informações do que seus equivalentes na rede IP e podem utilizar essas informações para detectar anomalias no processo de distribuição de conteúdo. Uma vez que cada conteúdo segue o mesmo caminho que o interesse que o solicitou, o número de interesses insatisfeitos (expirados) pode ser usado para determinar se um prefixo particular foi sequestrado. Além disso, roteadores podem manter estatísticas sobre o desempenho de cada enlace e a interface com relação a um prefixo particular (FIB), e assim mudar a sua estratégia de encaminhamento de acordo com esses dados.

Apesar de ser mais robusta do que a arquitetura TCP/IP aos ataques de DoS atuais, a arquitetura CCN possui ataques e vulnerabilidades identificados em trabalhos recentes [17, 37], que são discutidos na Seção 3.3.

## 3.3 Novos Ataques em CCN

Novos ataques trazem novos desafios à segurança na CCN, novas vulnerabilidades podem ser exploradas por usuários maliciosos. Em seu trabalho AbdAllah *et. al.* [1] identificam alguns novos tipos de ataques à arquitetura CCN. Uma taxonomia desses ataques pode ser observada na Figura 3.1.

A classificação observada na Figura 3.1 divide os tipos de ataques em quatro grandes áreas: Nomeação, Roteamento, Armazenamento de Conteúdos e Outros. Os ataques de nomeação visam afetar a privacidade da rede, manipulando o fluxo da informação. Os ataques de roteamento podem afetar a FIB e levar à negação de serviço por inundação. Os ataques de armazenamento de conteúdos afetam o CS e podem afetar a distribuição de conteúdos legítimos na rede. Os demais ataques, descritos como outros, visam comprometer a rede para obter acesso não autorizado a um conteúdo privado, por exemplo.

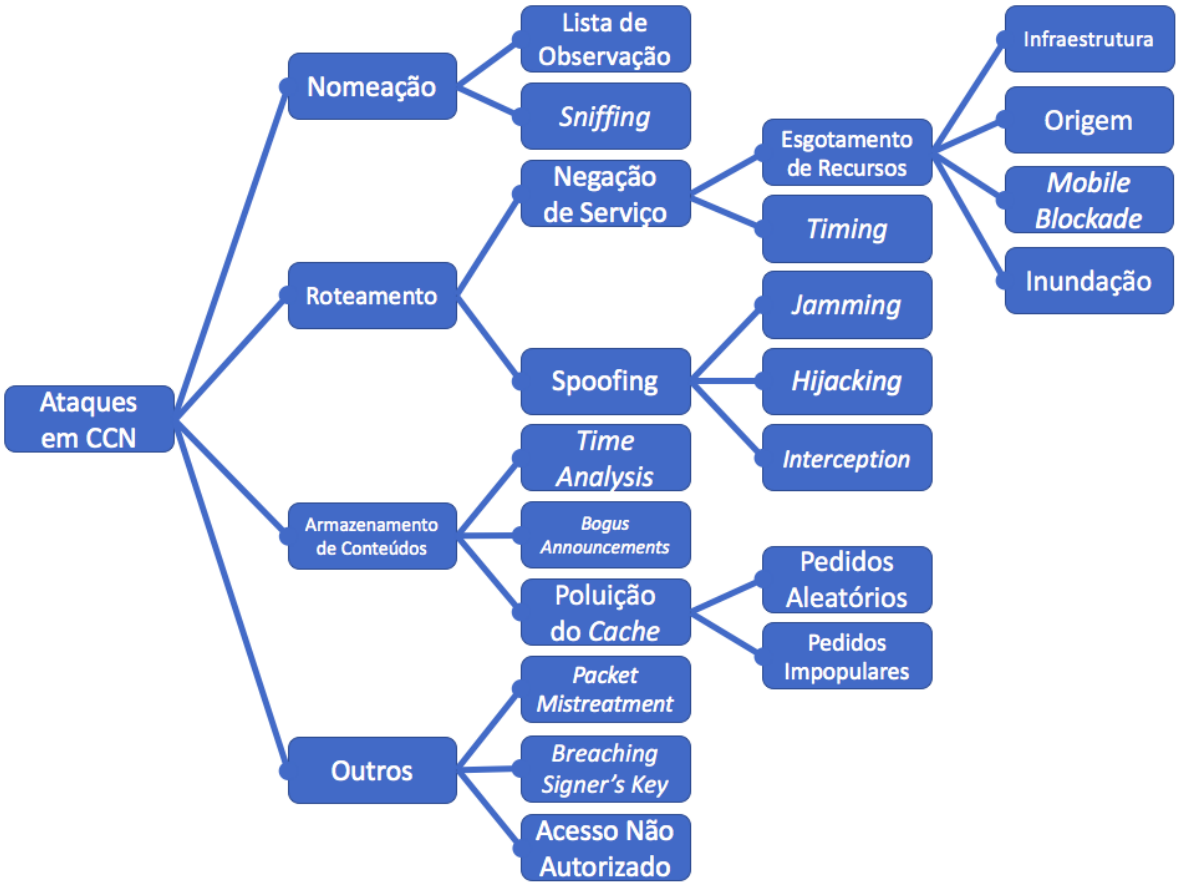



Figura 3.1: Tipos de ataques em CCN [1].

A seguir serão discutidos alguns desses ataques, bem como sua eficiência na CCN.

### 3.3.1 Ataques de Nomeação

Na CCN o armazenamento de dados está distribuído ao longo da rede, e até mesmo um consumidor pode servir conteúdo a um nó próximo. Isso implica que o armazenamento de dados pessoais dos clientes pode ser feito em roteadores. É importante ressaltar que nesse contexto um roteador pode ser outro cliente.

Esse método de distribuir informações dos usuários ao longo da rede pode trazer sérios problemas relacionados à privacidade do usuário, dados de acesso a sítios *Web* e arquivos baixados, podem estar disponíveis em outros clientes e roteadores [38]. Portanto, como são usados nomes de conteúdos para realizar várias tarefas no roteador, existe a possibilidade, por exemplo por má gestão, de um usuário mal intencionado obter dados referentes aos interesses e conteúdos recuperados por outro usuário. Se, por exemplo, um roteador de núcleo for comprometido, o atacante será capaz de monitorar os pedidos enviados pelos usuários que atravessam esse roteador, podendo comprometer a privacidade das

informações dos usuários.

### 3.3.2 Envenenamento do Cache

Um dos grandes desafios à CCN é saber se um determinado conteúdo é uma resposta válida a um interesse anteriormente enviado, ou seja, se o conteúdo é aquele que está sendo requisitado. Por exemplo, um nó malicioso pode manipular o pacote de dados de modo a unir uma assinatura válida com um bloco corrompido de dados. Assim, além de poluir os CSes dos nós intermediários, somente o consumidor poderá constatar que houve falha de integridade de dados. Existem trabalhos que apontam para possibilidade dos roteadores de conteúdo verificarem salto-a-salto a integridade dos pacotes de dados [36, 17, 35, 41], porém todos afirmam que fazer tal procedimento traria um alto *overhead* ao roteador. Assim propõem mecanismos probabilísticos de verificação de assinaturas que diminuem a possibilidade de um nó consumir um pacote corrompido, ou desse conteúdo acabar sendo armazenado em *cache* por outro nó no caminho. Deste modo pode-se afirmar que um roteador CCN enfrenta dois desafios: (1) sobrecarga de verificação de assinatura, e (2) gerenciamento de confiança [8].

O problema do envenenamento de conteúdos foi observado em redes P2P que até pouco tempo atrás eram bastante utilizadas, como a Gnutella e Kademila. Nessas redes, nas quais os conteúdos já estão bastante poluídos, tem-se uma grande probabilidade dos clientes consumirem conteúdos corrompidos. Esse problema pode afetar a CCN porque, assim como no caso do P2P, apenas o consumidor verificará se o *hash* do conteúdo e do nome do conteúdo condiz com o dado assinado pelo produtor. Alguns mecanismos de armazenamento tentam reduzir o efeito dessa poluição, porém esses mecanismos armazenam apenas os conteúdos pedidos com maior frequência [41].

### 3.3.3 Pedidos Aleatórios e Impopulares

Um problema importante de segurança em CCN é a vulnerabilidade à poluição do *cache* [36, 16]. Esse problema consiste na possibilidade de enviar pedidos de conteúdo aleatórios com o intuito de alterar a popularidade dos conteúdos, provocando o armazenamento de conteúdos de pouca popularidade nos *caches* dos roteadores. Outra versão dessa vulnerabilidade é fazer apenas requisições de conteúdos pouco populares, mas nesse caso, é necessário o conhecimento prévio da popularidade dos conteúdos. Em uma versão, talvez mais grave, do problema consiste em fazer pedidos de conteúdos falsos, criados

apenas para poluir os *caches* e que estão fora do universo de conteúdos válidos. Nesse caso o *cache* dos nós ficaria armazenando conteúdos poluídos e consequentemente deixaria de armazenar dados legítimos de usuários, o que reduziria a eficiência da distribuição de conteúdos, aumentando o tempo de recuperação. O ataque estudado nesse trabalho é classificado como ataque de poluição do *cache* que gera pedidos impopulares e que será detalhado no Capítulo 4.

## 3.4 Negação de Serviço

Os ataques de negação de serviço em CCN que esgotam os recursos da rede, afetando a infraestrutura, são classificados, principalmente, em dois tipos: ataques por inundação de interesses ou envenenamento de *cache* [35]. O objetivo dos ataques de inundação de interesses é sobrecarregar a PIT com solicitações de conteúdo enviadas por um nó malicioso a uma alta taxa [21]. Os pacotes de interesse maliciosos, em geral, solicitam conteúdos inexistentes, o que mantém por mais tempo a informação sobre esses interesses na PIT de um nó. A informação sobre um interesse pendente só é removida após o estouro de um temporizador. Enquanto aguarda pelo pacote de dados, o nó receberá novos interesses para outros conteúdos inexistentes. No pior caso, com a PIT cheia, um nó afetado não atenderá interesses legítimos, o que leva à queda de desempenho da rede.

Diferentemente dos ataques de inundação de interesses, o objetivo do ataque de envenenamento de *cache* é ocupar o *cache* dos nós com conteúdo poluído. Esse conteúdo é enviado por consumidores maliciosos para fazer com que nós armazenem um conteúdo que possua uma assinatura válida, porém corrompido ou aumentem a popularidade de conteúdos menos populares. No primeiro caso, o objetivo é reduzir o espaço disponível em *cache* para armazenar conteúdos legítimos e fazer com que consumidores recebam conteúdos corrompidos. No segundo, o objetivo é remover do *cache* conteúdos legítimos assumindo que uma política de substituição baseada na popularidade dos conteúdos é usada. Uma contramedida ao ataque de envenenamento de *cache* é a verificação da assinatura contida nos pacotes de dados [42]. Por padrão, a assinatura dos conteúdos é verificada apenas pelos nós de borda, ou seja, os consumidores, e não pelos nós intermediários da rede. Essa característica garante que os consumidores não recebam pacotes de dados contendo conteúdo malicioso. Nesse caso, o serviço da CCN é negado se os consumidores sempre receberem conteúdos inválidos. A solução de obrigar a verificação da assinatura de todos os conteúdos em todos os nós implica sobrecarga de processamento e, por isso, é de difícil adoção prática [17].

## 3.5 Trabalhos Relacionados

Todos os trabalhos descritos a seguir avaliam e/ou propõem contramedidas para os ataques de negação de serviço por inundação de interesses e por envenenamento de *cache*. Porém, nenhum, especificamente, propõe uma contramedida para comedir ataques em que consumidores e produtores maliciosos agem em conluio para gerar, disponibilizar e manipular a popularidade de conteúdos. O mecanismo proposto nesse trabalho visa comedir esse tipo de ataque, como detalhado no Capítulo 4.

Gasti *et al.* propõem um mecanismo de *push-back* como contramedida ao ataque de inundação de interesses [17, 12]. Esse mecanismo monitora a ocupação da PIT e identifica quando uma determinada interface está próxima de atingir seu número máximo de entradas. Desta forma, o mecanismo avalia e controla o fluxo de pacotes de interesse que contêm os mesmos nomes de prefixos para determinar um limiar. Além disso, a contramedida envia uma notificação na interface supostamente atacada que será recebida por um nó vizinho. Esse nó, por sua vez, deve propagar tal informação no sentido das interfaces atacadas e, ao mesmo tempo, limitar a taxa de interesses encaminhados que contenham o prefixo sob ataque. Portanto, o objetivo da contramedida é empurrar o ataque para o caminho de volta até o atacante, ou pelo menos para um nó no qual seja detectado [17]. A principal característica dessa contramedida é não modificar a arquitetura padrão proposta para a CCN. O ponto fraco do trabalho de Gasti *et al.* é que nem o impacto do ataque e nem a contramedida proposta são avaliados por simulação ou experimentos práticos.

Choi *et al.* [10], por outro lado, avaliam através de simulações a efetividade do ataque de inundação de interesses. Os autores mostram que em uma rede, em malha, com poucos nós, o desempenho é comprometido. Conclui-se que a vazão de dados total de consumidores legítimos diminuiu cerca de 65%. Da mesma forma, observa-se que o tempo médio de recuperação de conteúdos aumenta rapidamente, logo após o início do ataque. Os autores utilizam topologias simples que permitem apenas validar os testes em pequena escala. Não variam, por exemplo, a posição do atacante e nem do produtor para observar como a rede se comportaria nesses casos. Por fim, fazem testes solicitando interesses falsos de um produtor legítimo e, não avaliam o conluio.

Afanasyev *et al.* [2] também avaliam o ataque de inundação de interesse através de simulações, porém consideram diferentes cenários e uma rede de maior escala do que a usada no trabalho de Choi *et al.* Os autores também avaliam a contramedida baseada em

um mecanismo de *push-back* proposta por Gasti *et al.*. Os resultados mostram que essa contramedida é eficiente, pois isola por completo os atacantes de modo que eles causem pouco ou nenhum impacto no desempenho percebido por usuários legítimos. Os autores não variam a posição do atacante. Para validar o mecanismo definiram o pior cenário, onde os interesses não serão satisfeitos com *cache* de um nó intermediário. Deste modo os dados são sempre encaminhados diretamente para o produtor. Os autores justificam essa medida afirmando que esse seria o pior caso para o ataque avaliado.

Bernardini *et al.* [6] apresentam uma estratégia de *cache* adaptada às redes CCN chamada MPC (*Most Popular Content*). A técnica funciona armazenando em *cache* apenas o conteúdo popular em relação à vizinhança. Os nós vizinhos trocam informações sobre os conteúdos armazenados em *cache* de modo a construir uma tabela de conteúdos mais populares. Os autores demonstram, através de experimentos, que o MPC é capaz de armazenar em *cache* menos conteúdos, enquanto, ao mesmo tempo, ele ainda consegue uma maior taxa de acerto do *cache* e supera a estratégia de *cache* padrão existente na CCN. Todos os experimentos somente avaliam a otimização do *cache*, e nenhuma forma de ataque é avaliada. Os autores finalizam sua análise comprovando que sua solução é uma estratégia de descarte mais eficiente que a nativa da rede.

Goergen *et al.* [20] observam a detecção, através das anomalias geradas, de ataques comuns em redes CCN. Os autores analisam as três tabelas que compõem um nó CCN: PIT, FIB e CS. São definidas métricas que ajudam a medir a eficácia de um ataque na rede. Propõem uma solução para detectar ataques em tempo real utilizando dados estatísticos como a quantidade de interesses por segundo numa interface de rede. Utilizam um algoritmo de classificação SVM (*Support Vector Machines*) para detectar ataques de inundação de interesses. Por outro lado os autores se concentram apenas em ataques que afetem a PIT de várias formas, como interesses rápidos para conteúdos inexistente ou impopulares. Não foi levado em conta o efeito que será causado por dois ataques em conjunto e ataques em conluio.

Ghodsi *et al.* identificam as semelhanças existentes e as diferenças importantes sobre arquiteturas de rede orientadas a dados ou centradas em conteúdo [19]. Os autores também discutem alguns problemas de pesquisa nessa área com relação aos parâmetros adotados. Põem em dúvida se os benefícios da abordagem ICN podem ser obtidos de forma completa, ou podem ser obtidos de forma mais incremental, com adoção de *cache* sobre a rede IP vigente. Por fim, definem valores que podem ser utilizados para validar essa abordagem para a rede.

Guimarães *et al.* [21] propõem um modelo analítico que ajuda a compreender as condições que tornam a arquitetura CCN mais ou menos suscetível a ataques de inundação de interesses na PIT. O modelo é útil para mostrar que apenas uma pequena fração de tráfego malicioso é suficiente para aumentar significativamente as chances de sucesso de um ataque de negação de serviço na PIT. Também mostram que o parâmetro *timeout*, quando definido corretamente, pode reduzir a vulnerabilidade do sistema, ou seja, os autores mostram que existe um *timeout* ideal para o PIT. Em sua avaliação, os autores mostram que o modelo é útil para analisar as circunstâncias em que a PIT é mais vulnerável a ataques de inundação, que é definido de acordo um *timeout* específico. O autores ainda formulam uma técnica de otimização que maximiza o rendimento do sistema. Os autores não avaliam a possibilidade do ataques acontecerem em conluio produtor-consumidor.

Os trabalhos a seguir tratam dos ataques de envenenamento de *cache*. Ribeiro *et al.* [35] propõem um mecanismo de verificação probabilística de assinaturas. O mecanismo proposto é eficiente, porém se mostrou dependente da topologia de rede utilizada. Quanto maior o número de saltos, maior a probabilidade do conteúdo poluído ser descartado ao longo do caminho.

Kim *et al.* [25] investigam o impacto de fluxos de conteúdo de longa duração na CCN. A presença de fluxos de longa duração pode ter efeito similar ao ataque de envenenamento de *cache*. Se fluxos de longa duração ocuparem temporariamente um *cache* de um nó por determinado conteúdo, eles podem expulsar pedaços de conteúdos populares do *cache*. Consequentemente, reduz-se a taxa de acertos do *cache* (*cache hit*). Os resultados das simulações mostram que há degradação da taxa de acertos do *cache* quanto maior é o número de fluxos de longa duração.

Deng *et al.* [14] fazem uma classificação dos ataques ao *cache* em dois tipos: aqueles que solicitam conteúdos não populares para aumentar a popularidade desses conteúdos, e outro pela emissão de conteúdos novos (não populares) para comprometimento do *cache* local. Essa segunda, de certa, forma assemelha-se ao ataque em conluio. Apesar dos produtores não agirem em conluio com consumidores atacantes, os consumidores solicitam sempre novos conteúdos pouco populares na rede com o objetivo de alterar a popularidade dos conteúdos. Os autores não falam da possibilidade de conteúdos corrompidos estarem em *cache* e nem de verificação de assinaturas, isso porque não tratam especificamente da arquitetura CCN. Por fim, uma contramedida é proposta, porém não se mostra eficaz quando a topologia é em larga escala.

Cieza *et al.* [11] avaliam o impacto dos ataques de poluição do *cache* em redes CCN

sem-fio. Através de simulações, os autores comparam o desempenho da rede considerando o uso da políticas de *cache* proativo e não proativo. Realizam experimentos com e sem mobilidade, em condições normais de operação e sob ataque. O desempenho é avaliado em relação à taxa de entrega, atraso, à ocupação maliciosa do *cache* dos nós, taxa de acerto do *cache* e número médio de saltos atravessados pelos conteúdos. Nos resultados, o *cache* proativo potencializou o ataque de poluição dos *caches*, em contrapartida melhorou a eficiência da rede. Apenas uma topologia em grade 7x7 foi usada nas simulações, houve poucas variações de cenários e os autores não avaliaram a possibilidade do ataque acontecer em conluio.

Conti *et al.* [12] enfocam ataques de poluição de *cache* em CCN. Mostram, através de simulações, que esses ataques podem ser implementados em CCN, e que a sua eficácia não está limitada a pequenas topologias. Em seguida, ilustram que as contramedidas proativas existentes, são ineficazes contra adversários realistas. Finalmente, mostram uma nova técnica para detectar ataques de poluição baseada em um algoritmo que primeiramente aprende sobre o ataque, definindo limiares e fatores estatísticos, e depois implementa uma limitação em fazer *cache* de conteúdos possivelmente maliciosos. O trabalho tem como objetivo comparar a contramedida proposta com o CacheShield. Outro problema é que além da contramedida de não reagir ao ataque para inibi-lo, o que indica que pacotes maliciosos serão entregues, os autores não abordam o conluio.

## 3.6 O Mecanismo CacheShield

Xie *et al.* [41] propõem um mecanismo chamado CacheShield, cujo objetivo é aumentar a robustez do *cache* contra ataques de poluição, em particular perturbações de localidade (*locality disruption*). O mecanismo tenta impedir que conteúdos não populares sejam armazenados no *cache*. Quando um roteador habilitado para o CacheShield recebe um pedaço de conteúdo, o nó executa uma função de proteção que determina se o pedaço de conteúdo deve ser armazenado em *cache* ou não. Se a função de proteção retornar verdadeiro, o roteador encaminha o interesse e armazena em *cache* o pedaço de conteúdo. Se a função de proteção retorna falso, apenas o nome do pedaço de conteúdo (ou o *hash* de seu nome) e um contador são armazenados no *cache* num espaço reservado, fora da área do CS. Se o mesmo pedaço de conteúdo, que ainda não está armazenado em *cache*, é solicitado novamente, o contador correspondente é incrementado, e a função de proteção é novamente executada. Assim a cada vez que o pedaço de conteúdo é avaliado pela função aumenta-se a chance desse ser efetivamente armazenado no *cache*. A função de proteção executa uma

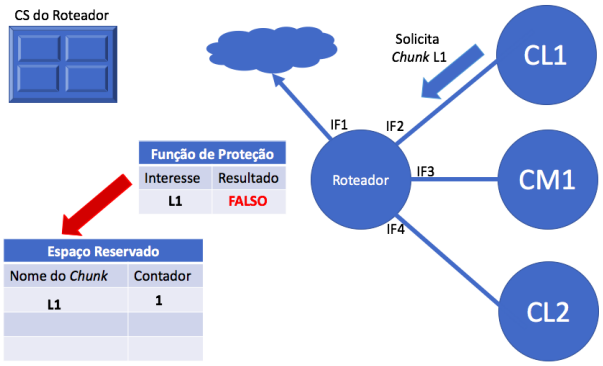
escolha aleatória com base em uma função probabilística: $\psi(t) = \frac{1}{1+e^{(p-t)/q}}, t = 1, 2, 3, ...$

Onde $t$ representa a *t-ésima* requisição para um determinado pedaço de conteúdo no CS, e $p$ e $q$ são os parâmetros da função. Os parâmetros $p$ e $q$ podem ser ajustados a fim de alcançar os melhores resultados. Usando topologias pequenas, de até nove roteadores, determinou-se que a função de proteção geralmente funciona bem com $p = 20$ e $q = 1$.

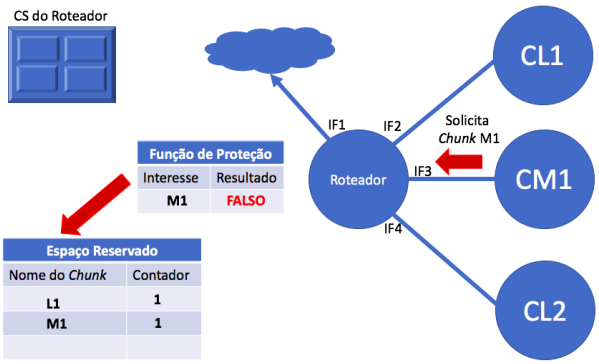
Como exemplo de funcionamento da função de proteção, tem-se o observado na Figura 3.2. De acordo com que pode ser observado na Figura 3.2(a), o consumidor legítimo 1 (CL1), solicita o conteúdo legítimo L1. Como esse conteúdo ainda não foi solicitado por nenhum outro nó da rede, ao passar pela função de proteção as chances desse conteúdo ir para o CS do roteador são pequenas. Então após receber uma retorno falso da função de proteção, apenas o nome do conteúdo e um contador são armazenados numa área reservada, fora do CS. De acordo com a Figura 3.2(b), quando o consumidor malicioso 1 (CM1), solicita o conteúdo malicioso M1, que também não havia passado pela função de proteção e não tem cópia no CS do roteador, passa pela função de proteção, as chances desse conteúdo ir diretamente ao CS do roteador são pequenas. Então, após receber falso da função de proteção, M1 segue para o espaço reservado com contador incrementado de 1, do mesmo modo que ocorreu com L1. Quando o consumidor legítimo 2 (CL2) solicita o mesmo conteúdo L1, já solicitado por CL1, as chances da função de proteção retornar verdadeiro são maiores, visto que a probalidade calculada pela função de proteção se aproxima de 1 quanto mais vezes o conteúdo é solicitado. No exemplo, a função retorna verdadeiro e então L1 segue para o CS do roteador.

O objetivo principal da função de proteção é dificultar que objetos de conteúdo impopulares sejam armazenados no *cache*. Para impedir os atacantes de prever a decisão, utiliza-se a função de proteção. A função calcula uma probabilidade de *cache* para cada pedaço de conteúdo solicitado. Quanto mais solicitações um pedaço de conteúdo recebe, maior será a probabilidade correspondente, ou seja, é mais provável que seja armazenado em *cache*.

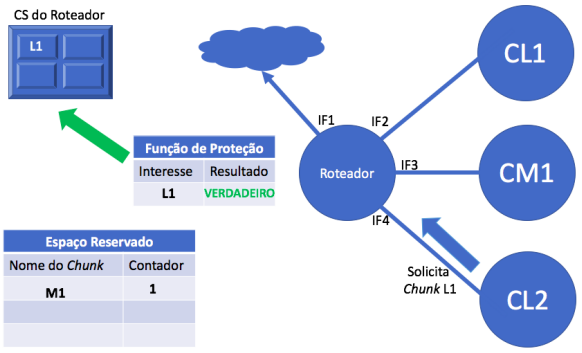
Qualquer política de substituição, como LRU ou LFU, pode ser usada em conjunto com o CacheShield. Uma vez que o *cache* estiver cheio, espaços reservados de conteúdo, ou seja, nomes e contadores, estão sujeitos às mesmas regras de substituição como conteúdo em *cache*. São realizadas análises dos empregos de políticas de substituição de *cache* por tempo, (LRU e LFU), baseada em um esquema de função, *Greedy Dual-Size Frequency* (GDSF). Além destas análises, são medidos os impactos das variáveis da função de decisão de armazenamento em *cache* realizado pelo algoritmo. Diversas outras medidas, como o




(a) CL1 Solicita L1.




(b) CM1 Solicita M1.




(c) CL2 Solicita L1.

Figura 3.2: Exemplo de funcionamento do CacheShield.

ganho sem a inserção de um atacante na rede e a robustez do sistema quando um atacante está presente, também foram medidos. As simulações são efetuadas em duas topologias pequenas, uma com 4 roteadores de conteúdos e total de 10 nós, e outra com 9 roteadores de conteúdo e total de 18 nós, ambas em malha. Os autores avaliam principalmente a melhora na taxa de acerto do *cache* em relação a rede sem uso de contramedida e com uso das políticas de descarte LRU e LFU. O CacheShield reduz a efetividade do ataque pois trabalha com probabilidade, nesse caso, a chance de pegar um pacote malicioso operando em alta taxa é grande. Isso não demonstra a efetividade do mecanismo, pois muitos falsos-positivos podem acontecer, principalmente se atacantes operam em altas taxas. O mecanismo não impede o encaminhamento do pacote, somente impede de ser armazenado. Isso pode levar, em cenários maiores, a negação de serviço em nós mais distantes que os atacados inicialmente. O mecanismo CacheShield será usado, para fins de comparação, com o mecanismo proposto por esse trabalho.

# Capítulo 4

# Ataque de Negação de Serviço por Conluio Consumidor-Produtor

Conforme abordado no Capítulo 3, os ataques de negação de serviço para CCN identificados na literatura, bem como os mecanismos de contramedida utilizados, se baseiam na possibilidade de um nó em particular, produtor ou consumidor, manipular as estruturas da CCN para afetar seu funcionamento. Deng *et al.* e Xie *et al.* [14, 41] identificam a possibilidade de consumidores e produtores agirem em conluio para manipular o *cache* dos nós. Este capítulo define o ataque por conluio produtor-consumidor na CCN e explica porque a política padrão de verificação de assinaturas da CCN não é suficiente para evitar tal ataque.

## 4.1 Definição do Ataque

Com o ataque em conluio, o objetivo é prejudicar o consumo indireto de conteúdos, isto é, obrigar um consumidor legítimo a recuperar o conteúdo desejado diretamente do produtor. Esse objetivo é alcançado através da manipulação da popularidade dos conteúdos armazenados em *cache*. Consumidores maliciosos enviam pacotes de interesse para um grupo de conteúdos que existem e que são respondidos pelo produtor malicioso. Assim, se solicitado com frequência, um conteúdo se torna popular, apesar de não ter sido solicitado por usuários legítimos. Por isso, o conteúdo é dito malicioso.

O ataque em conluio é possível, porque a CCN emprega políticas de substituição do *cache* baseadas, em sua maioria, na popularidade dos conteúdos. Assim, se um determinado conteúdo não é solicitado com frequência ou não foi solicitado recentemente pelos consumidores, ele é considerado menos popular. Dessa forma, terá prioridade de

descarte quando houver necessidade de armazenar novos conteúdos. Ao solicitar um conjunto específico de conteúdos e em taxas altas, os nós maliciosos manipulam a política de substituição do *cache*. Com mais conteúdos maliciosos em *cache*, maior a taxa de erro para os conteúdos legítimos e, consequentemente, maior a necessidade de nós legítimos terem que recuperar o conteúdo diretamente do seu produtor. Mesmo que os consumidores legítimos não tenham que consumir diretamente dos produtores, eles terão seus interesses encaminhados por mais saltos até conseguir o conteúdo desejado.

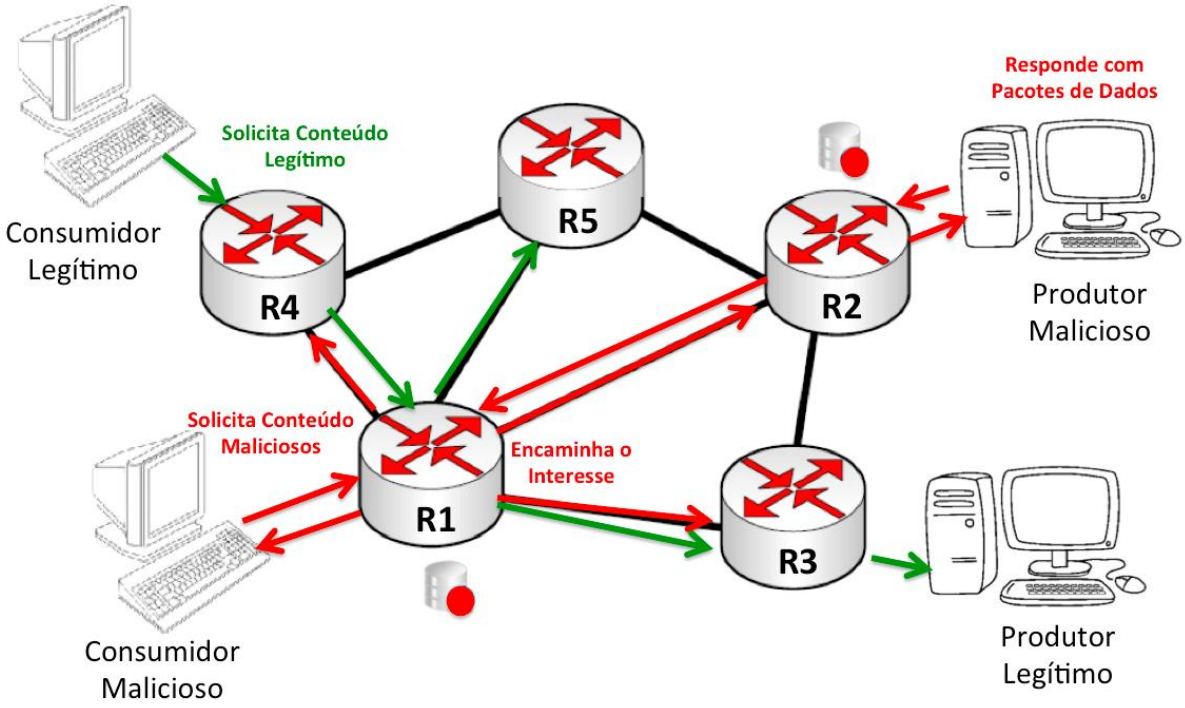



Figura 4.1: O ataque em conluio produtor-consumidor: nós legítimos e maliciosos em ação.

A Figura 4.1 ilustra como os consumidores e produtores maliciosos atuam no ataque. O tráfego malicioso é representado por setas vermelhas e o tráfego legítimo é representado por setas verdes. O consumidor legítimo apenas envia pacotes de interesse para o conteúdo publicado pelo produtor legítimo, bem como o consumidor malicioso envia pacotes de interesse para o conteúdo publicado pelo produtor malicioso. Roteadores de conteúdo, $Rn$, encaminham e armazenam em *cache* tráfego legítimo e malicioso. Neste exemplo, os roteadores $R1$ e $R2$ estão no caminho entre o produtor malicioso e o consumidor malicioso. $R1$ também está no caminho entre o consumidor legítimo e o produtor legítimo. Nesse caso, se $R1$ receber continuamente pacotes de interesse para diferentes conteúdos impopulares publicados pelo produtor malicioso, o conteúdo legítimo pode ser removido de seu *cache*. Consequentemente, a disponibilidade de conteúdos legítimos diminui e esse

conteúdo só pode ser recuperado diretamente no produtor legítimo, no pior dos casos.

Uma das principais razões para que o ataque em conluio consumidor-produtor seja bem sucedido é o fato de que os pacotes de interesse e de dados usados no ataque são legítimos para a rede e, portanto, não são detectados por mecanismos de verificação de assinaturas. O pacote com o conteúdo malicioso possui uma assinatura válida, carrega a chave do publicador, ou informações para obtê-la, e assim, passa no teste de verificação de integridade e autenticidade. Logo, não é identificado como malicioso e nem descartado.

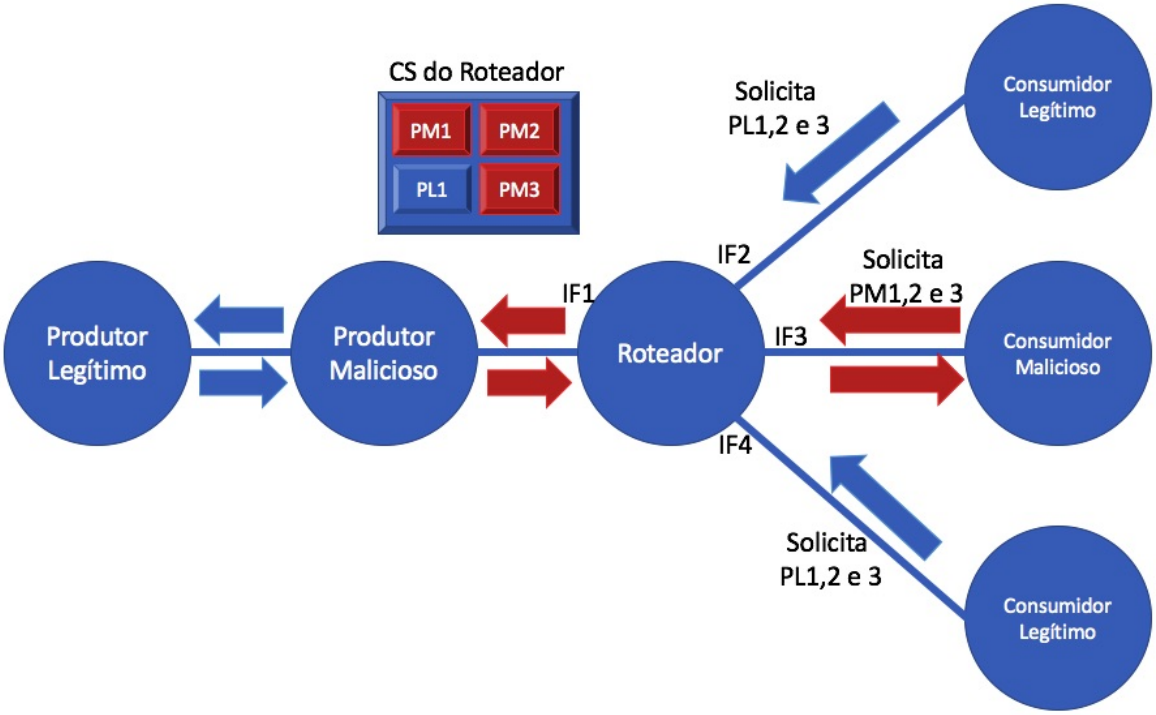



Figura 4.2: Rede sob ataque em conluio produtor-consumidor.

De acordo com o observado na Figura 4.2, um consumidor malicioso, solicitando conteúdo malicioso em altas taxas de um produtor malicioso, pode comprometer o consumo legítimo de conteúdo por consumidores legítimos. Isso ocorre, por exemplo, se o produtor malicioso e o consumidor malicioso estiverem utilizando um caminho que também é utilizado pelos consumidores legítimos para recuperar o conteúdo do produtor legítimo. No exemplo da Figura 4.2, o consumidor malicioso solicita os pedaços de conteúdo PM1, PM2 e PM3 numa taxa alta que compromete o CS do roteador vizinho dos consumidores legítimos. Quando os consumidores legítimos precisam recuperar os pedaços PL1, PL2 e PL3 encontram apenas o PL1 no CS do roteador, sendo obrigados a recuperar o restante dos pedaços de conteúdo diretamente do produtor legítimo.

Outro objetivo do ataque em conluio é reduzir a eficiência da PIT, ao enviar pedidos

de interesses para diferentes conteúdos maliciosos disponibilizados por produtores maliciosos a uma alta taxa. Dessa forma, é possível burlar o mecanismo de *push-back* proposto por Gasti *et al.* [17]. Esse mecanismo é eficiente contra a inundação de pacotes de interesse porque consegue identificar prefixos de nomes de conteúdo que frequentemente estão pendentes na PIT, uma vez que o conteúdo solicitado é inexistente. Porém se o consumidor e produtor estiverem agindo em conluio, os pacotes de interesse terão uma entrada na PIT de um nó somente até o conteúdo malicioso, que existe, retornar. Não haverá *timeout* da entrada da PIT para esse conteúdo, nesse caso. Portanto, o mecanismo *push-back* não terá sucesso ao tentar identificar o ataque, pois os pacotes de interesse receberão uma resposta legítima e suas entradas serão removidas da PIT. Nesse caso, o ataque em conluio não provoca o esgotamento de recursos de armazenamento da PIT em um nó. O objetivo do ataque é gerar uma grande quantidade de pacotes de interesses, fazendo com que um nó tenha que manipular muitas solicitações de conteúdo maliciosas em detrimento a interesses legítimos, o que pode levar a negação de serviço nesse nó.

A negação de serviço do *cache* se difere da forma clássica. Um nó nega serviço em *cache* quando não dispõe do conteúdo solicitado por ter seu CS armazenando de conteúdos poluídos e impopulares. Deste modo, diz-se que o nó nega serviço quando pela impossibilidade de responder a um pacote de conteúdo, encaminha o interesse ao nó vizinho. Ou seja, o serviço geralmente é negado em roteadores de conteúdo, onde um conteúdo popular não está no CS em detrimento a conteúdos impopulares.

# Capítulo 5

# O Mecanismo Cache nFace

De acordo com o que pôde ser observado no Capítulo 4, o ataque em conluio é efetivo porque pacotes maliciosos de interesse e de conteúdo são tratados pelos roteadores da mesma forma que pacotes legítimos. Os pacotes maliciosos não são corrompidos ou têm campos de cabeçalho alterados durante o encaminhamento. Assim, é difícil identificar e bloquear o tráfego malicioso. Políticas de descarte, baseadas em popularidade de conteúdos, também podem ser ineficientes, em virtude da alta taxa de envio de interesses maliciosos. A contramedida proposta chamada de Cache nFace não tenta identificar o tráfego malicioso. A ideia principal da proposta é criar sub-*caches* em um nó a partir da divisão do espaço total de armazenamento em *cache* desse nó. As próximas seções detalham o algoritmo de divisão de *cache*, explicam as modificações necessárias nos procedimentos de encaminhamento de pacotes e armazenamento de conteúdos com o Cache nFace e exemplificam o uso do mecanismo proposto.

## 5.1 Funcionamento do Cache nFace

O Cache nFace divide o CS de um nó em sub-*caches*. Cada sub-*cache* está associado a uma das interfaces de rede do nó, e o seu tamanho é definido de acordo com a taxa de transmissão nominal da interface, como mostra o Algoritmo 1. Esse algoritmo recebe como entradas o tamanho total do *cache* do nó, $S$, o número de interfaces do nó, $N$, $I$ o conjunto composto por todas as interfaces, e a taxa de transmissão $q_i$ de cada interface $i$ do nó e retorna o tamanho do sub-*cache* $s_i$ de cada interface $i$. $Q$ é soma das taxas de transmissão das $N$ interfaces. Dessa forma, o tamanho de cada sub-*cache* é proporcional à capacidade de transmissão de cada interface.

**Algoritmo 1:** MECANISMO DE DIVISÃO DO CACHE NFACE

**Entrada:** $S, q_i, N, I$

1 { *O tamanho total do cache do nó S;*
2 *A taxa de transmissão $q_i$ de cada interface i;*
3 *O número de interfaces do nó N;*
4 *O conjunto composto por todas as interfaces I. }*

**Saída:** Sub-Cache $s_i$

5 **início**
6 $Q \leftarrow \sum_{n=1}^{N} q_n$
7 **para** *cada* $i \in I$ **faça**
8 $s_i \leftarrow \frac{S * q_i}{Q}$
9 **fim**
10 **fim**
11 **retorna** $s_i$

A Figura 5.1 ilustra um exemplo de funcionamento do Cache nFace. Nesse exemplo, o nó possui um *cache* de capacidade $S = 1000$ conteúdos e três interfaces de rede ($N = 3$): IF1, IF2 e IF3, cujas taxas de transmissão são, respectivamente, $q_1 = 2$ Mb/s, $q_2 = 3$ Mb/s e $q_3 = 5$ Mb/s. Com isso, o Algoritmo 1 produz as seguintes saídas: $s_1 = 200$ conteúdos; $s_2 = 300$ conteúdos e $s_3 = 500$ conteúdos.

## 5.2 Encaminhamento e Armazenamento com o Cache nFace

Com o Cache nFace, somente o CS de um nó é modificado, ou seja, ao invés de um único *cache* agora existe um sub-CS para cada interface. Tanto a FIB quanto a PIT não são alteradas. Além disso, o encaminhamento de pacotes de interesse e de pacotes de dados não se alteram. Durante o encaminhamento de um pacote de interesse, todos os sub-*caches* são verificados para saber se o nó possui uma cópia do pedaço de conteúdo solicitado e não somente o sub-*cache* da interface de recepção do interesse. Os conteúdos continuam sendo encaminhados pelo caminho reverso ao percorrido pelo interesse. A única modificação é que, quando um interesse é atendido por um pacote de dados, o pedaço de conteúdo contido nesse pacote é armazenado no sub-*cache* correspondente à interface que recebeu o interesse.

Vale ressaltar também que um mesmo conteúdo não é armazenado em diferentes sub-

*caches*. Isso acontece, pois apesar de cada sub-*cache* ter seu espaço de armazenamento separado dos demais, juntos eles formam o CS do nó, seguindo características globais. Assim, impede-se a duplicação de conteúdos no *cache* de um nó. Esse mesmo motivo obriga todos os sub-*caches* a usarem a mesma política de descarte definida para o CS. O pacote de dados ficará armazenado sempre no sub-*caches* da interface que primeiramente receber o pacote de interesse.

No exemplo da Figura 5.1, o pacote de interesse indicado pela Seta 1, é recebido pelo nó na IF1. Após a análise do CS, da PIT e da FIB, o nó decide encaminhar o pacote de interesse pela IF2, como indica a Seta 2. Por fim, o pacote de dados é recebido através da IF2, como indica a Seta 3. Ao ser recebido, o pedaço de conteúdo contido nessa pacote é armazenado no sub-*cache* $s_1$, associado à IF1.

O princípio de funcionamento do Cache nFace é simples. Se muitos interesses maliciosos inundam uma determinada interface, esses pacotes apenas afetarão o sub-*cache* associado a essa interface. Deste modo o nó pode continuar a responder interesses legítimos com pedaços de conteúdo legítimos mesmo sob ataque. Isso ocorre porque um pacote legítimo pode estar armazenado em outro sub-*cache* diferente do sub-*cache* da interface que está sendo atacada e, assim, produzir um acerto de *cache*, mesmo que o interesse tenha sido recebido através da interface atacada.

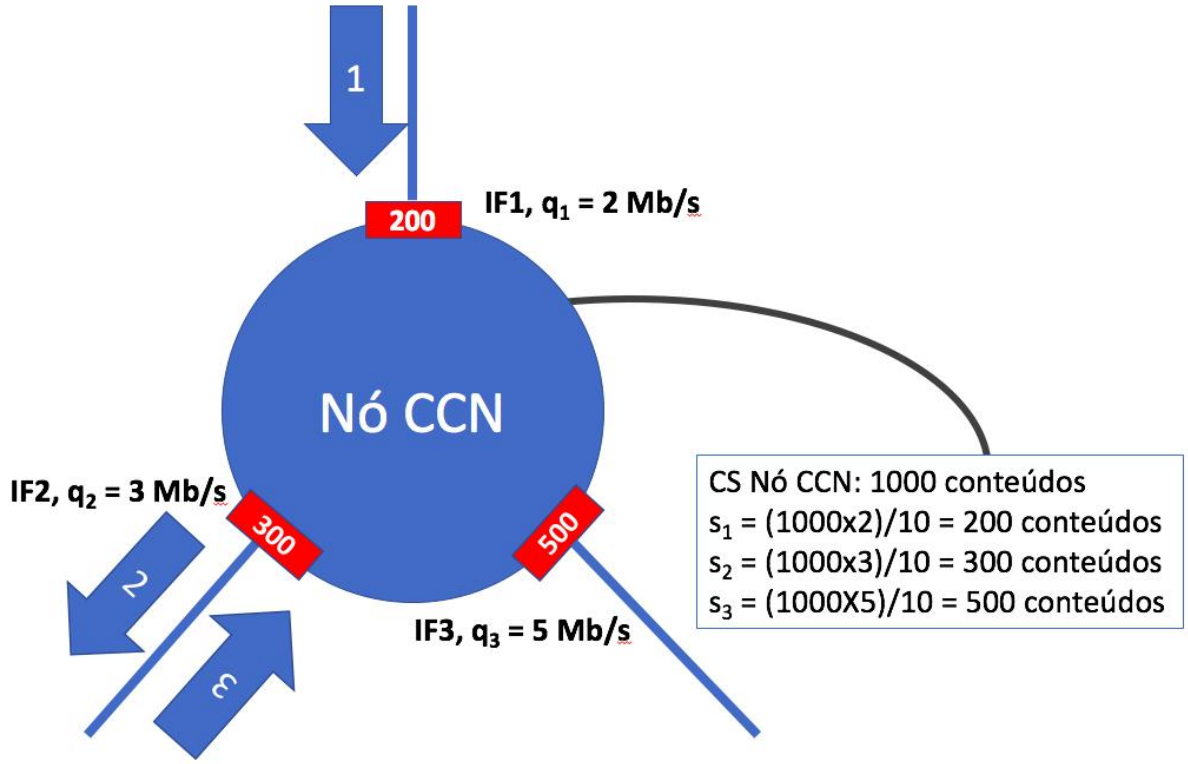



Figura 5.1: Um exemplo de uso do mecanismo Cache nFace.

A premissa para o bom funcionamento do Cache nFace é que a probabilidade de pacotes de interesse usados em um ataque de conluio produtor-consumidor serem recebidos a altas taxas por todas as interfaces de um nó é pequena. Os roteadores de borda geralmente são mais afetados por estarem ligados a uma interface que pertence à rede do atacante. Assim, eles recebem altas taxas de interesses pela interface ligada à rede do atacante. Porém, a probabilidade de que as demais interfaces de rede do roteador de borda pertençam a redes de outros atacantes e/ou recebam tráfego malicioso encaminhado por outros roteadores em altas taxas é pequena. Deste modo, pelo menos uma interface que não pertence à rede do atacante pode responder a interesses legítimos que estão armazenados no seu sub-*cache*. Isso pode acontecer em cenários em que produtores e consumidores maliciosos tenham poucos saltos entre eles e a topologia não possua muitos caminhos alternativos entre eles. Isso indica que posição do atacante dentro da topologia e a quantidade de caminhos entre eles é fundamental para a eficiência do ataque.

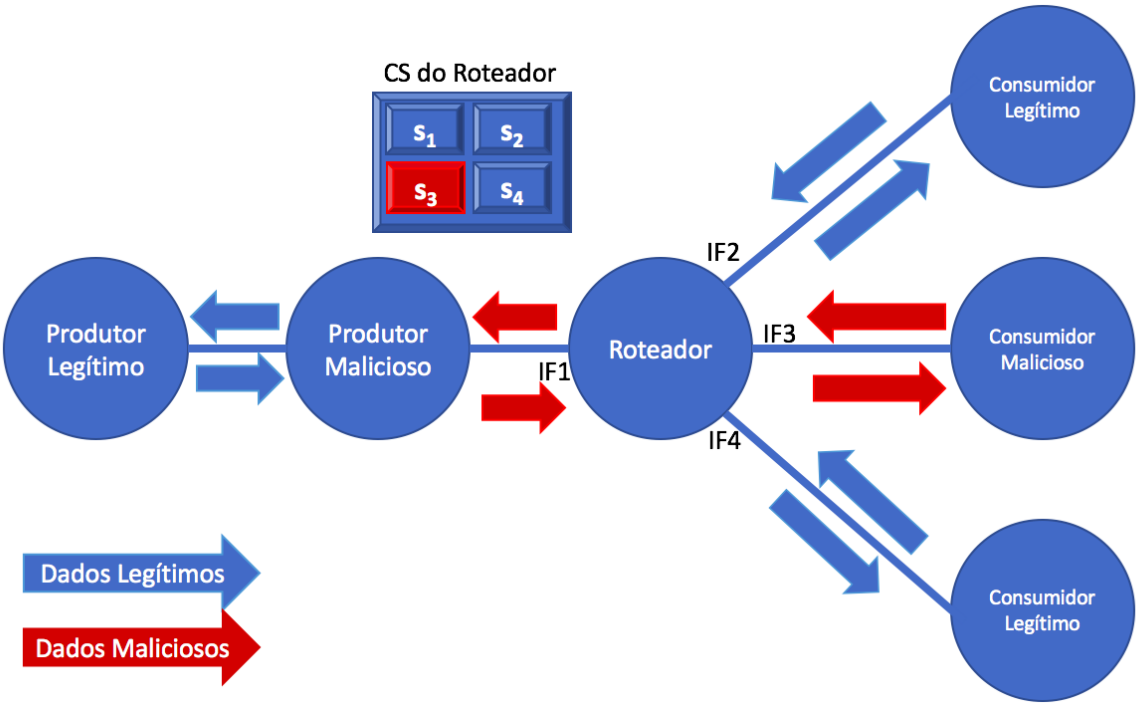



Figura 5.2: Cache nFace em ação.

Como pode ser observado pela Figura 5.2, mesmo que o produtor malicioso esteja localizado no caminho entre os consumidores legítimos e o produtor legítimo, pior caso possível, ainda assim o CS do roteador não seria totalmente afetado. Como os interesses maliciosos, enviados pelo consumidor malicioso, chegam ao roteador pela IF3, apenas o sub-*cache* $s_3$ do nó armazenará o conteúdo malicioso enviado pelo produtor malicioso e recebido pela IF1 do roteador. Isso permite que o roteador ainda manipule interesses

legítimos utilizando os sub-*caches* $s_2$ e $s_4$ que estão ligados aos consumidores legítimos. Isso acontece porque consumidores legítimos não solicitam pacotes maliciosos, então os sub-*caches* das interfaces que recebem interesses legítimos enviados por esses nós não armazenarão conteúdos gerados pelo produtor malicioso. Deste modo, os sub-*caches* $s_1$, $s_2$ e $s_4$ garantiriam que o roteador continuasse a responder a interesses legítimos, mesmo sob ataque. Caso o mecanismo Cache nFace não estivesse em ação, o CS do roteador poderia ser afetado como um todo, levando à negação de serviço aos consumidores legítimos.

O tamanho de um sub-*cache* não varia de acordo com à vazão na sua interface associada. Caso ele fosse proporcional a vazão, um ataque de inundação poderia reduzir o tamanho dos sub-*caches* de interfaces que não estão sob ataque. Isso provocaria o descarte de pedaços de conteúdo legítimo e o desempenho da rede, nesse caso, seria semelhante ao do cenário sem nenhuma contramedida.

# Capítulo 6

# Avaliação do Mecanismo Cache nFace

Nesse capítulo serão discutidos os cenários utilizados na avaliação dos mecanismos Cache nFace e CacheShield. Ainda serão definidas as configurações, métricas e topologias utilizadas nas simulações. Por fim, uma análise dos resultados é realizada para discutir a eficiência de ambos mecanismos sob ataque em conluio.

## 6.1 Cenários de Avaliação

As topologias de rede, usadas na avaliação do ataque em conluio e das contramedidas, são as seguintes. Uma topologia composta por 32 nós dispostos em forma de árvore, como mostra a Figura 6.1(a). Outra topologia em malha, criada com o mapeador de topologias Rocketfuel [39], com 192 nós, como mostra a Figura 6.1(b).

Na topologia em árvore, os 24 nós-folha são consumidores. O número de consumidores legítimos (CL) é fixo em todas as configurações e igual a 16. O número de consumidores maliciosos (CA) varia de 0 a 8. A posição dos CLs e dos CAs é definida aleatoriamente em cada rodada de simulação. O produtor legítimo (PL) é sempre o nó raiz. O produtor malicioso (PA) é o nó filho do nó raiz. Os demais 6 nós que compõem a topologia são os roteadores da rede (RTR). Os enlaces que interconectam os nós possuem taxa de transmissão de 100 Mb/s e atraso de 1 ms.

Na topologia em malha, tem-se 92 nós-folha como consumidores. O número de consumidores legítimos é fixo em todas as configurações e igual a 68. O número de consumidores maliciosos varia de 0 a 24, valores escolhidos para manter a mesma proporção da topologia em árvore. A posição dos consumidores legítimos e consumidores maliciosos é definida, aleatoriamente, em cada rodada de simulação. Existem 2 produtores legítimos

e 1 produtor malicioso com posições aleatórias na rede. Os demais 97 nós que compõem a topologia são os roteadores (58 de borda e 39 de núcleo). Os enlaces que interconectam os nós possuem taxa de transmissão entre 124 Kb/s e 92 Mb/s e atraso variante entre 1 e 70 ms.

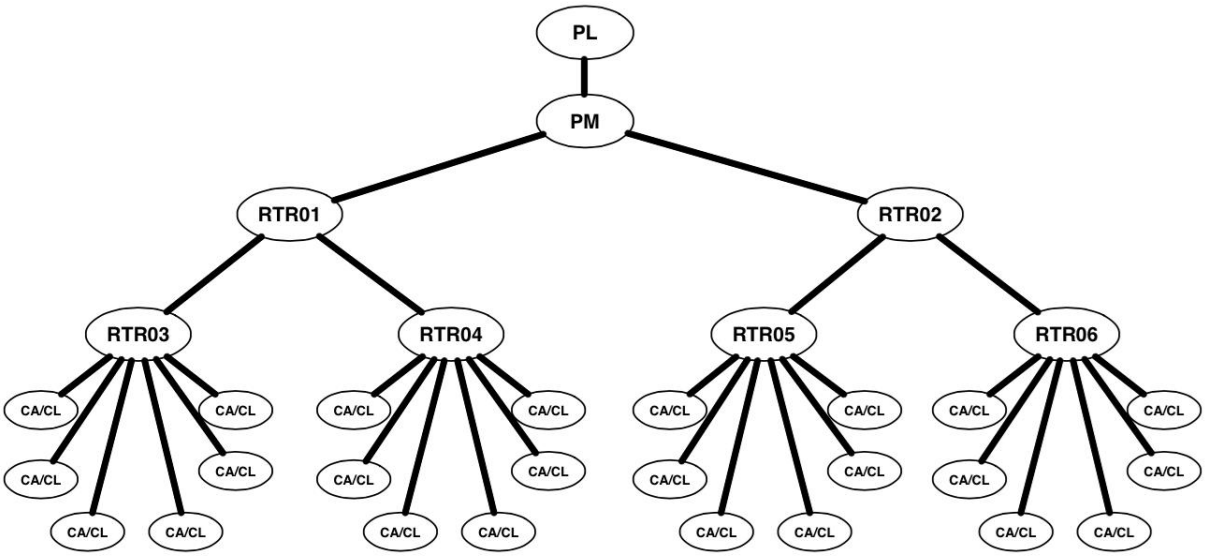
Os conteúdos são solicitados da seguinte forma. Os consumidores maliciosos enviam interesses para 12 conteúdos que são disponibilizados pelo produtor malicioso a taxas de 10, 100 e 500 interesses por segundo. Como cada pacote, seja de dados ou de interesse, tem tamanho máximo de 65.563 bytes [24, 15], cada consumidor malicioso, no pior caso, vai enviar interesses a uma taxa de 65 Kbps, 6,5 Mbps e 32,5 Mps respectivamente. Cada conteúdo malicioso possui 100 pedaços (*chunks*) e prefixos de nome diferentes. Os pedaços de conteúdos maliciosos são solicitados de acordo com o consumo sequencial, no qual um consumidor envia pacotes de interesse ordenados e de forma cíclica. Os consumidores legítimos sempre enviam 10 interesses/s para outros 12 conteúdos disponibilizados pelo(os) produtor(es) legítimo(s). Cada conteúdo legítimo possui 100 pedaços (*chunks*) e prefixos de nome diferentes. Os pedaços legítimos são solicitados seguindo uma distribuição Zipf com parâmetro $\alpha = 0,7$ [7].

O *cache* dos consumidores legítimos e dos roteadores tem capacidade para armazenar até 1000 pedaços de conteúdo, e cada pedaço possui 1024 bytes. Os consumidores maliciosos não possuem *cache* para potencializar o ataque, isto é, sempre enviam interesses, independentemente se já receberam o conteúdo anteriormente ou não. A PIT tem tamanho ilimitado para que seja possível avaliar apenas o efeito do aumento da ocupação maliciosa no *cache* dos nós. A política de substituição de *cache* é a *Least Recently Used* (LRU) e para todos os sub-*caches*.

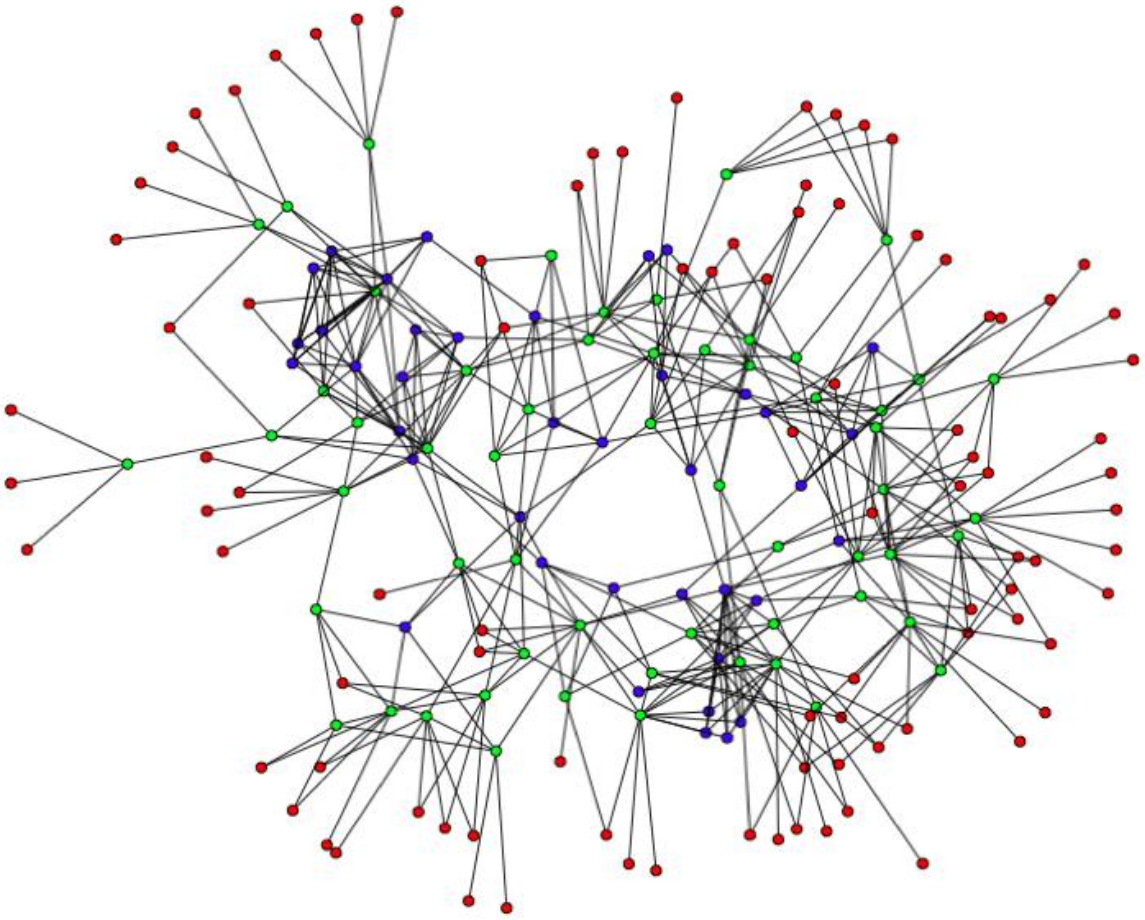
O módulo ndnSIM [3, 27] do simulador NS-3 é usado na avaliação. Para cada configuração, são realizadas 50 rodadas de simulação, cada uma com duração de 180 s. Para os pontos dos gráficos obtidos, são calculados intervalos de confiança representados por barras verticais para um nível de confiabilidade de 95%.

As métricas foram obtidas de acordo com com alterações feitas no código fonte do simulador e utilizando arquivos de registros (*tracers*) nativos do ndnSIM. De forma geral, as métricas foram obtidas da seguinte forma:

- Tempo médio de recuperação de conteúdos legítimos: obtido utilizando o *AppDelayTracer*, que torna possível obter dados sobre atrasos entre o envio do pacote de interesse e a recepção do pacote de dados equivalente;




(a) 32 nós.



(b) 192 nós.

Figura 6.1: Topologias utilizadas nas simulações.

- A ocupação maliciosa média do *cache* dos roteadores: obtida através de uma alteração no código do simulador que lista o CS do nó a cada segundo. Então faz-se

um percentual dos prefixos maliciosos, já que cada entrada no CS grava o prefixo do produtor;

- A taxa média de erros de *cache* dos conteúdos legítimos: obtidos com o *CsTracer*. O *tracer* gera uma saída indicando quando houve acerto ou erro do *cache* de determinado prefixo de conteúdo. Assim, é possível fazer um percentual de erro do *cache* em cada rodada de simulação;

- Percentual de conteúdos recuperados do produtor legítimo: obtida utilizando o *AggregateTracer* que torna possível o rastreamento do número agregado de pacotes de Interesses/Dados encaminhados por um nó CCN. Então, basta registrar os pacotes de dados que saem em resposta a um pacote de interesse no produtor legítimo em relação ao total de interesses gerados pelos consumidores legítimos.

Para as simulações, os parâmetros da função probabilística usada pelo CacheShield são $p = 20$ e $q = 1$. De acordo com os autores, esses valores são ideais para as topologias avaliadas [41].

## 6.2 Resultados

Os resultados apresentados têm como objetivo (i) avaliar a eficiência no ataque e (ii) avaliar e comparar a contramedida proposta Cache nFace com a arquitetura CCN padrão e com a contramedida CacheShield proposta por Xie *et al* [41].

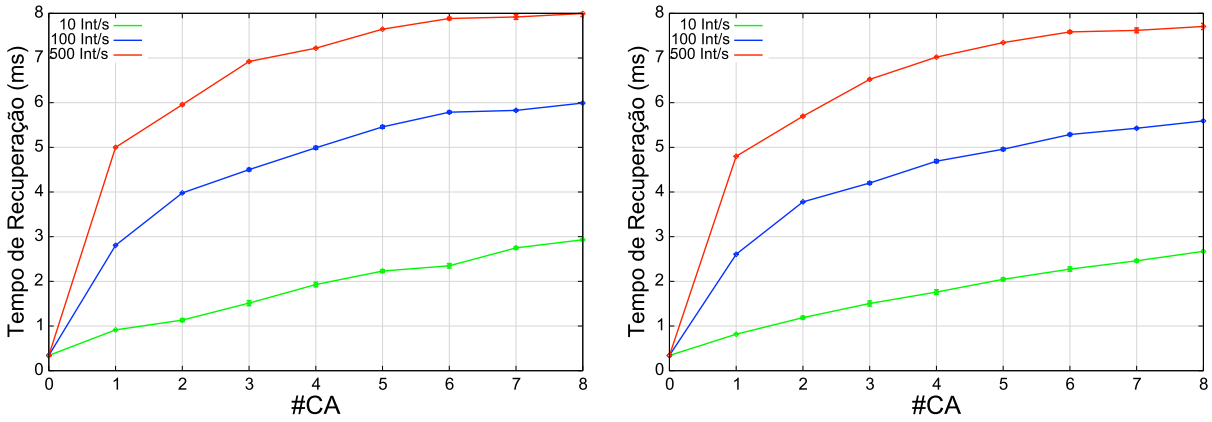
### 6.2.1 Avaliação do Ataque

Avalia-se, primeiramente, o ataque em conluio produtor-consumidor em relação às políticas de descarte do *cache* e do tamanho do *cache*. Para essa análise, utiliza-se apenas a topologia em árvore e os nós não implementam nenhuma contramedida. O objetivo é verificar qual política é mais efetiva sob ataque e qual o impacto do tamanho do *cache* dos nós na rede na efetividade do ataque.

#### 6.2.1.1 Políticas de Descarte do Cache

O primeiro objetivo da avaliação é verificar qual política de descarte é mais robusta ao ataque. Na topologia em árvore foram efetuadas rodadas de simulações comparando as políticas LRU e LFU para as métricas, tempo de recuperação de conteúdos legítimos

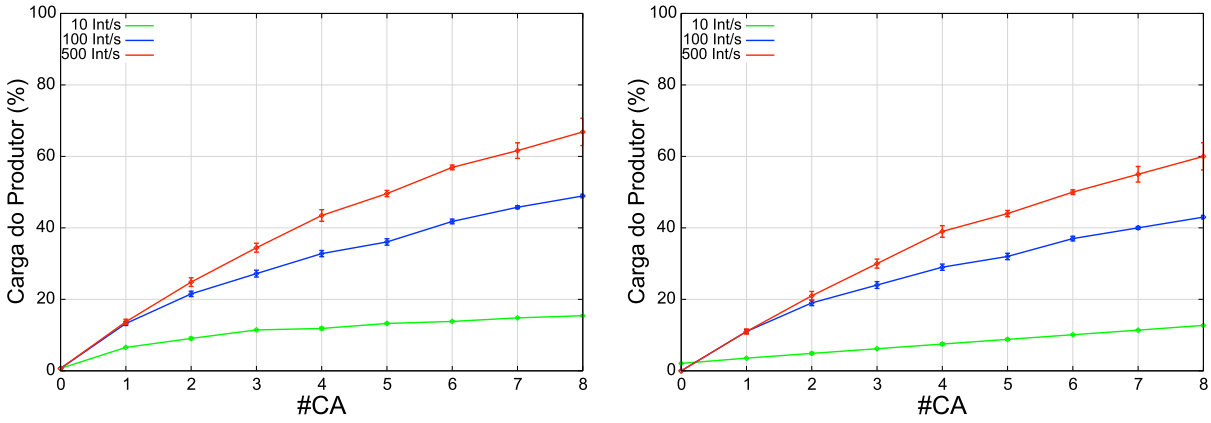
e percentual de carga do produtor. Foram escolhidas essas políticas porque o objetivo é avaliar uma política que testa popularidade (LFU) em comparação à nativa da CCN, que avalia o último uso (LRU). Todos os outros parâmetros de simulação são os apresentados na Seção 6.1. As Figuras 6.2(a) e 6.2(b) mostram o comportamento do tempo médio de recuperação de conteúdos legítimos em função do número de consumidores maliciosos para as duas políticas. Observa-se que, quanto mais consumidores maliciosos, maior o tempo médio de recuperação de conteúdos. Da mesma forma, quanto maior a taxa de interesses maliciosos, maior o tempo médio de recuperação de conteúdos legítimos. Conclui-se, portanto, que o ataque é efetivo nas configurações analisadas, independentemente da política de *cache* empregada pelos nós da rede. Mais explicações sobre o que torna o ataque efetivo são apresentadas na Seção 6.2.1.2. Pode-se observar que a política de *cache* tem influência na efetividade do ataque. Quando tem-se a política de consumo LFU, o tempo de recuperação cai em média 8% em relação à LRU. Isso deve-se ao fato do consumo LRU eliminar do *cache* os conteúdos não solicitados há mais tempo, o que leva o *cache* a remover os conteúdos legítimos com mais facilidade, visto que o conteúdo malicioso é solicitado de forma cíclica e em taxas de envio de interesses maiores. No caso da LFU, o conteúdo com menor frequência de uso é candidato ao descarte e, nesse caso, os conteúdos legítimos são solicitados com base em sua popularidade, o que dificulta mais a substituição dos conteúdos legítimos do *cache*. Observa-se, que mesmo no caso da LFU, a efetividade do ataque ainda afeta o consumo legítimo, pois apesar de mais robusto ao ataque, o consumo malicioso é cíclico e, assim, um conteúdo malicioso também é solicitado por outros nós e em uma frequência alta.




(a) LRU. (b) LFU.

Figura 6.2: O tempo de recuperação de conteúdos legítimos (LRU x LFU), árvore.

Nas Figuras 6.3(a) e 6.3(b) pode-se confirmar a maior robustez da política LFU em relação à LRU. Reduz-se o consumo direto do produtor legítimo em 10% se a política de




(a) LRU. (b) LFU.

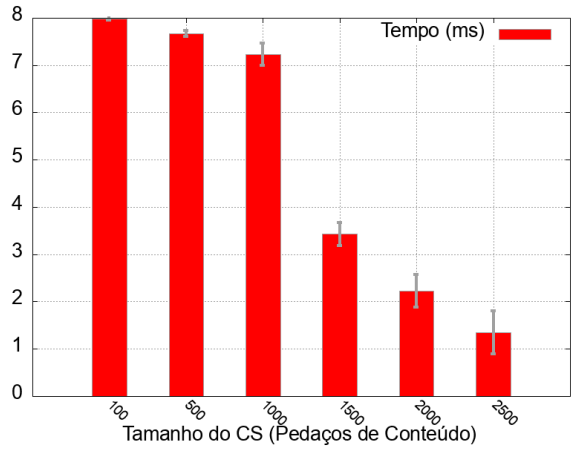
Figura 6.3: Percentual de carga do produtor (LRU x LFU), árvore.

descarte for LFU de maneira geral. Isso indica um maior número de acertos do *cache* nos roteadores de conteúdo e uma menor quantidade de saltos para recuperar o conteúdo. Resta ressaltar que tal robustez não é suficiente para impedir a eficácia do ataque, pois ainda assim, como abordado nos parágrafos anteriores, a quantidade de saltos estimada pelo tempo de recuperação, bem como o consumo direto do produtor ainda são elevados. Então essa comparação serve apenas para mostrar que as políticas baseadas na popularidade do conteúdo são mais eficazes que as baseadas no último uso para reduzir a efetividade do ataque. Entretanto, a política padrão da CCN é a LRU e é essa política de substituição que é considerada em todas as próximas avaliações.

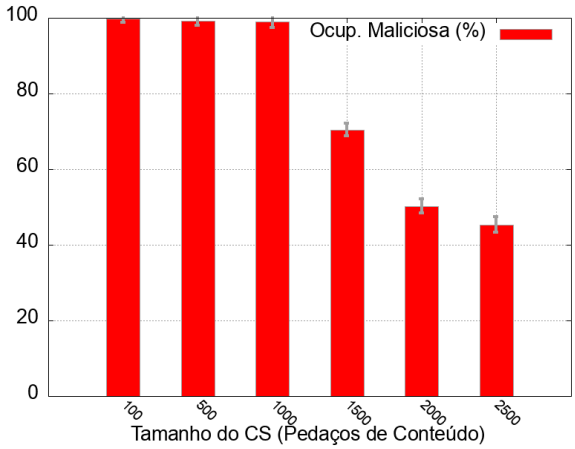
#### 6.2.1.2 Tamanho do Cache

A Figura 6.4 mostra os resultados das avaliações das quatro métricas de desempenho para diferentes tamanhos de CS (100, 500, 1000, 1500, 2000 e 2500 pedaços de conteúdos). Considera-se a topologia em árvore com 8 consumidores maliciosos operando a 500 interesses/s. Todos os outros parâmetros de simulação são os apresentados na Seção 6.1. Observa-se que quando se tem um *cache* grande o suficiente para comportar todos os dados legítimos e maliciosos da rede o ataque é ineficaz, como nos casos em que o CS tem tamanho 2000 e 2500 pedaços de conteúdos. Isso ocorre porque, nesse caso, não haverá descarte. Porém, basta ter concorrência de recursos para que o ataque se torne efetivo como observado pelas Figuras 6.4(b) e 6.4(c). Com um CS de tamanho 100 pedaços de conteúdos, que representa menos de 5% da quantidade total de conteúdos disponíveis tem-se ocupação maliciosa e taxa de erro altas, na ordem de 99%. Isso se repete para CS de tamanho 500 e 1000 pedaços de conteúdos. Por exemplo, com CS de tamanho 1000 pedaços de conteúdos tem-se, praticamente, 99% de ocupação maliciosa e 99% de taxa de

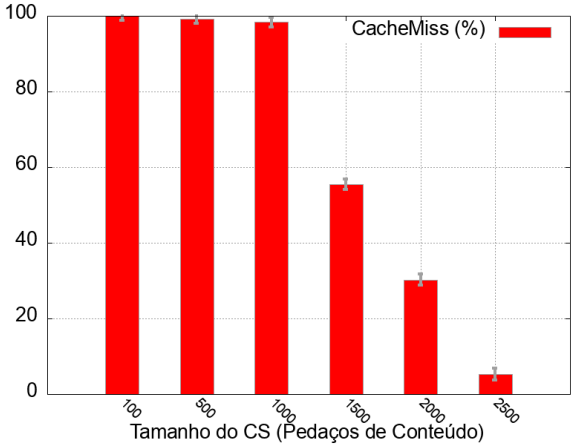
erro do *cache*. Isso implica que basta ter um *cache* que seja menor que a quantidade de conteúdos maliciosos solicitados para que esses removam os conteúdos legítimos do *cache*. Em altas taxas os pacotes maliciosos removem os pacotes legítimos do CS sempre que há concorrência de recurso.



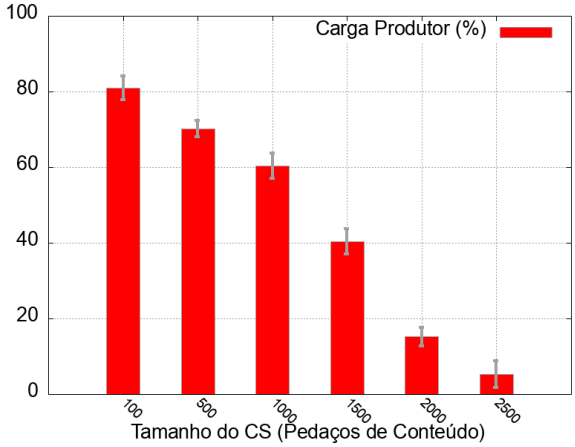

(a) Tempo de recuperação de conteúdos legítimos, árvore.




(b) Ocupação Maliciosa do Cache, árvore.




(c) Taxa de Erro do Cache, árvore.




(d) Carga no Produtor Legítimo, árvore.

Figura 6.4: Métricas em relação ao tamanho do CS, topologia em árvore.

Outra observação importante é que a alta taxa de ocupação maliciosa e alta taxa de erro de *cache* reflete no tempo de recuperação e na carga do produtor legítimo, como mostram as Figuras 6.4(a) e 6.4(d). No caso de acontecer concorrência de uso do *cache*, como nos tamanhos 100, 500 e 1000 conteúdos, essas métricas estão sempre elevadas. Porém, em cenários menos concorridos (1500, 2000 e 2500 pedaços conteúdos) que tem-se valores compatíveis com a rede sem ataque. Não pode-se prever quantos atacantes, quantos conteúdos e em que taxas esses operam, então é necessário avaliar o ataque em cenários onde há concorrência de recursos de *cache*. Em seu trabalho Ghodsi *et al.* [19], sugerem que uma ROC deve estar preparada para lidar com pelo menos $10^{12}$ objetos com

base no tamanho atual da web. E esta é uma estimativa extremamente conservadora, segundo os autores.

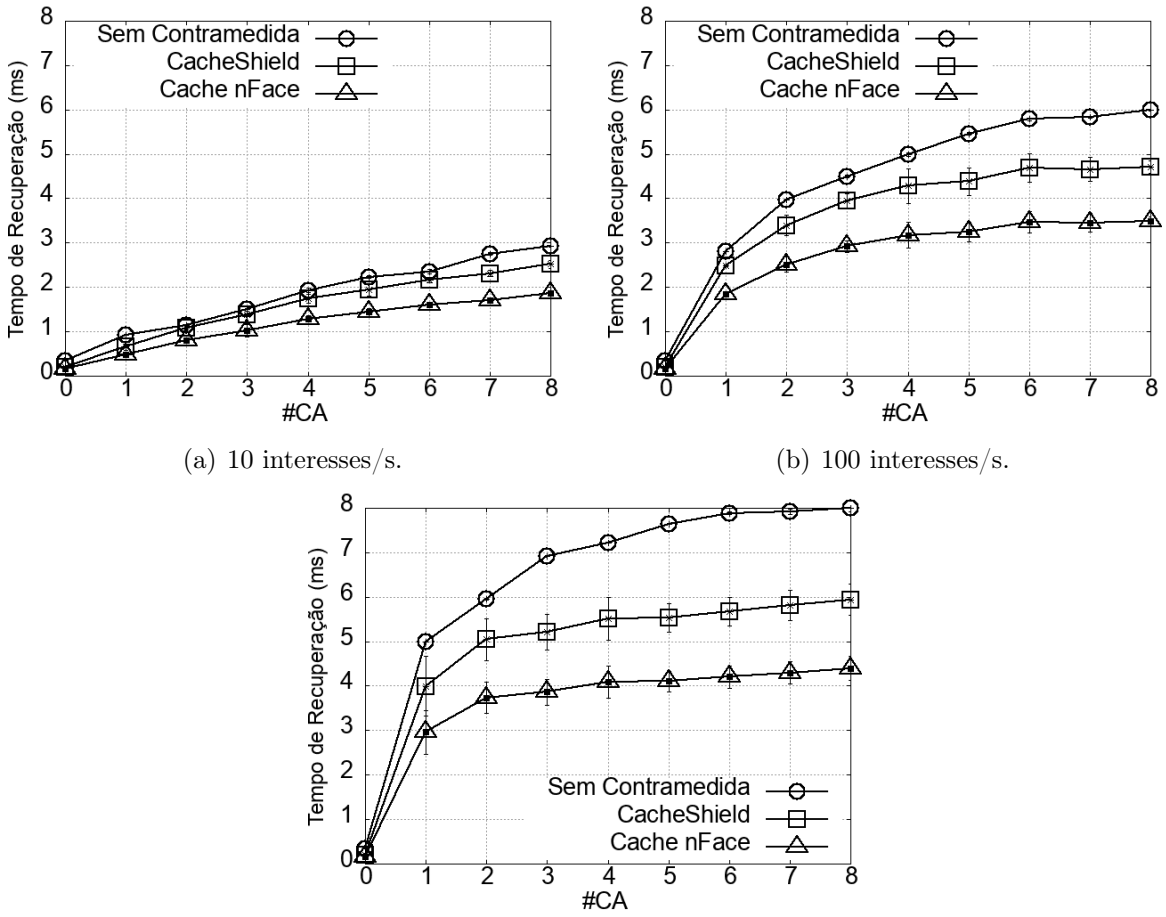
### 6.2.2 Avaliação das Contramedidas

O objetivo das seções seguintes é avaliar as contramedidas Cache nFace e CacheShield nas topologias em árvore e em malha e para as quatro métricas de desempenho, variando o número de consumidores maliciosos e a taxa com que esses nós enviam interesses maliciosos. Os parâmetros de simulação são os apresentados na Seção 6.1. Os resultados estão divididos de acordo com a métrica de desempenho avaliada.

#### 6.2.2.1 Tempo Médio de Recuperação de Conteúdos Legítimos

As Figuras 6.5 e 6.6 mostram o comportamento do tempo médio de recuperação de conteúdos legítimos em função do número de consumidores maliciosos, sendo que as Figuras 6.5(a), 6.5(b) e 6.5(c) referem-se à topologia em árvore, enquanto as Figuras 6.6(a), 6.6(b) e 6.6(c) referem-se à topologia em malha. O comportamento observado é o mesmo: quanto mais consumidores maliciosos, maior o tempo médio de recuperação de conteúdos. Da mesma forma, quanto maior a taxa de interesses maliciosos, maior o tempo médio de recuperação de conteúdos legítimos. Para a topologia em árvore observa-se que o mecanismo CacheShield (Figura 6.5(c)), no pior caso em que atacantes enviam interesses a 500 interesses/s, é capaz de reduzir o tempo de recuperação de conteúdos em 2 ms quando comparado com a rede sob-ataque e sem uso de nenhuma contramedida. Isso representa uma redução de 25% na eficiência do ataque. Observa-se que o mecanismo Cache nFace, por sua vez, reduz em quase 50% a eficiência do ataque, se comparado com a rede sem contramedida, e em cerca de 35%, quando atacantes operam a 500 interesses/s, em relação ao CacheShield. É importante ressaltar que como os consumidores legítimos possuem *cache* e como eles sempre consomem de acordo com a popularidade, pode-se observar que sem ataque o consumo é, muitas vezes, feito do próprio *cache* do nó, o que resulta em um tempo de recuperação inferior a 1 ms.

A efetividade do ataque pode ser observada também na topologia em malha. Quando não há contramedida na rede, Figuras 6.6(a), 6.6(b), 6.6(c), tem-se um tempo médio de recuperação de conteúdos 2 vezes maior quando tem-se 24 consumidores maliciosos enviando 10 interesses/s, 12 vezes maior quando enviam 100 interesses/s e 16 vezes maior quando operam a 500 interesses/s, se comparado com o Cache nFace em mesmas condições. Outro resultado interessante para topologia em malha, quando atacantes operam

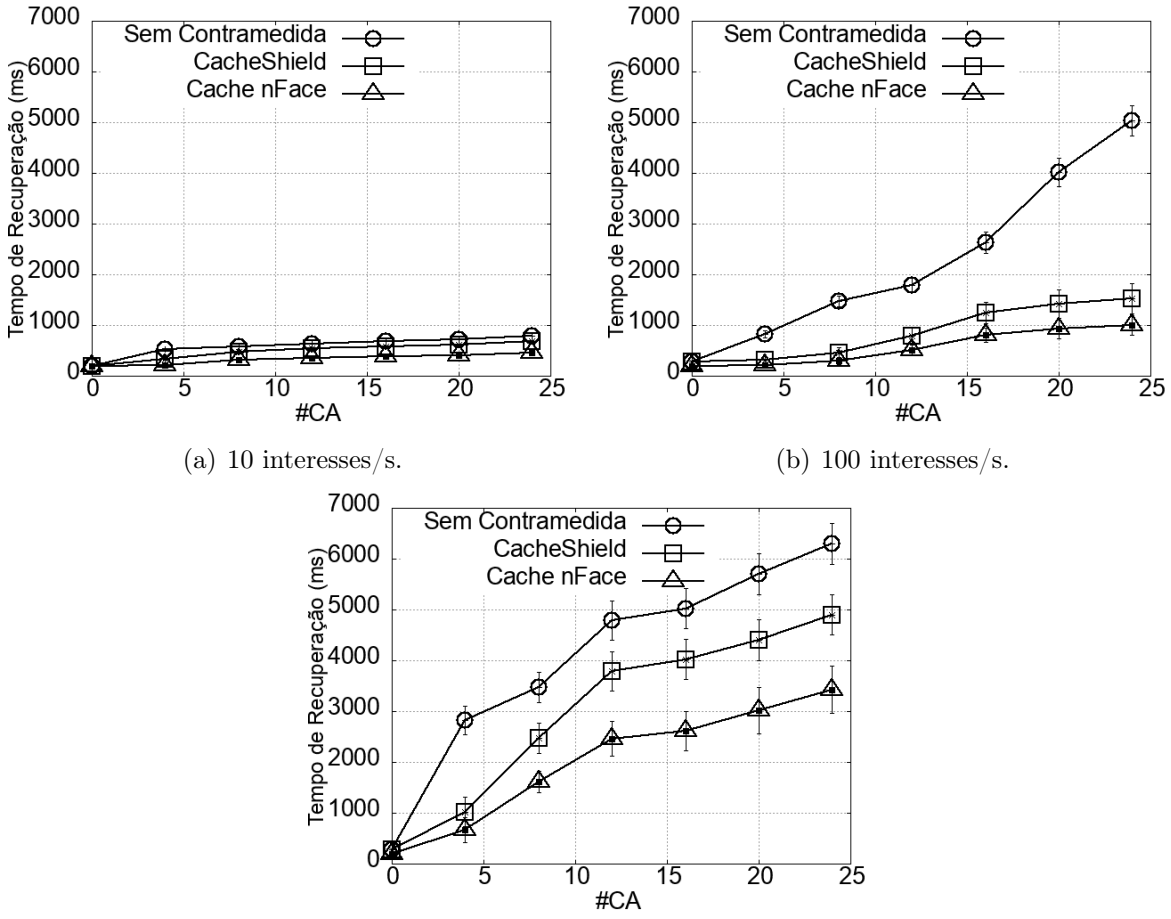



(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.5: O tempo de recuperação de conteúdos legítimos na topologia em árvore.




(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.6: O tempo de recuperação de conteúdos legítimos na topologia em malha.

a 500 interesses/s, é que o mecanismo Cache nFace (Figura 6.6(c)) é cerca de 35% mais eficaz que o CacheShield, e ainda reduz, em cerca de 50%, a eficiência do ataque quando comparado com a rede sem contramedida.

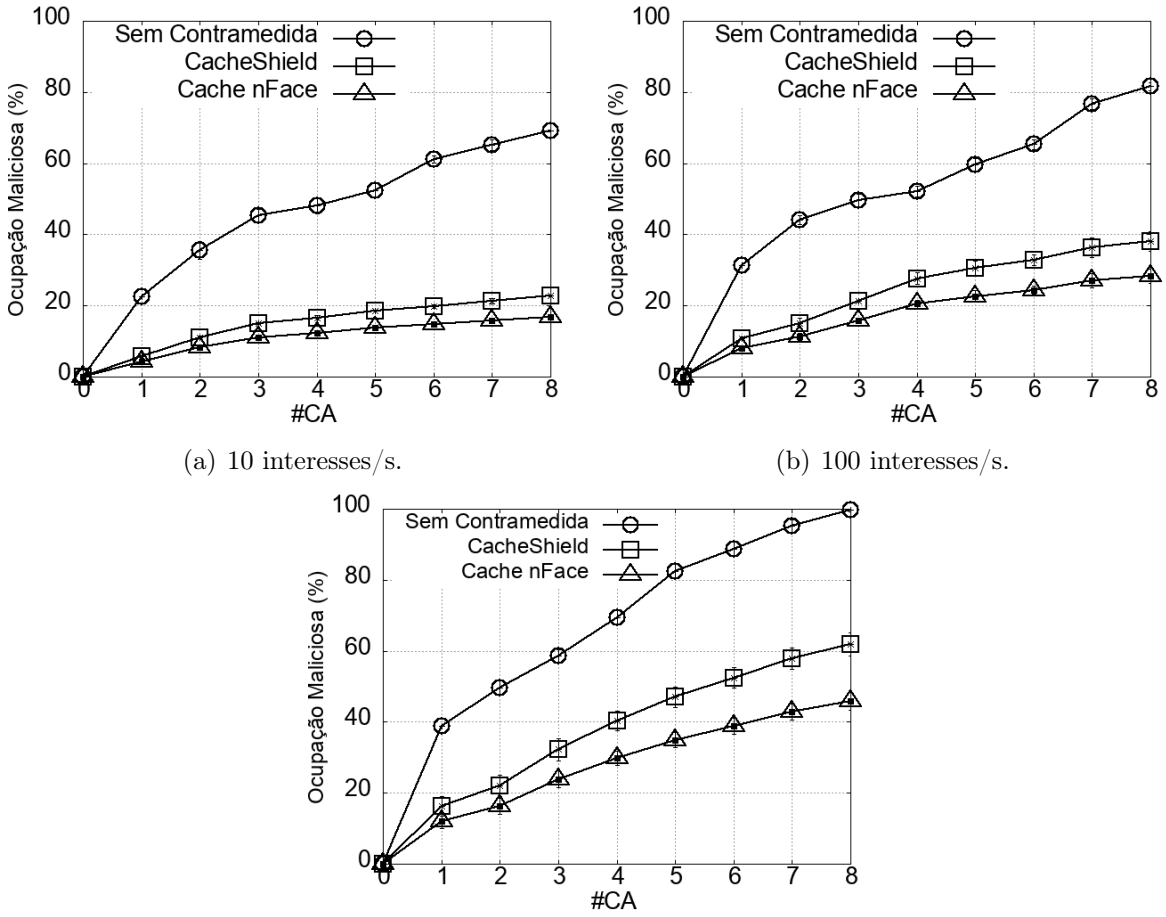
#### 6.2.2.2 Ocupação Maliciosa dos Caches e Taxa de Erro do Cache

O motivo da contramedida Cache nFace ser mais eficiente é que ela limita a ocupação maliciosa dos *caches* dos roteadores e, consequentemente, reduz a taxa de erro de *cache* dos conteúdos legítimos se comparada ao CacheShield. Esses resultados são mostrados pelas Figuras 6.7, 6.8, 6.9 e 6.10. Na topologia em árvore, enquanto tem-se na curva sem ataque da Figura 6.5(b) o tempo de recuperação de 6 ms para taxa de 100 interesses/s com 8 consumidores maliciosos, tem-se ocupação maliciosa e taxa de erro do *cache* na ordem de 80% (Figura 6.9(b)). Essa característica implica que, sob ataque, os consumidores devem dar mais saltos para recuperar o conteúdo solicitado. Como, na topologia em árvore, o atraso de cada enlace é de 1 ms, conclui-se que os conteúdos legítimos são recuperados mais frequentemente de nós que estão a mais saltos do consumidor do que os nós de borda. Isso indica que os roteadores de borda estão com uma alta ocupação de conteúdos maliciosos em seu *cache*.

Os mecanismos avaliados reduzem para, aproximadamente 5 ms com o CacheShield e 3 ms com o Cache nFace, com 8 consumidores maliciosos enviando 100 interesses/s. Isso indica que com CacheShield, na topologia em árvore, faz-se em média 3 saltos para recuperar o conteúdo, enquanto com Cache nFace são necessários em média 2 saltos. Quando a rede está sob ataque e não utiliza nenhum dos mecanismos estudados, operando na mesma taxa e quantidade de consumidores do exemplo anterior, tem-se em média 4 saltos. Deste modo, pode-se dizer que o Cache nFace reduz pela metade o tempo de recuperação de conteúdos legítimos nesse cenário, reduzindo a eficiência do ataque.

Na topologia em malha, Figuras 6.8(a), 6.8(b) e 6.8(c), observa-se uma ocupação maliciosa de cerca de 20% quando existem na rede 20 consumidores maliciosos enviando 500 interesses/s. Portanto, na topologia em malha, se cerca de 10% da rede é composta de consumidores maliciosos, a ocupação maliciosa dos *caches* do roteadores de conteúdo já compromete o consumo de conteúdo legítimo. Por outro lado, o mecanismo CacheShield pode reduzir em cerca de 5 pontos percentuais e o mecanismo Cache nFace em 10 pontos percentuais essa ocupação maliciosa, como se observa na Figura 6.8(b).

Na topologia de em malha, observa-se uma ocupação maliciosa e taxa de erro do *cache* menor que na topologia em árvore. Isso deve-se ao fato que a topologia em ár-

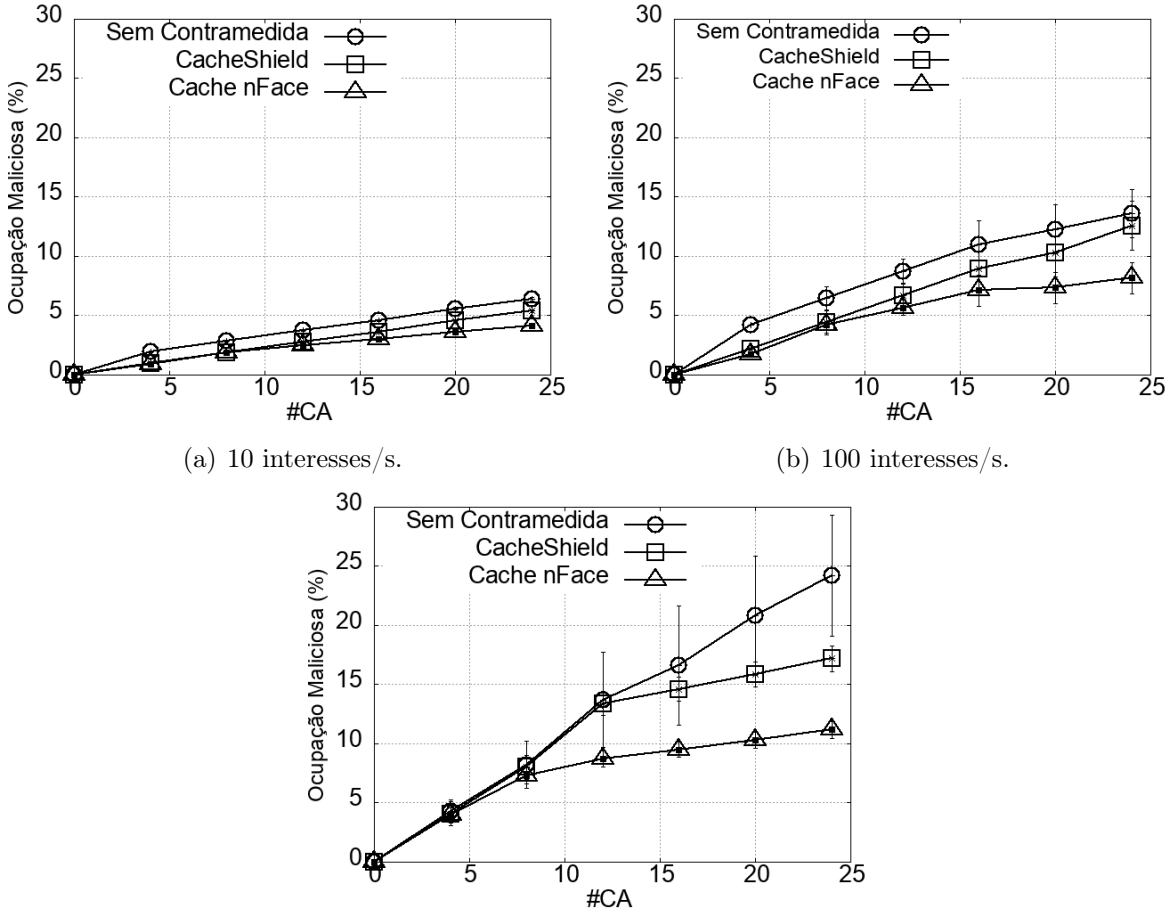



(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.7: O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos para as contramedidas avaliadas na topologia em árvore.




(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

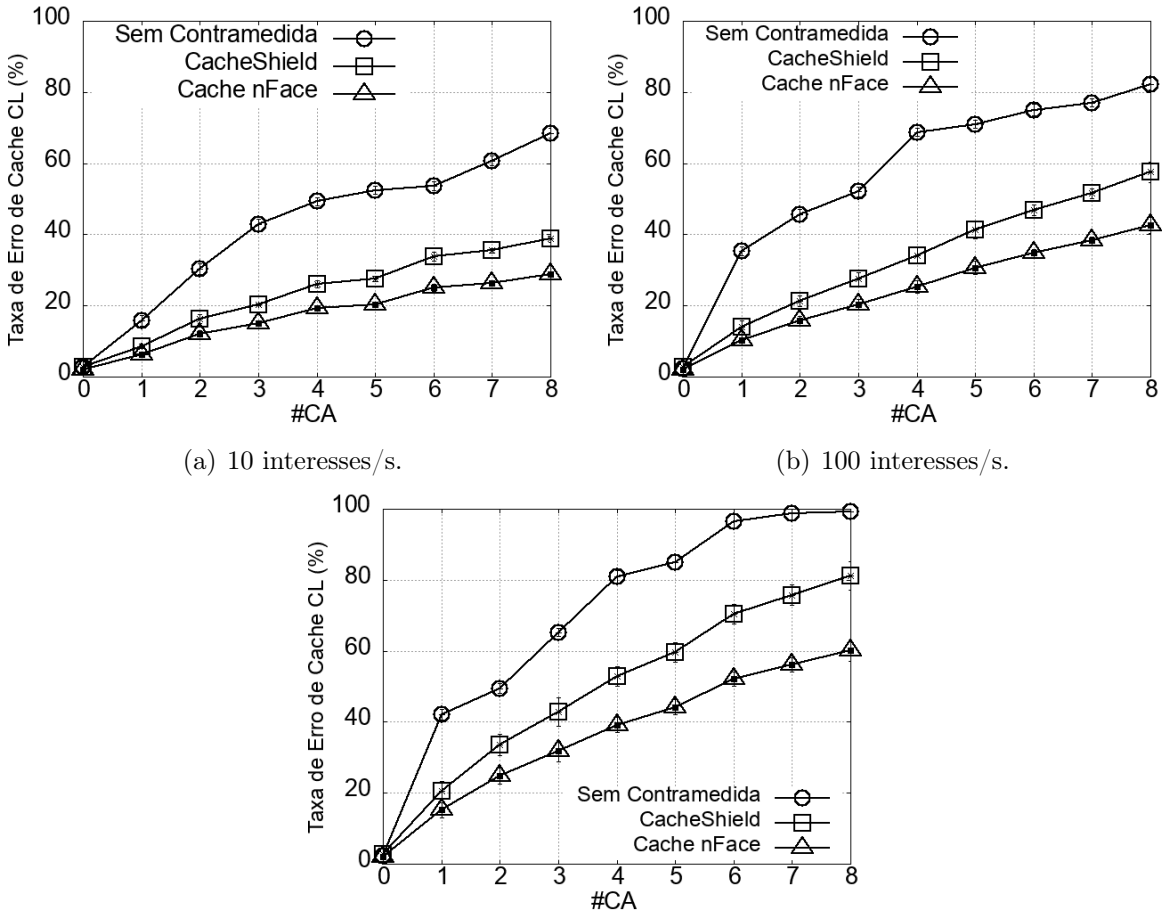
Figura 6.8: O percentual de ocupação do *cache* dos roteadores por conteúdos maliciosos para as contramedidas avaliadas na topologia em malha.

vore tem a característica de ter o produtor malicioso no caminho do produtor legítimo, obrigando, assim, um consumidor legítimo a utilizar o *cache* de, pelo menos, um nó comprometido para recuperar um conteúdo legítimo. Ainda é possível observar uma maior eficiência do ataque em comprometer o consumo de conteúdos legítimos quando se tem 24 consumidores maliciosos enviando 500 interesses/s. Nessa curva, observa-se uma elevada ocupação maliciosa e taxa de erro do *cache*, como mostrado nas Figuras 6.8(c) e 6.10(c) em ordens de 25% e 42% respectivamente. Pode-se concluir que com 24 consumidores maliciosos, que representam 12,65% do total de nós, pode-se aumentar a taxa de erro do *cache*, comprometendo o consumo de conteúdos legítimos. Os mecanismos estudados, porém, reduzem a eficiência do ataque. O CacheShield pode, por exemplo, reduzir a taxa de erro do cache (Figura 6.10(b)) para 100 interesses/s com 15 consumidores maliciosos em cerca de 40%, enquanto o Cache nFace, na pior configuração (500 interesses/s para 24 consumidores maliciosos) pode reduzir a ocupação maliciosa (Figura 6.8(c)) também em cerca de 40%. Ou seja, em taxas de 500 interesses/s com 24 consumidores maliciosos, o mecanismo Cache nFace tem a mesma eficiência que o CacheShield em taxa inferior de 100 interesses/s com 24 consumidores maliciosos, se mostrando mais eficaz para reduzir os efeitos do ataque nos cenários avaliados.

#### 6.2.2.3 Conteúdos Recuperados do Produtor Legítimo

As Figuras 6.11 e 6.12 mostram o percentual de conteúdos legítimos recuperados do produtor legítimo em função do número de consumidores maliciosos e da taxa de envio de interesses por esses nós. Esses resultados corroboram que o ataque em conluio reduz e eficiência do emprego do *cache* pela CCN. A Figuras 6.11(a), 6.11(b) e 6.11(c) mostram que, se não há ataque, cerca de 0,5% dos conteúdos solicitados são recuperados diretamente do produtor original. Nesse caso, cada conteúdo legítimo é recuperado do produtor original no máximo duas vezes, até que seja armazenado pelos nós RTR1 e RTR2. Porém, basta se ter 4 consumidores maliciosos enviando 10 interesses/s (Figura 6.11(a)) para que esse valor aumente para cerca de 12%. No pior caso, os consumidores legítimos estão recuperando cerca de 67% dos conteúdos legítimos diretamente do produtor legítimo na topologia em árvore. Com os mecanismos Cacheshield e Cache nFace tem-se redução de carga no produtor legítimo. Isso deve-se ao fato de que roteadores de conteúdo estarem com menos ocupação maliciosa, fazendo aumentar a taxa de acerto do *cache* e recuperando conteúdos legítimos em menos saltos.

É interessante observar que, sob ataque, apenas 4 atacantes, enviando 100 interes-

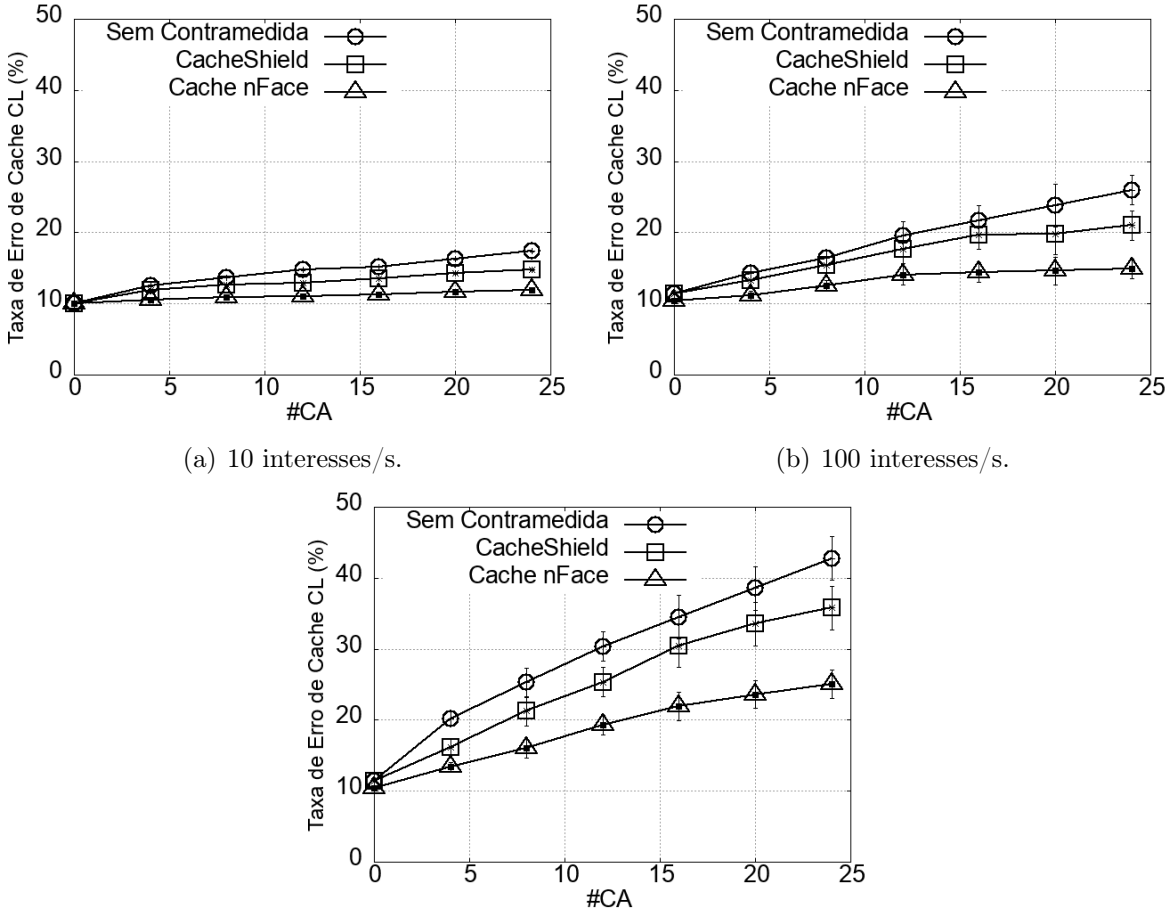



(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.9: A taxa de erros de *cache* para os conteúdos legítimos para as contramedidas avaliadas na topologia em árvore.

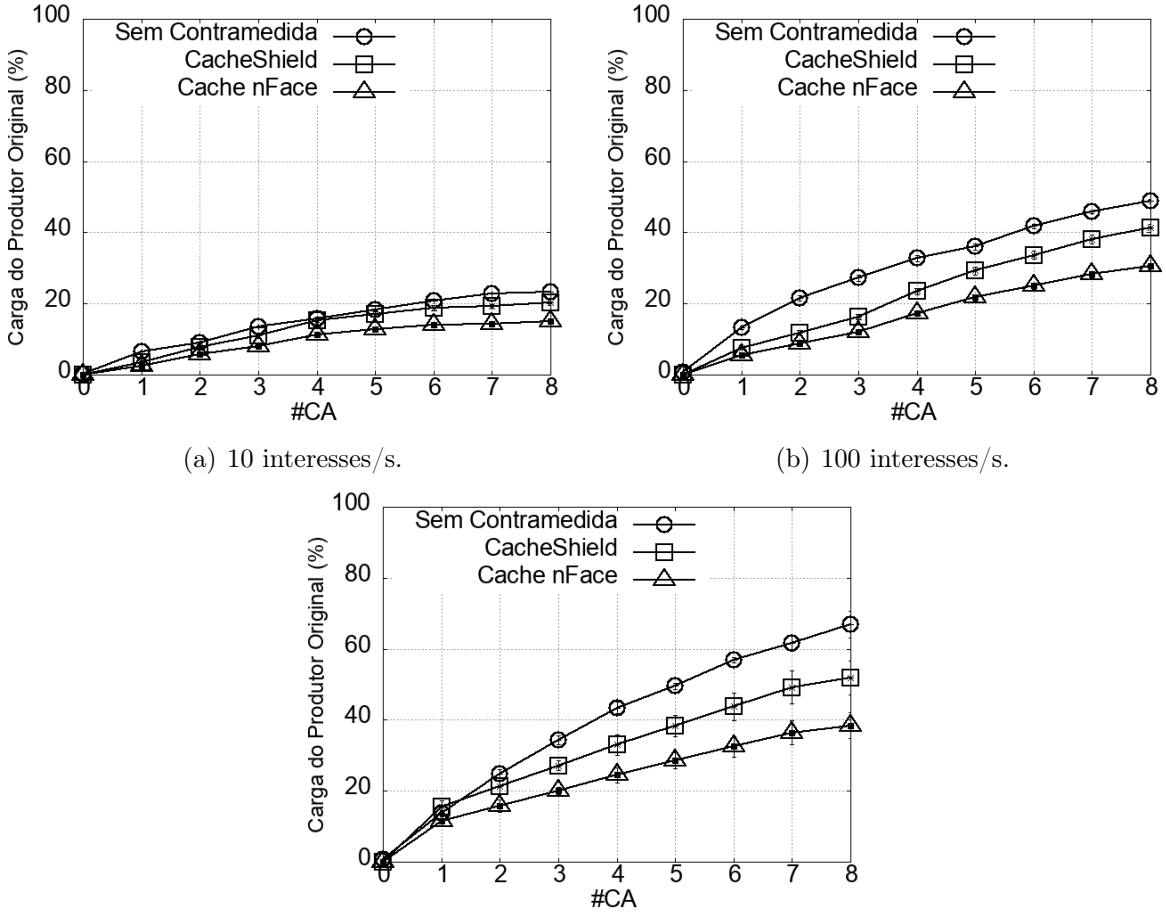



(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.10: A taxa de erros de *cache* para os conteúdos legítimos para as contramedidas avaliadas na topologia em malha.

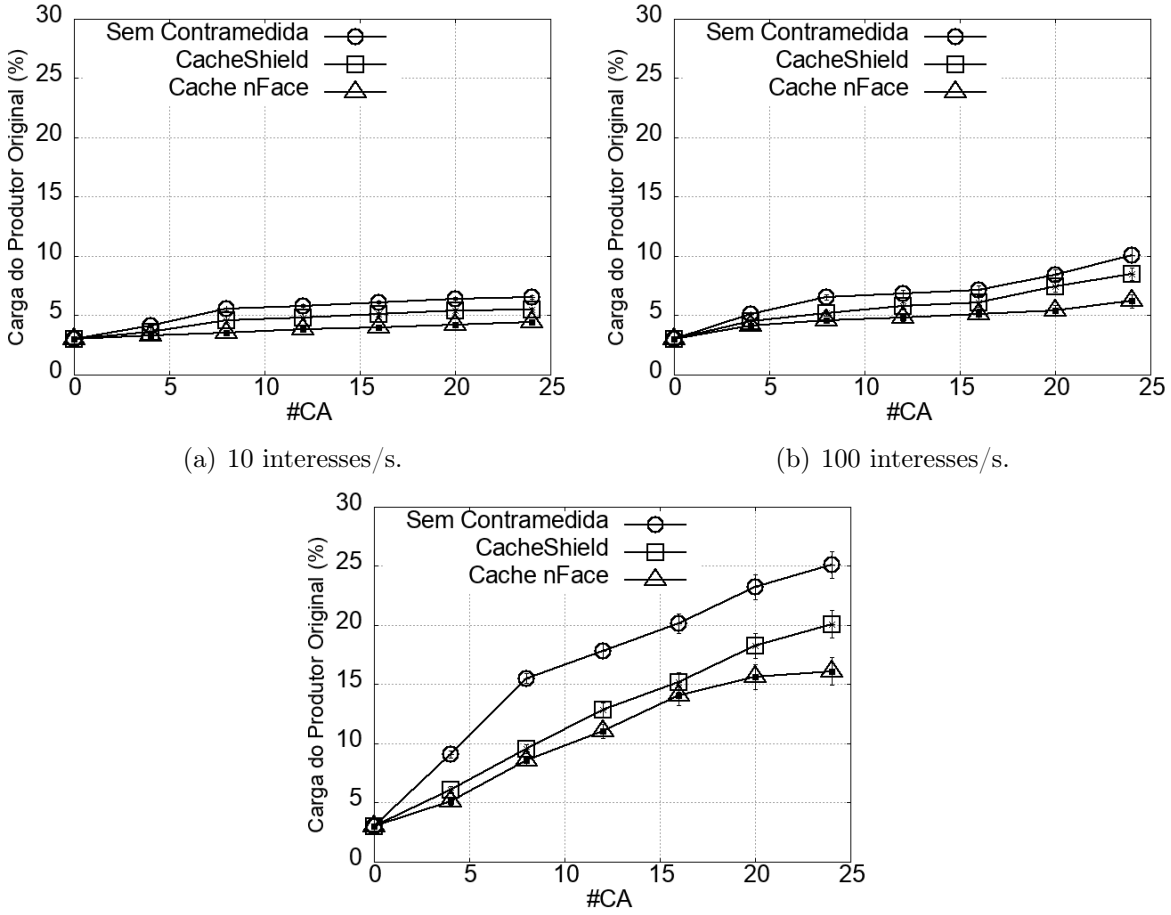



(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.11: Percentual de carga do produtor para as contramedidas avaliadas na topologia em árvore.




(a) 10 interesses/s.

(b) 100 interesses/s.

(c) 500 interesses/s.

Figura 6.12: Percentual de carga do produtor para as contramedidas avaliadas na topologia em malha.

ses/s, elevam a carga do produtor a cerca de 38% (Figura 6.11(b)), com CacheShield nas mesmas configurações tem-se apenas 22%. Para o mecanismo Cache nFace, nas mesmas configurações, observa-se que os mesmos 4 atacantes deixam a carga no produtor legítimo em 19%. O mecanismo Cache nFace mostra-se mais eficiente em reduzir o tempo de recuperação, a ocupação maliciosa, a taxa de erro do *cache* e a carga no produtor que o CacheShield. As características observadas na topologia em árvore podem ser observadas também na topologia em malha para o consumo direto do produtor legítimo, Figuras 6.12(a), 6.12(b) e 6.12(c). Na topologia em malha, ainda tem-se no pior caso, no qual 24 consumidores maliciosos enviam 500 interesses/s, uma carga do produtor legítimo de cerca de 25% sem o uso das contramedidas. Assim, como os pacotes de dados são recuperados diretamente do produtor, a eficiência da rede em entregar conteúdos legítimos mais próximos do consumidor é comprometida. Observa-se, ainda, que o mecanismo CacheShield reduz a carga produtor em cerca de 20%, enquanto o mecanismo Cache nFace reduz a carga do produtor em cerca de 45% nas configurações avaliadas.

Uma característica relevante para a avaliação é que o Cache nFace divide o CS do nó em sub-*caches*. Essa característica é importante porque se um determinado nó, que conta com quatro interfaces de rede, por exemplo, usar o Cache nFace, terá o CS divido em quatro sub-*caches*, que tomam decisões sobre armazenar um conteúdo de forma independente. Isso dificulta a ação do atacante, pois seria necessário afetar todos os quatro sub-*caches* para o nó negar serviço à rede. O funcionamento do Cache nFace impede que um sub-*cache* seja poluído por outro do mesmo nó. Nas topologias estudadas, o mecanismo Cache nFace se mostrou mais eficaz que o CacheShield em todos os cenários avaliados.

# Capítulo 7

# Conclusão

Nesse trabalho foi proposta uma contramedida chamada de Cache nFace. O funcionamento do mecanismo baseia-se em criar sub-*caches* em um nó, a partir da divisão do espaço total de armazenamento em *cache* desse nó. Cada sub-*cache* está associado a uma das interfaces de rede do nó. O sub-*cache* tem seu tamanho definido de acordo com a taxa de transmissão nominal da interface. Essa abordagem mostrou-se eficiente em todos os cenários estudados, inclusive tendo desempenho superior ao mecanismo CacheShield, proposto na literatura. Também avaliou-se o ataque negação de serviço em conluio consumidor-produtor para a arquitetura CCN. Esse ataque visa aumentar o tempo de recuperação de conteúdos, aumentando a ocupação do *cache* dos nós intermediários com conteúdos maliciosos.

## 7.1 Resultados Obtidos

Por meio de simulações comprovou-se que o mecanismo Cache nFace é superior ao mecanismo CacheShield em todas as avaliações feitas. Diferentes configurações foram usadas nas simulações, variando-se o número de consumidores maliciosos, a política de consumo desses consumidores, o tamanho do *cache* e a taxa de pacotes de interesse maliciosos.

Os resultados mostram que o ataque em conluio é efetivo, o que compromete o emprego do *cache* pela CCN. No pior caso, o tempo de recuperação aumentou 23,5 vezes para as configurações avaliadas. Esse aumento se deve a uma ocupação maliciosa média de 99% e, consequentemente, a uma taxa de erro de *cache* de 99%. Com isso, os consumidores legítimos recuperaram 67% dos conteúdos solicitados diretamente do produtor. Mostra-se, também, que a distribuição de consumidores maliciosos é mais efetiva do que o aumento da taxa agregada de envio de interesses maliciosos. Também foi realizada uma avaliação

de políticas de descarte de *cache*, LRU e LFU, para verificar qual se mostra mais robusta ao ataque.

Com relação à comparação dos mecanismos Cache nFace e CacheShield, duas topologias foram usadas nas simulações, uma em árvore de 32 nós e outra em malha de 192 nós. A contramedida Cache nFace reduz em cerca de 50% a efetividade do ataque e se mostra mais eficaz nos cenários avaliados que o CacheShield em cerca de 30% de forma geral. A contramedida Cache nFace é capaz de comedir o ataque nas configurações avaliadas, reduzindo sua eficiência na topologia malha e na topologia em árvore.

## 7.2 Limitações da Proposta

O mecanismo Cache nFace, com a divisão do CS do nó em sub-*caches* não reage ao ataque, ele apenas faz uma contenção desse ataque de modo a reduzir danos a rede. Essa contenção se mostrou eficiente nos cenários e topologias avaliadas. Apesar do ataque estar em andamento, nos casos avaliados, ele não surte muito efeito. Porém, em topologias com mais caminhos alternativos, como uma grade, ou com adversários móveis, o mecanismo pode não reunir condições suficientes para inibir o ataque. Nesse caso uma solução que consiga reagir ao ataque pode ser uma solução mais indicada.

A característica de armazenar o conteúdo no sub-*cache* associado a interface que primeiramente recebe o interesse pode trazer alguns problemas. Por exemplo, quando existir um conjunto de interesses legítimos associados a uma interface, o sub-*cache* dessa interface pode rapidamente ser ocupado. Mesmo que os demais sub-*cache* tenham condições de armazenar esses conteúdos, pela forma de funcionamento do Cache nFace, eles serão descartados.

## 7.3 Contribuições

Este trabalho propôs uma contramedida para o ataque em conluio produtor-consumidor numa rede CCN. Foram feitas avaliações do ataque com relação às políticas de descarte LRU e LFU e do tamanho do *cache*. O mecanismo proposto foi comparado com outro existente na literatura, utilizando as métricas de desempenho: tempo médio de recuperação de conteúdos legítimos, a ocupação maliciosa média do *cache* dos roteadores, a taxa média de erros de *cache* dos conteúdos legítimos e o percentual de conteúdos legítimos recuperados do produtor legítimo.

Chegou-se à conclusão que o ataque em conluio produtor-consumidor é eficiente em aumentar o tempo de recuperação de conteúdos legítimos, e pode levar a um alto índice de ocupação maliciosa nos *caches* dos roteadores de conteúdo. Observou-se que as políticas de descarte baseadas na popularidade são mais eficientes que as baseadas no último uso para comedir o ataque, porém não são suficientes para contê-lo. Ainda foi constatado, variando o tamanho do *cache*, que quando há concorrência de recursos, pacotes maliciosos podem expulsar pacotes legítimos do CS do nó. Por fim, a contramedida proposta é eficaz em conter a eficiência do ataque, e mostrou-se mais eficiente em todos os cenários avaliados que a existente na literatura.

## 7.4 Trabalhos Futuros

Como trabalhos futuros pretende-se investigar outros cenários e definir novas métricas e configurações para o mecanismo proposto. Uma nova configuração, onde produtor malicioso e consumidor maliciosos tenham papeis acumulados, produzindo e consumido ao mesmo tempo, será objeto de trabalho futuro. Pretende-se também avaliar o efeito de aumentar o número de produtores maliciosos para aumentar a chance de ter mais do que uma interface de rede sendo atacada. Ou, ainda, avaliar quantos sub-*caches* de um nó possuem conteúdos maliciosos. Por fim, testar o mecanismo proposto com outros tipos de ataque ao *cache* e em outras topologias.

# Referências


[1] AbdAllah, E.; Hassanein, H.; Zulkernine, M. A survey of security attacks in information-centric networking. *Communications Surveys Tutorials, IEEE 17*, 3 (third quarter 2015), 1441–1454.

[2] Afanasyev, A.; Mahadevan, P.; Moiseenko, I.; Uzun, E.; Zhang, L. Interest flooding attack and countermeasures in named data networking. In *IFIP Networking* (May 2013), pp. 1–9.

[3] Afanasyev, A.; Moiseenko, I.; Zhang, L. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN, October 2012.

[4] Arianfar, S.; Koponen, T.; Raghavan, B.; Shenker, S. On preserving privacy in content-oriented networks. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking* (New York, NY, USA, 2011), ICN '11, ACM, pp. 19–24.

[5] Baugher, M.; Davie, B.; Narayanan, A.; Oran, D. R. Self-verifying names for read-only named data. In *Workshop on Emerging Design Choices in Name-Oriented Networking - NOMEN* (Mar. 2012), pp. 274–279.

[6] Bernardini, C.; Silverston, T.; Festor, O. Mpc: Popularity-based caching strategy for content centric networks. In *2013 IEEE International Conference on Communications (ICC)* (June 2013), pp. 3619–3623.

[7] Breslau, L.; Cao, P.; Fan, L.; Phillips, G.; Shenker, S. Web caching and zipf-like distributions: evidence and implications. In *IEEE Conference on Computer Communications - INFOCOM* (Mar. 1999), pp. 126–134.

[8] Brito, G. M.; Velloso, P. B.; Moraes, I. M. Redes orientadas a conteúdo: Um novo paradigma para a Internet. In *Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC* (2012), pp. 211–264.

[9] Brito, G. M.; Velloso, P. B.; Moraes, I. M. *Information-Centric Networks, A New Paradigm for the Internet*, 1 ed. FOCUS - Networks and Telecommunications Series. Wiley-ISTE, 2013.

[10] Choi, S.; Kim, K.; Kim, S.; Roh, B. Threat of DoS by interest flooding attack in content-centric networking. In *Information Networking International Conference* (Jan. 2013), pp. 315–319.

[11] Cieza, E. G.; Moraes, I. M.; Velloso, P. B. Uma análise do impacto do ataque de poluição de cache em redes orientadas a conteúdo sem-fio. In *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)* (Nov. 2015), pp. 281–294.

[12] Conti, M.; Gasti, P.; Teoli, M. A lightweight mechanism for detection of cache pollution attacks in named data networking. *Computer Networks - Elsevier 57*, 1 (Aug. 2013), 3178–3191.

[13] da Silva, V. B. C.; Campista, M. E. M.; Costa, L. H. M. K. Trac: A trajectory-aware content distribution strategy for vehicular networks. *Vehicular Communications 5* (2016), 18 – 34.

[14] Deng, L.; Gao, Y.; Chen, Y.; Kuzmanovic, A. Pollution attacks and defenses for Internet caching systems. *Computer Networks: The International Journal of Computer and Telecommunications Networking 52*, 1 (Apr. 2008), 935–956.

[15] Gallo, M.; Perino, D.; Muscariello, L. Content-centric networking packet header format. Tech. Rep. BCP-78, Internet Engineering Task Force, 2015.

[16] Gao, Y.; Deng, L.; Kuzmanovic, A.; Chen, Y. Internet cache pollution attacks and countermeasures. In *IEEE International Conference on Network Protocols - ICNP* (Nov. 2006), pp. 54–64.

[17] Gasti, P.; Tsudik, G.; Uzun, E.; Zhang, L. DoS and DDoS in named-data networking. In *International Conference on Computer Communications and Networks - ICCCN* (Aug. 2013), pp. 1–7.

[18] Ghodsi, A.; Koponen, T.; Rajahalme, J.; Sarolahti, P.; Shenker, S. Naming in content-oriented architectures. In *ACM SIGCOMM Workshop on Information-Centric Networking - ICN* (Aug. 2011), pp. 1–6.

[19] Ghodsi, A.; Shenker, S.; Koponen, T.; Singla, A.; Raghavan, B.; Wilcox, J. Information-centric networking: Seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks* (New York, NY, USA, 2011), HotNets-X, ACM, pp. 1:1–1:6.

[20] Goergen, D.; Cholez, T.; François, J.; Engel, T. Security monitoring for content-centric networking. In *International Workshop on Autonomous and Spontaneous Security - SETOP* (Sept. 2013).

[21] Guimarães, F.; Rocha, A.; Albuquerque, C.; Ribeiro, I. Modeling ndn pit to analyze the limits of timeout on the effectiveness of flooding attacks. In *2016 IEEE Symposium on Computers and Communication (ISCC)* (June 2016), pp. 1245–1250.

[22] Jacobson, V.; Smetters, D.; Thornton, J.; Plass, M.; Briggs, N.; Braynard, R. Networking named content. In *International Conference on emerging Networking EXperiments and Technologies - CoNEXT* (Dec. 2009).

[23] Jacobson, V.; Smetters, D. K.; Thornton, J. D.; Plass, M.; Briggs, N.; Braynard, R. Networking named content. *Communications of the ACM 55*, 1 (Jan. 2012), 117–124.

[24] Kim, J.; Shin, D.; Ko, Y.-B. Top-ccn: Topology aware content centric networking for mobile ad hoc networks. In *2013 19th IEEE International Conference on Networks (ICON)* (Dec 2013), pp. 1–6.

[25] Kim, Y.; Kim, U.; Yeoml, I. The impact of large flows in content centric networks. In *IEEE International Conference on Network Protocols - ICNP* (Oct. 2013), pp. 1–2.

[26] Liao, W.-K.; Shih, P.-H. Architecture of proxy partial caching using http for supporting interactive video and cache consistency. In *Proceedings. Eleventh International Conference on Computer Communications and Networks* (Oct 2002), pp. 216–221.

[27] Mastorakis, S.; Afanasyev, A.; Moiseenko, I.; Zhang, L. ndnSIM 2.0: A new version of the NDN simulator for NS-3. Technical Report NDN-0028, NDN, January 2015.

[28] Mealling, M.; Denenberg, R. Report from the Joint W3C/IETF URI Planning Interest Group: Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations. IETF Network Working Group RFC 3305, Aug. 2002.

[29] Nakamura, R.; Ohsaki, H. Performance comparison of shortest-path routing and optimal detour routing in content-centric networking. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (June 2016), vol. 1, pp. 494–495.

[30] Nasserala, A.; Moraes, I. M. The producer-consumer collusion attack in content-centric networks. *ENIGMA - Brazilian Journal of Information Security and Cryptography 2*, 1 (Aug. 2015), 48–55.

[31] Nasserala, A.; Moraes, I. M. Uma avaliação do ataque de negação de serviço em conluio consumidor-produtor em redes orientadas a conteúdo. In *XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)* (2015), pp. 641–654.

[32] Nasserala, A.; Moraes, I. M. Analyzing the producer-consumer collusion attack in content-centric networks. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)* (Jan 2016), pp. 849–852.

[33] Nasserala, A.; Moraes, I. M. The producer-consumer collusion attack in content-centric networks. *IEEE Latin America Transactions 14*, 6 (June 2016), 3003–3010.

[34] Plagemann, T.; Goebel, V.; Mauthe, A.; Mathy, L.; Turletti, T.; Urvoy-Keller, G. From content distribution networks to content networks - issues and challenges. *Internation Journal for the Computer and Telecommunications Industry 29* (Mar. 2005), 551–566.

[35] Ribeiro, I.; Rocha, A.; Albuquerque, C.; Guimarães, F. On the possibility of mitigating content pollution in content-centric networking. In *39th Annual IEEE Conference on Local Computer Networks* (Sept 2014), pp. 498–501.

[36] Ribeiro, I. C. G.; de Q. Guimarães, F.; Albuquerque, C. V. N.; de A. Rocha, A. A. CCNcheck: um mecanismo de mitigação para poluição de conteúdos em redes centradas em conteúdo. In *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)* (Nov. 2013), pp. 114–127.

[37] Ribeiro, I. C. G.; Guimarães, F. Q.; Kazienko, J. F.; de A. Rocha, A. A.; Velloso, P. B.; Moraes, I. M.; Albuquerque, C. V. N. Segurança em redes centradas em conteúdo: Vulnerabilidades, ataques e contramedidas. In *Minicursos do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSeg* (2012), pp. 101–150.

[38] Smetters, D.; Jacobson, V. Securing network content. Tech. Rep. TR-2009-1, Xerox Palo Alto Research Center - PARC, 2009.

[39] Spring, N.; Mahajan, R.; Wetherall, D.; Anderson, T. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking 12*, 1 (Feb 2004), 2–16.

[40] Wang, L.; Hoque, A. K. M. M.; Yi, C.; Alyyan, A.; Zhang, B. An OSPF based routing protocol for named data networking. Technical Report NDN-0003, NDN, July 2012.

[41] Xie, M.; Widjaja, I.; Wang, H. Enhancing cache robustness for content-centric networking. In *IEEE Conference on Computer Communications - INFOCOM* (Mar. 2012), pp. 2426–2434.

[42] Zhang, L.; Estrin, D.; Burke, J.; Jacobson, V.; Thornton, J.; Smetters, D. K.; Zhang, B.; Tsudik, G.; Claffy, K.; Krioukov, D.; Massey, D.; Papadopoulos, C.; Abdelzaher, T.; Wang, L.; Crowley, P.; Yeh, E. Named Data Networking (NDN) project. Tech. Rep. NDN-0001, Xerox Palo Alto Research Center - PARC, 2010.