Universidade Federal do Paraná

Daniel de Rezende

# Uma Estratégia de Publicação do Status de Chaves Públicas em Redes Centradas na Informação

Curitiba PR
2017

Daniel de Rezende

# Uma Estratégia de Publicação do Status de Chaves Públicas em Redes Centradas na Informação

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Carlos Alberto Maziero.

Curitiba PR
2017

R467e Rezende, Daniel de
Uma estratégia de publicação do status de chaves públicas em redes centradas na informação / Daniel de Rezende. – Curitiba, 2017.
75 f. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2017.

Orientador: Carlos Alberto Maziero .
Bibliografia: p. 73-75.

1. Redes de informação. 2. Banco de dados – Medidas de segurança. 3. Infraestrutura de chaves públicas (Medidas de segurança para computadores). I. Universidade Federal do Paraná. II.Maziero, Carlos Alberto. III. Título.

CDD: 005.8

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós-Graduação INFORMÁTICA

# TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **DANIEL DE REZENDE** intitulada: **Uma Estratégia de Publicação do Estado de Chaves Públicas em Redes Centradas na Informação**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 11 de Agosto de 2017.

CARLOS ALBERTO MAZIERO
Presidente da Banca Examinadora (UFPR)

ANELISE MUNARETTO FONSECA
Avaliador Externo (UTFPR)

LUIZ CARLOS PESSOA ALBINI
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

*Dedico esta dissertação a minha família e amigos que me apoiaram durante esta jornada.*

# Agradecimentos

Agradeço primeiramente ao meu orientador, professor Carlos Maziero, que conduziu de forma brilhante os meus estudos. Obrigado por aceitar me orientar, por confiar na minha capacidade, pelo apoio com os artigos e por todos os ensinamentos.

Ao professor Luiz Carlos Erpen de Bona pela recomendação. Aos meus professores de graduação, Marcos Selmini e Francisco Barrela, pelas recomendações. Aos meus pais Sebastião Rezende e Neuza Rezende que sempre me incentivaram nos estudos. Ao meu irmão Wezen e às minhas filhas Anna Clara e Daniella pelo apoio.

Aos meus colegas que colaboraram com o meu trabalho, Ivan Pires, Elaine Alves, Santiago Virtel, Jeferson Paizano, Renato Melo e Valber Zacarkim.

Aos novos amigos que fiz em Curitiba, Paulo, André, Leandro, Ramiro, Walmir e Diego. Obrigado pela companhia no almoço e nas risadas e bate papo nos horários de café. Aos colegas Peter e Felipe Bombardelli pelas discussões políticas e filosóficas.

Aos professores da banca examinadora, Anelise Munaretto e Luiz Albini, obrigado pela disponibilidade, pela correção e pelas ótimas sugestões. Aos professores, Elias Duarte, André Grégio e Eduardo Cunha Almeida, obrigado pelos ensinamentos em suas respectivas disciplinas.

# Resumo

As redes centradas na informação (*Information-Centric Networking* - ICN) representam uma abordagem para aprimorar a infraestrutura da Internet, introduzindo dados nomeados como primitiva de rede. Os conteúdos tornam-se independentes de localização, aplicação, armazenamento e meios de transporte, permitindo o armazenamento em *cache*. Os benefícios esperados são melhorias na segurança, escalabilidade e redução da utilização de banda. Os conteúdos são nomeados pelos seus publicadores e precisam ser assinados digitalmente para garantir a sua integridade e proveniência. Para assinar os conteúdos, o publicador precisa utilizar os recursos disponibilizados por um sistema de infraestrutura de chaves públicas. Uma vez assinado, o conteúdo é servido aos clientes que devem recuperar a chave pública correspondente para validá-lo. Essa chave deve estar de acordo com as regras impostas pelo modelo de confiança adotado e precisa ser válida, ou seja, não pode ter sido revogada. Uma vez que uma chave é revogada, é necessário notificar os seus usuários. Este trabalho apresenta uma abordagem de um serviço para divulgação do status das chaves utilizadas para validação dos conteúdos. Este trabalho propõe um serviço distribuído e atualizado por replicação, visando melhorar sua robustez e diminuir o tempo necessário à propagação da informação do status das chaves na rede. Os principais resultados mostram que o serviço proposto pode reduzir significativamente o tempo de recuperação do status da chave, refletindo o estado atual do certificado. Os resultados também apontam que quando o consumidor recupera as informações de status da chave pelo serviço de consulta em vez do cache, ele obtém uma informação mais precisa.

**Palavras-chave:** Segurança, Redes Centradas em Informações, Revogação de chaves.

# Abstract

The Information Centric Networks (ICN) is an approach to improve the Internet infrastructure, introducing data named as network primitive. The contents are independent of location, application, transport, allowing the storage in cache. The expected benefits are improvements in security, scalability, and reduced bandwidth usage. The contents are named by their publishers and must be digitally signed to ensure their provenance. To sign the contents the publisher must use the resources provided by a public key infrastructure system. Once signed, the content is offered to customers, which should retrive the corresponding public key to validate it. This key must be valid in conformity with the rules enforced by the trust model and must be valid, i.e. it should not have been revoked. Once a key is revoked, its users should be notified. This paper proposes a service approach to disseminate key status information of the keys used to validate the contents. Main results shows that the proposed service is able to significantly reduce key status retrieval time, reflecting the actual state of the certificate. When the consumer retrieves the key status information by our query service instead of the cache, it gets a more accurate information.

**Keywords:** Security, Information Centric Networking, Key Revocation.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Acrônimos

| | |
|---|---|
| CA | Certificate Authority |
| CCN | Content-Centric Network |
| CRL | Certificate Revocation List |
| CS | Content Store |
| DER | Canonical Encoding Rules |
| DNS | Domain Name System |
| DONA | Data-Oriented Network Architecture |
| EC | Elliptic Curve |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| FIB | Forwarding Information Base |
| HMAC | Hash-based Message Authentication Code |
| ICN | Information Centric Networking |
| ICP | Infraestrutura de Chave Pública |
| IP | Internet Protocol |
| NDN | Named-Data Network |
| ndnSim | Named-Data Network Simulator |
| NRS | Name Resolution Service |
| OCSP | Online Certificate Status Protocol |
| PARC | Palo Alto Research Center |
| PIT | Pending Interest Table |
| PKI | Public-Key Infrastructure |
| PKIX | Public-Key Infrastructure (X.509) |
| PURSUIT | Publish Subscribe Internet Technology |
| RSA | Rivest-Shamir-Adleman |
| SDSI | Simple Distributed Infrastructure Security |
| SPKI | Simple Public-Key Infrastructure |
| TCP | Transmission Control Protocol |
| URI | Uniform Resource Identifier |

# Lista de Símbolos

| | |
|---|---|
| $P_x$ | Publicador |
| $C_x$ | Coordenador |
| $R_x$ | Réplicas |
| $Cl_x$ | Clientes |
| $Rc_x$ | Roteador de conteúdo |
| $ut$ | Tempo de atualização |
| $rt$ | Tempo de resposta |
| $rr$ | Taxa de resposta das réplicas |
| $ur$ | Taxa de informações atualizadas |

# Capítulo 1

# Introdução

A Internet original foi projetada para pesquisas com intuito de compartilhar recursos raros e caros, como computadores e periféricos em *links* de comunicação de longa distância. O requisito básico da Internet em seus primeiros anos, era apenas a de encaminhar pacotes de dados entre um número limitado de máquinas com relações de confiança bem estabelecidas. Quando a Internet passou a ser utilizada para efetuar transações comerciais, algumas iniciativas tentaram implementar segurança em algumas camadas [Bhimani, 1996]. Para fornecê-la, uma série de protocolos criptográficos foram propostos, entre eles, podemos destacar o SSL [Freier, 1996]. Que posteriormente foi substituído pelo TSL [Eastlake et al., 2011]. Entretanto, os esforços para aprimorar a segurança da arquitetura atual estão mais voltados para os *hosts* e não nos conteúdos gerados e armazenados por eles. Além disso, a arquitetura atual possui uma série de problemas relacionados à disponibilidade de conteúdo e dependência de localização [Jacobson et al., 2009].

Uma das propostas para aprimorar a distribuição de conteúdo, principal função da Internet atual, é o paradigma de redes centradas na informação (*Information-Centric Network* - ICN) [Ghodsi et al., 2011]. O seu objetivo é fornecer um serviço de infraestrutura de rede que seja mais adequado para o uso atual (em particular distribuição de conteúdo e mobilidade) e mais resistente a interrupções e falhas [Ahlgren et al., 2012]. De acordo com [Jacobson et al., 2009] uma das alternativas para aprimorar a distribuição de conteúdo seria substituir o "onde" pelo "o que". Portanto, nas redes centradas em informação os consumidores enviam solicitações de conteúdo pela sua identificação (nome), considerar o local onde este conteúdo esteja armazenado.

Nas redes centradas na informação o foco principal é o conteúdo, desta forma, a segurança também é aplicada diretamente aos conteúdos. Uma forma de aplicar a segurança é através da utilização de assinaturas digitais. A assinatura garante a integridade do conteúdo, ou melhor, garante que o conteúdo não foi modificado durante o transporte entre a sua origem e destino. Garante também a proveniência, isto é, que o conteúdo realmente foi produzido pelo publicador no qual o usuário confia, independentemente de onde ele tenha sido obtido. Essa relação de confiança pode ser construída através de um modelo de confiança a ser adotado pela aplicação que fará uso da ICN [Jacobson et al., 2009].

Quando uma chave criptográfica é comprometida, o usuário mal-intencionado que estiver em posse dessa chave pode gerar conteúdos falsos causando uma série de problemas para os usuários que desejam recuperar conteúdos legítimos. Esses problemas podem ser caracterizados como ataques de negação de serviço e ataque de envenenamento de conteúdo [Gasti et al., 2013]. Uma forma de evitar que os clientes aceitem os conteúdos assinados por chaves comprometidas é verificar o status da chave antes de utilizá-la para validar o conteúdo.

Além do modelo de confiança, é necessária uma infraestrutura para gerenciar a criação, distribuição e revogação das chaves criptográficas que serão utilizadas para assinar os conteúdos.

Sendo assim, um dos processos fundamentais do gerenciamento de chaves é a revogação e divulgação do status dessas chaves, tema pouco explorado em ICN. Uma vez que uma chave é revogada, é necessário notificar os seus usuários. Isso pode ser feito de duas formas: enviando essa informação para os usuários ou permitindo que eles a consultem quando necessário.

Considerando esse contexto e o desafio mencionado, este trabalho tem como objetivo propor um serviço de divulgação do status das chaves criptográficas em ICN. Esta abordagem consiste em utilizar um serviço de consulta do status de chaves, similar ao que é utilizado hoje na Internet. Este serviço poderá ser distribuído pela rede com intuito de reduzir o tempo de resposta e melhorar a disponibilidade da informação, reduzindo a dependência de um único ponto de acesso. Uma das formas de alcançar esse objetivo é distribuir várias cópias do serviço de consulta proposto pela rede. Para distribuir as informações do status dos certificados para as diversas cópias do serviço espalhados pela rede adotou-se uma solução de replicação de dados. Um dos principais benefícios da utilização da replicação de dados é a redundância, que torna o sistema tolerante a falhas.

Para avaliar o comportamento do serviço proposto, foram realizados experimentos por meio de simulações utilizando o simulador ndnSim [Mastorakis et al., 2016]. Este simulador é disponibilizado pelo projeto Named-Data Network (NDN) [NDN, 2010], arquitetura escolhida para realizar este trabalho. As simulações foram realizadas em um cenário bem próximo da Internet real, com o objetivo de medir o tempo de resposta de uma consulta do status de uma chave. Os resultados da simulação mostram que o consumidor pode obter o status da chave mais rápido do que consultando diretamente o produtor de conteúdo (o titular da chave). Além disso, os consumidores obtêm uma informação de status da chave de forma mais precisa do que se obtida através dos *caches* da ICN.

Este texto está organizado da seguinte forma: o Capítulo 2 fornece uma visão geral sobre as redes centradas na informação. Aborda também a forma como os conteúdos são assinados e validados. O Capítulo 3 uma visão geral da revogação de certificados na Internet e na NDN, e os problemas envolvidos nesse processo. O Capítulo 4 apresenta a proposta desenvolvida neste trabalho, uma abordagem para divulgação do status dos certificados em NDN. No Capítulo 5 apresenta o modelo de simulação construído para validar a proposta e os experimentos realizados sobre o mesmo, com os seus resultados. Finalmente, o Capítulo 6 apresenta conclusões e discute possíveis trabalhos futuros.

# Capítulo 2

# Redes centradas na informação

Este capítulo apresenta uma visão geral das redes centradas na informação (*Information Centric Networks* - ICN), mais especificamente a arquitetura *Named-Data Network* (NDN) [NDN, 2014]. Ele está divido em três seções. A Seção 2.1 exibe os principais fundamentos das redes centradas na informação. A Seção 2.2 mostra o funcionamento da arquitetura NDN. A Seção 2.3 aborda o processo de assinatura dos pacotes de dados em NDN. A Seção 2.4 apresenta um modelo de confiança proposta para NDN, que é necessário para validar a ligação entre o conteúdo e o seu publicador.

## 2.1 Fundamentos

As redes centradas na informação (*Information Centric Networks* - ICN) são caracterizadas por utilizarem conteúdos nomeados, por ter a segurança aplicada diretamente aos conteúdos e por permitir o armazenamento de conteúdos nos elementos do núcleo da rede [Jacobson et al., 2009]. Esse armazenamento permite aos usuários consultar os conteúdos em locais mais próximos à sua localização, reduzindo o tempo de consulta e diminuindo o tráfego da rede.

O modelo de comunicação da ICN é voltado para o conteúdo, conforme ilustrado na Figura 2.1, onde uma cópia confiável do conteúdo pode ser recuperada a partir de conexões e hospedeiros não confiáveis. Na ilustração, um usuário faz uma requisição de um conteúdo que está distribuído em diversos hospedeiros não confiáveis, que estão interligados por conexões não seguras. Em ICN, o processo de validação dos conteúdos é feito pelo usuário após obtê-los, sendo assim não existe preocupação com a origem do conteúdo e nem com caminho percorrido por ele para chegar até o solicitante.

Várias arquiteturas de ICN foram propostas, dentre elas: DONA [Koponen et al., 2007], PURSUIT [PURSUIT, 2010], NDN (*Named-Data Network*) [NDN, 2010], NetInf/SAIL [SAIL, 2010]. Cada uma das arquiteturas propostas enfatiza diferentes aspectos desse projeto, porém compartilham alguns conceitos [Ahlgren et al., 2012]:

- *Modelo de segurança orientado para o conteúdo*: Todos os projetos de ICN utilizam um modelo de segurança orientada para o conteúdo, no qual o conteúdo é assinado por seu publicador. Desse modo, elementos da rede e clientes podem verificar a autenticidade e a integridade do conteúdo através da verificação da assinatura;

- *Cache*: Nos projetos de ICN, quando um elemento de rede recebe uma solicitação de conteúdo ele pode responder diretamente, se o conteúdo estiver armazenado no seu *cache*, ou solicitar o conteúdo a partir do próximo elemento da rede;

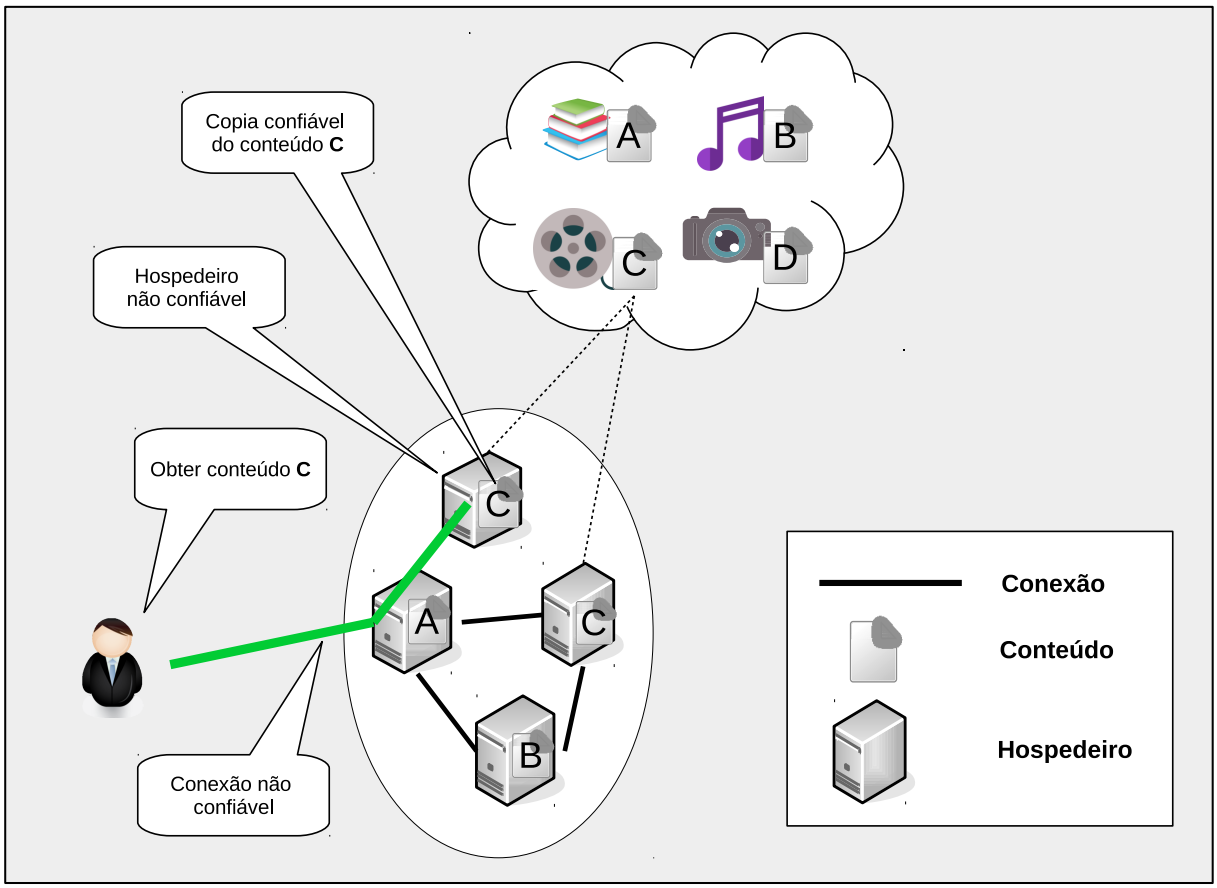


Figura 2.1: Modelo de comunicação da ICN na visão do cliente (adaptada de [Ahlgren et al., 2012]).

- *Roteamento*: A fim de satisfazer as solicitações de conteúdo, os pedidos devem ser roteados. O roteamento dos pedidos pode ser feito por nome ou pode ser auxiliado por um serviço de resolução de nomes (*Name Resolution Service* - NRS) similar ao *Domain Name System* (DNS).

Das várias arquiteturas propostas para ICN, a NDN (*Named-Data Network*) foi o que mais despertou o interesse dos pesquisadores. Tanto que vários projetos das demais arquiteturas propostas foram descontinuados. Atualmente a NDN conta com o maior número de publicação na área de ICN. Por esse motivo, e também por ter um simulador de código aberto, este trabalho escolheu essa arquitetura como referência.

## 2.2 Visão geral da NDN

A NDN surgiu tendo como base o projeto CCN (*Content Centric Network*) [Jacobson et al., 2009], criado pelo PARC (*Palo Alto Research Center*). O projeto NDN foi criado pela NSF (*National Science Foundation*), e faz parte do Programa de Arquiteturas da Internet do Futuro e tem como objetivo desenvolver uma nova arquitetura para a Internet. O projeto conta com a colaboração de pesquisadores de várias universidades de diversos países e de grandes corporações da área de tecnologia.

O sistema de nomeação de conteúdo adotado pela NDN é o de nomeação hierárquica, semelhante à URI[1]. Esse nome é formado por diferentes componentes que podem significar alguma informação a respeito da natureza do conteúdo, por exemplo: publicador, tipo, versão, etc. Por conveniência de notação, apresentamos nomes como URIs com caractere "/" para separação dos componentes, como na Figura 2.2, mas estes delimitadores não fazem parte dos nomes e não estão incluídos nas codificações de pacotes.


Figura 2.2: Exemplo de um dado nomeado.

A primeira parte do nome identifica o publicador do conteúdo, denominado de prefixo. Por exemplo, para o objeto `/ufpr.br/dinf/editais/edital.pdf` o nome do seu publicador é `ufpr.br`. Os últimos componentes dos identificadores são versão e segmentação do dado. Por exemplo, um cliente que quer a primeira versão do conteúdo `edital.pdf`, pode solicitar `/ufpr.br/dinf/editais/edital.pdf/1` com o seletor de interesse *ChildSelector* e receber um pacote de dados chamado `/ufpr.br/dinf/editais/edital.pdf/1/1` que corresponde ao primeiro pedaço. O cliente pode solicitar os pedaços (*chunks*) posteriores utilizando uma combinação da informação revelada pelo primeiro pacote de dados e a convenção de nomenclatura do aplicativo de publicação.

A comunicação em NDN é impulsionada pelos requisitantes, ou seja, os clientes de dados, através da troca de dois tipos de pacotes, indicados na Figura 2.3:

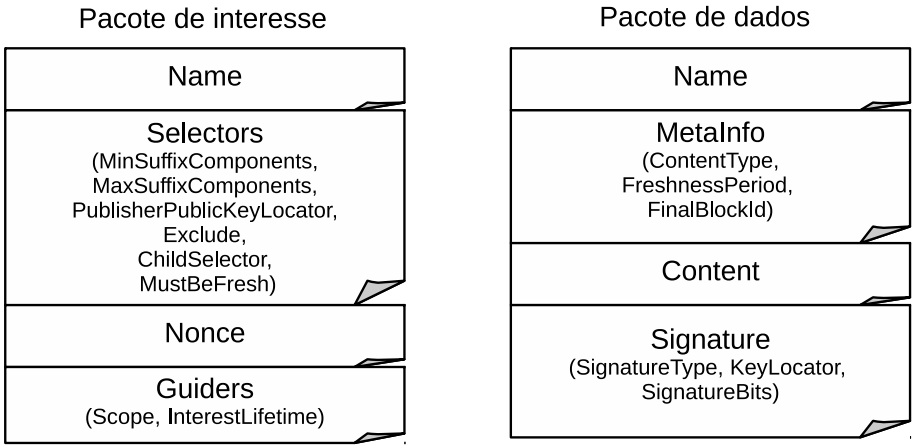


Figura 2.3: Pacotes na arquitetura NDN, figura baseada em [Jacobson et al., 2009] atualizada para a nova especificação em [NDN, 2014].

O pacote de interesse possui os seguintes campos:

- *Name*: nome que identifica o conteúdo requisitado;

[1]Um URI (*Uniform Resource Identifier*) é uma sequência de caracteres usada para identificar ou designar um recurso na Internet.

- *Selectors*: elementos opcionais que qualificam ainda mais dados que podem coincidir com o interesse. Eles são usados para descobrir e selecionar os dados que correspondem melhor ao que a aplicação deseja;
- *Nonce*: carrega uma sequência de caracteres gerados aleatoriamente. A combinação do *Name* e *Nonce* deve identificar exclusivamente um pacote de interesse;
- *Guiders*: especifica opções de comportamento do pacote de interesse, como limites de propagação e tempo de vida.

O pacote de dados possui os seguintes campos:

- *Name*: nome que identifica o conteúdo requisitado;
- *MetaInfo*: informações adicionais do pacote de dados, como o tipo de dado que o pacote carrega e o tempo de atualização. Por exemplo, um pacote de dados que carrega informações sobre uma chave possui o valor *KEY* no campo *ContentType*.
- *Content*: conteúdo do pacote, ou seja, os dados propriamente ditos.
- *Signature* : informações sobre a assinatura e localização da chave.

Ambas categorias de pacotes transportam um nome que identifica um pacote de dados. Um cliente coloca o nome que identifica o conteúdo desejado em um pacote de interesse e o envia para a rede. Os roteadores usam esse nome para encaminhar o interesse para os publicadores de dados. Uma vez que o pacote de interesse atinge um nó que possui os dados solicitados, o nó irá retornar um pacote de dados que contém o nome e o conteúdo, com uma assinatura feita pela chave privada do publicador. Este pacote de dados segue em sentido inverso do caminho tomado pelo interesse, de volta para o cliente que o solicitou [Zhang et al., 2014]. Os pacotes de dados são divididos em pedaços de tamanho padrão chamados de *chunks*, com objetivo de facilitar o controle da granularidade dos pacotes de dados para conteúdos muito grandes. Cada fragmento do pacote de dados é numerado sequencialmente para fins de controle.

Para encaminhar os pacotes, cada roteador NDN mantém três estruturas: tabela de interesses pendentes (*Pending Interest Table - PIT*), base de informações de encaminhamento (*Forwarding Information Base* - FIB) e um armazenador de conteúdo (*Content Store* - CS). Essas estruturas são responsáveis pelo recebimento e encaminhamento dos pacotes de interesse e dados. A FIB é uma estrutura semelhante à tabela de roteamento de um roteador IP. Entretanto, ela associa um prefixo de nome a cada interface de saída. Esse prefixo é a identificação do publicador de conteúdo, sendo armazenado na FIB do roteador quando um novo publicador se anuncia para a rede. A PIT é uma tabela que armazena os interesses por conteúdos recebidos pelo roteador através de pacotes de interesse. Ela só encaminha para a FIB um único pacote de interesse para o mesmo nome de conteúdo requisitado por um pacote de interesse. O CS tem como função aumentar a eficiência de obtenção dos conteúdos solicitados pelos clientes, através do armazenamento dos conteúdos em um *cache* local.

Para requisitar um conteúdo, o cliente deve enviar um pacote de interesse à rede contendo o nome do mesmo junto com o prefixo, que é a identificação do publicador. Após receber o interesse, o roteador verifica se esse conteúdo está armazenado no CS. Caso esteja, o roteador gera um pacote de dados para o conteúdo e o envia para a interface de chegada do pacote de interesse. Caso contrário, o roteador verifica se já existe uma entrada em sua PIT para o conteúdo. Se existir, o interesse é agregado, ou seja, sua interface de entrada é armazenada na entrada

correspondente na PIT. Caso contrário, ele faz uma busca pelo nome global nas informações de roteamento da FIB. Se uma rota para o interesse for encontrada, uma nova entrada para ele é criada na PIT e o pacote é encaminhado pela interface especificada na FIB. Caso contrário ele descarta o pacote ou envia um NACK[2] para a interface de chegada do pacote de interesse. O processo de tratamento do pacote de interesse é ilustrado na Figura 2.4.

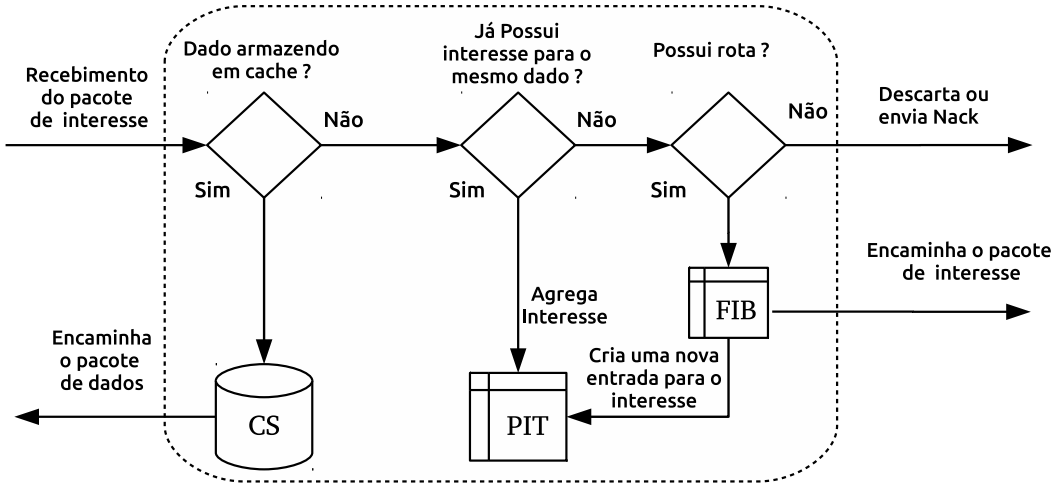


Figura 2.4: Tratamento de um pacote de interesse em roteador da NDN (adaptada de [Zhang et al., 2014]).

Quando um pacote de dados chega em uma das interfaces de um roteador de conteúdo da NDN, primeiro verifica-se a existência de algum interesse pendente para ele na PIT. Se existir um interesse pendente para esse mesmo conteúdo, de acordo com as políticas de *cache* o seu conteúdo pode ser armazenado no CS, sua entrada correspondente é removida da PIT e o pacote é encaminhado para a interface de destino. Caso não tenha interesse pendente para esse pacote de dados na PIT, o pacote de dados é descartado. Mesmo que não exista interesse pendente para o pacote de dados, e de acordo com as políticas de *cache*, ele pode ser armazenado no CS. O processo de tratamento do pacote de dados é ilustrado na Figura 2.5.

A Figura 2.6 mostra como as estruturas da NDN são preenchidas durante o processo de recuperação de um conteúdo. O solicitante lança um pacote de interesse para o conteúdo `/ufpr.br/docs/edital.pdf`, que é recebido pelo roteador de conteúdo $Rc_1$. Ao receber o $Rc_1$ registra o interesse em sua PIT o encaminha para o $Rc_2$ cuja de acordo com a rota definida da FIB. Ao receber o interesse o roteador de conteúdo $Rc_2$ registra o interesse em sua PIT e encaminha o interesse para o publicador1 de acordo com a rota definida em sua FIB. O publicador1 recebe o interesse, como possui o dado para satisfazer o interesse ele encaminha o dado para o $Rc_2$. Após receber o pacote de dados o $Rc_1$ guarda uma cópia do pacote de dados em seu CS, encaminha o pacote de dados de volta por onde foi solicitado e apaga o registo do interesse sua PIT. Ao receber o pacote de dados o $Rc_1$ guarda uma cópia do pacote de dados em seu CS, encaminha o pacote de dados para o solicitante e apara o registro do interesse de sua PIT. Após receber o pacote de dados, basta ao solicitante apenas validá-lo, processo que será abordado na seção seguinte.

Conforme mencionado anteriormente, a comunicação na NDN é impulsionada por dois tipos de pacote: interesse e dados. Ambos os pacotes são identificados pelo campo *Name*. No pacote de interesse os *Selectors* são elementos opcionais que qualificam ainda mais dados que

[2]NACK (*Negative Acknowledgement*) é um tipo de mensagem enviada em muitos protocolos de comunicação para reconhecer negativamente ou rejeitar uma mensagem recebida anteriormente.

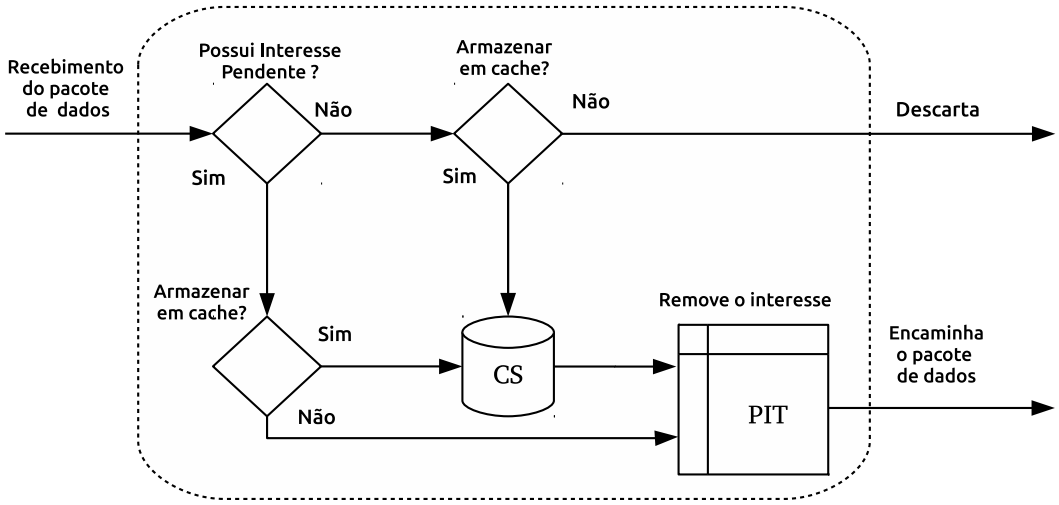


Figura 2.5: Tratamento de um pacote de dados em roteador da NDN (adaptada de [Zhang et al., 2014]).

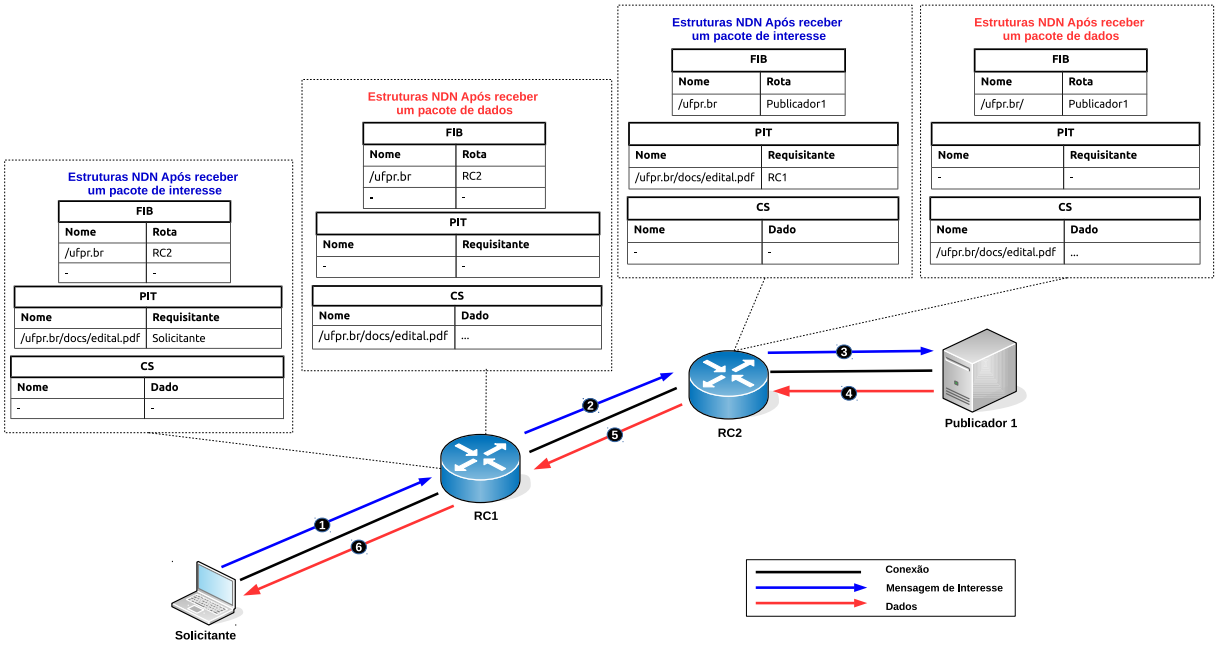


Figura 2.6: Tratamento de um pacote de dados em roteador da NDN (adaptada de [Zhang et al., 2014]).

podem coincidir com o interesse. Eles são usados para descobrir e selecionar os dados que corresponde melhor ao que o aplicativo deseja. De acordo com a especificação [NDN, 2014] os *Selectors* são:

*MinSuffixComponents e MaxSuffixComponents*: Estes dois parâmetros referem-se ao número de componentes do nome além do prefixo. Permitem que o cliente de dados indique se o nome do interesse é um nome completo. O padrão para *MinSuffixComponents* é 0 e para *MaxSuffixComponents* é efetivamente infinito, o que significa qualquer dado cujo nome começa a partir de um prefixo. Muitas vezes, apenas um deles será necessário para obter o efeito desejado;

*PublisherPublicKeyLocator*: Este elemento especifica o nome da chave, que é utilizada para assinar o pacote de dados que o cliente está solicitando. Esta é uma maneira para o interesse selecionar respostas de um publicador em particular;

*Exclude*: Permite ao solicitante especificar uma lista e/ou intervalo de nomes de componentes que não devem aparecer como uma continuação do prefixo do nome na resposta do pacote de dados para o interesse especificado. Por exemplo, se o interesse é expresso para `/ufpr.br/dinf` e especificar `larsis` no *Exclude*, então nem o produtor de dados assim como os roteadores estão autorizados a devolver qualquer pacote de dados que tem prefixo `/ufpr.br/dinf/larsis`;

*ChildSelector*: Muitas vezes um interesse pode encontrar mais de um dado dentro de um determinado armazenador de conteúdo. O *ChildSelector* fornece uma maneira de expressar uma preferência por qual deles devem ser devolvidos, assumindo que na hierarquia do nome o componente imediatamente após o prefixo do nome é o número da versão, conforme exemplificado na Figura 2.2.

*MustBeFresh*: Quando o *MustBeFresh* não é especificado no pacote de interesse, o roteador pode responder com um pacote de dados a partir de sua CS cujo *FreshnessPeriod* seja ainda válido ou expirado. Quando ele está presente em um pacote de Interesse, o roteador não deve retornar de pacotes de dados a partir de sua CS cujo *FreshnessPeriod* expirou. O *FreshnessPeriod* é um tempo inicial definido pelo publicador para cada pacote de dados. Uma contagem regressiva deste tempo começa quando o pacote de dados chega a um nó;

*InterestLifetime*: Campo opcional da sessão *Guiders* que indica o tempo que um interesse pode ser mantido na PIT.

Um pacote de dados representa os dados solicitados pelo respectivo pacote de interesse identificado pelo campo *Name*, ambos devem possuir a mesma informação neste campo. Além da carga de dados, o pacote possui alguns *bits* adicionais de meta-informação e uma assinatura digital. Conforme especificação [NDN, 2014] as meta-informações são:

*ContentType*: Tipo de conteúdo que será transportado pelo pacote de dados, conforme descrito na Tabela 2.1;

Tabela 2.1: Tipos de conteúdo de um pacote de dados NDN.

| Tipo de Conteúdo | Valor atribuído | Descrição do conteúdo |
|---|---|---|
| BLOB | 0 | Carga identificada pelo nome de dados;<br>este é o tipo de conteúdo padrão |
| LINK | 1 | Outro nome que identifica o conteúdo dos dados reais |
| KEY | 2 | Chave pública |
| NACK | 3 | NACK a Nível de Aplicação |

*FreshnessPeriod*: Indica quanto tempo um nó deve esperar após a chegada destes dados antes de marcá-lo como obsoleto. O valor codificado é o número de milissegundos. Deve-se observar que os dados obsoletos ainda são válidos. A expiração do *FreshnessPeriod* significa apenas que o produtor pode ter produzido dados mais recentes. Quando *FreshnessPeriod* é omitido, o pacote de dados não pode ser marcado como obsoleto;

*FinalBlockId*: Identificador do final do bloco de uma sequência de fragmentos. Ele deve estar presente no próprio bloco final, e pode também estar presente em outros fragmentos. O valor deve ser igual ao do último componente *Name* explícito do bloco final;

*Signature*: Define a assinatura e é dividido em dois blocos consecutivos: *SignatureInfo e SignatureValue*. As considerações gerais sobre os blocos *SignatureInfo e SignatureValue* se aplicam para todos as categorias de assinatura. O *SignatureInfo*, que é incluído no cálculo da assinatura, descreve totalmente a assinatura e quaisquer outras informações relevantes para obter o certificado, como *KeyLocator*. O *SignatureValue*, que é excluído do cálculo da assinatura, representa os *bits* reais da assinatura e qualquer outro material de suporte da assinatura.

Tabela 2.2: Tipos de assinaturas utilizadas na NDN.

| Valor | Referência | Descrição |
|---|---|---|
| 0 | DigestSha256 | Proteção de integridade com Hash Sha-256 |
| 1 | SignatureSha256WithRsa | Proteção da integridade e proveniência usando assinatura RSA sobre um *hash* SHA-256 |
| 3 | SignatureSha256WithEcdsa | Proteção da integridade e proveniência usando ECDSA sobre um *hash* SHA-256 |
| 4 | SignatureHmacWithSha256 | Integridade e proteção de proveniência usando código de autenticação de mensagem com base em *hash* SHA-256 |
| 2,5-200 | | Reservados para futuras atribuições |
| >200 | | Não atribuído |

*KeyLocator*: Especifica um nome que aponta para outro pacote de dados contendo certificado ou chave pública ou *KeyDigest* para identificar a chave pública dentro de um modelo de confiança específico. Deve-se observar que embora *KeyLocator* seja definido como um campo opcional no bloco *SignatureInfo*, algumas categorias de assinatura podem exigir a presença do mesmo e alguns requerem sua ausência.

De acordo com a Tabela 2.2 os seguintes tipos de assinaturas podem ser utilizados nos pacotes de dados da NDN:

- *DigestSha256*: Essa categoria de assinatura não garante a proveniência de um pacote de dados, ou seja, não garante de que o pacote foi recuperado a partir da fonte original. Essa categoria de assinatura destina-se apenas para fins de depuração e circunstâncias limitadas em que é necessário proteger apenas contra modificação inesperada durante a transmissão. *DigestSha256* é definido como um *hash* tipo *SHA256* do nome, *MetaInfo*, Conteúdo e *SignatureInfo*. Essa categoria de assinatura não requer nenhuma informação no campo *KeyLocator*, pois, para calcular o *hash* e verificá-lo não é necessário qualquer informação adicional. Se *KeyLocator* está presente em *SignatureInfo*, ele deve ser ignorado.

- *SignatureSha256WithRsa*: Algoritmo básico de assinatura que deve ser suportado por qualquer aplicativo desenvolvido para a arquitetura NDN. Define uma assinatura de chave pública RSA calculada sobre o *hash* tipo *SHA256* do nome, *MetaInfo*, Conteúdo e *SignatureInfo*. Essa categoria de assinatura garante estrita proveniência de um pacote

de dados, desde que a assinatura seja verificada e tenha sido produzida por um emissor que está autorizado a assinar o pacote de dados. O emitente da assinatura é identificado através do bloco *KeyLocator* no bloco *SignatureInfo* do *SignatureSha256WithRsa*. O campo *KeyDigest*, opcional no *KeyLocator*, é definido como *hash* do tipo *SHA256* sobre a codificação DER[3] do *SubjectPublicKeyInfo* para uma chave RSA, conforme definido em [Polk et al., 2002]. É responsabilidade do aplicativo definir regras (modelo de confiança) de quando um emissor específico (*KeyLocator*) está autorizado a assinar um pacote de dados específico.

- *SignatureSha256WithEcdsa*: Define uma assinatura de chave pública ECDSA (*Elliptic Curve Digital Signature Algorithm*) que é calculada sobre o *hash* tipo SHA256 do nome, *MetaInfo*, Conteúdo e *SignatureInfo*. Esse algoritmo de assinatura é definido na RFC5753 [Turner et al., 2009]. Essa categoria de assinatura garante estrita proveniência de um pacote de dados, desde que o emissor seja autorizado a assinar o pacote de dados. O emitente da assinatura é identificado usando o bloco *KeyLocator* no bloco *SignatureInfo* do *SignatureSha256WithEcdsa*. O campo *KeyDigest*, opcional no *KeyLocator*, é definido como *hash* do tipo *SHA256* sobre a codificação DER do *SubjectPublicKeyInfo* para uma chave de EC (*Elliptic Curve*), conforme definido pela RFC 5480 [Turner et al., 2009].

- *SignatureHmacWithSha256*: Define um código de autenticação de mensagens envolvendo uma função *hash* criptográfica (*Hash-based Message Authentication Code*) que é calculada sobre o Nome, *MetaInfo*, Conteúdo e *SignatureInfo*, usando *SHA256* como a função *hash*, combinados com uma chave secreta compartilhada. Este algoritmo de assinatura é definido na RFC 2104 [Madson e Glenn, 1998]. A chave secreta compartilhada não está incluída na assinatura e não deve ser incluída em qualquer parte do pacote de dados, uma vez que invalidaria as propriedades de segurança do HMAC. Essa categoria de assinatura garante a proveniência, pois, o pacote de dados foi assinado por uma das partes que detém a chave compartilhada. É responsabilidade do aplicativo estabelecer uma associação entre a chave compartilhada e as identidades das partes que a detêm.

## 2.3 Assinatura dos pacotes

Em contraste com as redes TCP/IP, que deixam a responsabilidade pela segurança (ou a falta dela) para os dispositivos, a NDN protege os dados, obrigando os publicadores a assinar criptograficamente cada pacote de dados [Jacobson et al., 2009]. Os pacotes de dados são assinados utilizando criptografia de chave pública, também conhecida como criptografia assimétrica. O algoritmo padrão utilizado para assinar os pacotes NDN é o RSA [Rivest et al., 1978]. A assinatura é produzida sobre o resultado de uma função de *hash*, que é computado utilizando todos os elementos do pacote de dados, com exceção do campo onde a assinatura é armazenada. Esta assinatura é armazenada no final do pacote para facilitar sua verificação, conforme mostra a Figura 2.7. Cada pacote de dados assinado contém informações que permitem ao cliente obter a chave pública necessária para verificá-lo. A informação de localização da chave está armazenada no campo *KeyLocator* do pacote de dados.

[3]DER são regras de codificação que definem como os dados serão criados e transmitidos. Destina-se a situações em que é necessária uma única codificação, tais como em criptografia, e garante que uma estrutura de dados que

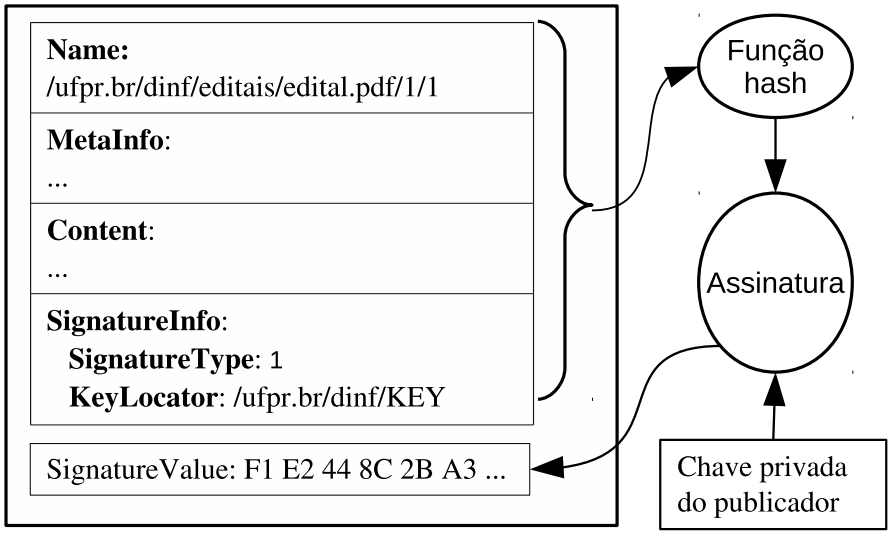


Figura 2.7: Processo de assinatura de pacotes.

Em NDN, um pacote de dados que se refere a um indivíduo ou organização e cujo conteúdo é uma chave pública, essencialmente é um certificado digital [Jacobson et al., 2012]. O pacote de dados de chaves também deve ser assinado para garantir a sua integridade e proveniência, esse processo é repetido consecutivamente até chegar até a âncora de confiança do modelo de confiança adotado, conforme ilustrado na Figura 2.8. Sendo assim, cada pacote de dados deve ser verificado com a respectiva chave informada no *KeyLocator*. Mais detalhes sobre modelo de confiança serão abordados na próxima seção.

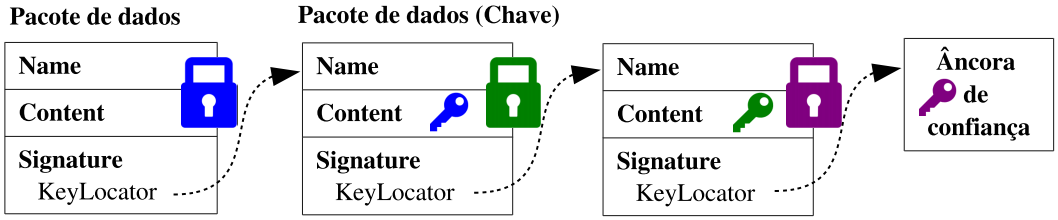


Figura 2.8: Assinatura de pacotes [Yu et al., 2015]).

Qualquer elemento da rede pode validar um pacote de dados e não apenas os pontos finais da comunicação. Essa verificação mínima é útil para detectar pacotes corrompidos e para a defesa contra ataques na rede. Os roteadores de conteúdo podem optar por verificar todos, alguns ou nenhum dos pacotes de dados. Seus recursos podem permitir que eles se adaptem dinamicamente, em resposta a um ataque detectado ou a alguma orientação ao cliente [Jacobson et al., 2012]. No entanto, para fazer essa verificação, os roteadores precisar conhecer e validar o modelo de confiança adotado pela aplicação; Isso aumenta a complexidade do processo de validação, portanto, é recomendado que a validação seja feita pelo cliente. Desta forma, os clientes devem recuperar as chaves necessárias para validar os pacotes de dados de acordo com a Figura 2.9.

tem de ser assinada digitalmente produz uma representação em série única. DER é amplamente utilizado para certificados digitais, tais como X.509.

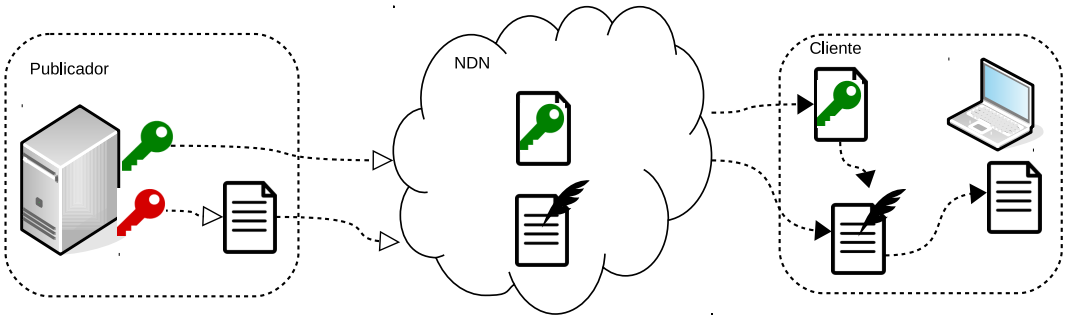


Figura 2.9: Assinatura e verificação dos pacotes de dados [Yu et al., 2015]).

## 2.4 Modelo de confiança

Para verificar a ligação entre o conteúdo e o publicador, a NDN recomenda que os usuários adotem um modelo de confiança. A confiança é estabelecida entre os publicadores de conteúdo e os clientes. O que é apropriado para uma aplicação pode não ser para outra, sendo assim, os usuários são livres para reutilizar modelos existentes (por exemplo, PKI[4]) para estabelecer a confiança em chaves ou para definir novos modelos mais apropriados para cada categoria de aplicação [Jacobson et al., 2009].

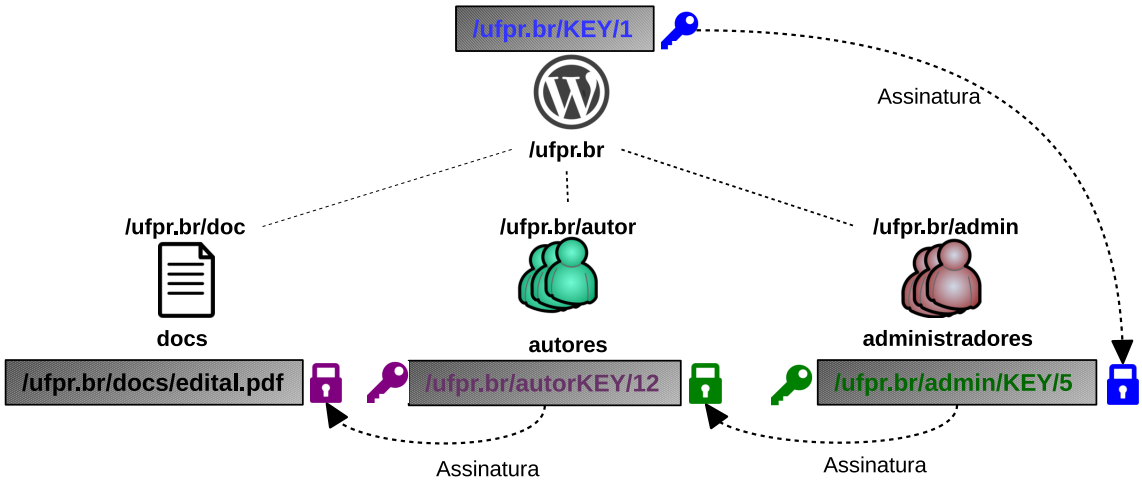


Figura 2.10: Exemplo de modelo de confiança [Yu et al., 2015]).

Um modelo de confiança foi proposto para NDN [Yu et al., 2015] composto por regras e esquemas de confiança. Expressões regulares são utilizadas para definir as regras entre os elementos envolvidos. Esse modelo funciona de forma independente de aplicação e vem sendo utilizado para prover o gerenciamento de confiança em diversas aplicação da NDN. O modelo proposto é exemplificado na Figura 2.10, onde um *site* ou aplicação possui uma âncora de confiança (`ufpr.br`) , que autoriza administradores (`/ufpr.br/admin`) a delegar funções como autorizar os autores (`/ufpr.br/autor`) a assinar os conteúdos publicados pela aplicação.

O objetivo deste trabalho não é avaliar ou apontar qual o modelo de confiança mais indicado para uma determinada aplicação em NDN. No entanto, ele destaca a importância do modelo de confiança no processo de validação de um pacote de dados.

[4]Infraestrutura de Chave Pública (*Public-Key Infrastructure* - PKI) é um sistema de gerenciamento de chaves públicas especificado em [Housley et al., 1999].

## 2.5 Considerações finais

Este capítulo apresentou uma visão geral das redes centradas em informações, descrevendo as particularidades da arquitetura NDN. A Seção 2.1 descreve o funcionamento da arquitetura NDN, detalhando como a troca de pacotes é efetuada e quais são os componentes envolvidos nesse processo. A Seção 2.2 aborda o processo de assinatura dos pacotes, onde os pacotes de dados devem ser assinados digitalmente para garantir sua integridade e proveniência. Os pacotes são assinados com a chave privada do publicador do conteúdo. Os clientes precisam obter a chave pública correspondente para validar os conteúdos recuperados. A Seção 2.3 apresenta um modelo de confiança proposta para NDN, necessário para validar a ligação entre o conteúdo e o seu publicador.

As chaves utilizadas para validar os pacotes de dados não devem estar expiradas ou não podem estar revogadas. O responsável pelas chaves deve manter os usuários informados sobre as chaves que foram revogadas. O próximo capítulo apresenta uma visão detalhada do processo de revogação e validação de chaves na Internet e na ICN, destacando o problema e as soluções propostas para solucioná-lo.

# Capítulo 3

# Revogação de certificados

Este capítulo apresenta uma visão geral da dinâmica e das tecnologias envolvidas no processo de revogação de certificados digitais na Internet e em ICN. A Seção 3.1 apresenta o conceito de infraestrutura de chave pública e gerenciamento de certificados digitais. A Seção 3.2 aborda os métodos de revogação de certificados digitais utilizados na Internet. A Seção 3.3 apresenta as propostas para revogação de certificados digitais em ICN.

## 3.1 Gerenciamento de certificados digitais

A criptografia de chave pública ou criptografia assimétrica é um método que utiliza um par de chaves: uma chave pública e uma chave privada. A chave pública é distribuída para os interessados em enviar mensagens cifradas para o proprietário da chave, enquanto a chave privada deve ser conhecida apenas pelo seu proprietário. Quando uma mensagem é cifrada com a chave pública do destinatário, a confidencialidade da mensagem é garantida, pois, somente o destinatário da mensagem conseguirá decifrá-la, utilizando a sua chave pública [Diffie e Hellman, 1976]. Alguns algoritmos de chave pública também podem ser utilizados para garantir a autenticidade de uma mensagem. Uma vez que a mensagem é cifrada com a chave privada, ela só poderá ser decifra-la com a chave pública correspondente, garantindo que a mensagem não foi forjada. Um dos algoritmos mais populares de criptografia assimétrica que pode ser utilizado para produzir assinaturas digitais é o RSA [Rivest et al., 1978].

O conceito de certificados digitais foi desenvolvido para associar os nomes dos comunicantes às suas respectivas chaves públicas. Os certificados eram armazenadas em um arquivo público que funcionava como um cartório, onde somente ele tem permissão para emitir os certificados [Kohnfelder, 1978]. Desde então, esse conceito tem sido utilizado para verificar se uma chave pública realmente pertence ao seu usuário. Os certificados digitais foram sendo aprimorados e vêm sendo utilizados como credenciais de segurança nas mais diversas aplicações onde é necessário utilizar criptografia de chave pública.

Uma infraestrutura de chave pública (ICP ou *Public-Key Infrastructure* - PKI) é um sistema utilizado para publicar as chaves públicas utilizadas na criptografia de chave pública. Uma PKI fornece uma estrutura com ampla variedade de componentes, aplicações, políticas e práticas combinadas para atingir quatro funções de segurança: integridade, autenticação, confidencialidade e não repúdio. A PKI é uma combinação de hardware e produtos de software, políticas e procedimentos. Ela fornece a segurança básica necessária para realizar negócios na Internet, para que os usuários que não conhecem uns aos outros ou estão distribuídos à distância possam se comunicar de forma segura através de uma cadeia de confiança. Certificados digitais são componentes vitais na PKI, pois, eles agem como passaporte digital através da

ligação da assinatura digital do usuário para sua chave pública. As principais funções de uma PKI são: registro, emissão e revogação de certificados, criação e publicação de lista de certificados revogados (LCR ou *Certificate Revocation List - CRL*), armazenamento e recuperação de certificados e listas de certificados revogados, bem como gerenciamento do ciclo de vida de chaves [Hunt, 2001].

O modelo de PKI descrito pela RFC 5280 consiste em cinco categorias de componentes conforme ilustrado na Figura 3.1:

- *Entidade final*: Usuário do certificado PKI e/ou sistema do usuário final que é o sujeito de um certificado;
- *Autoridades de Certificação* (CA do acrônimo em inglês para *Certificate Authority*): Entidades que emitem e revogam os certificados;
- *Autoridades de Registro* (RA do acrônimo em inglês para *Registration Authority*): Um componente opcional que pode assumir diversas funções administrativas da CA;
- *Emissor da CRL*: Um componente opcional que uma CA pode delegar para publicar as listas de revogação de certificados;
- *Repositório*: Um sistema ou conjunto de sistemas distribuídos que armazena certificados e CRLs, e serve para distribuir esses certificados e CRLs para entidades finais.

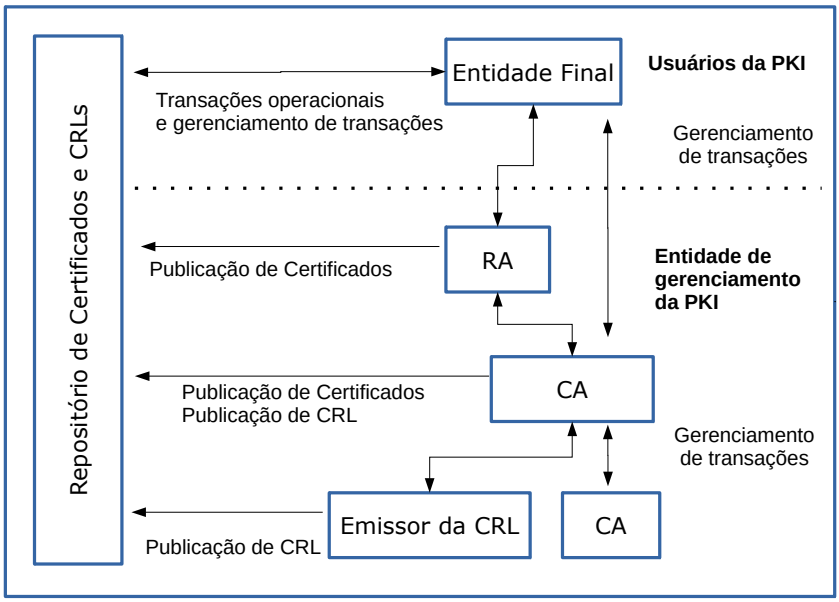


Figura 3.1: Entidades PKI e seus processos. Figura traduzida de [Cooper et al., 2008].

Os certificados digitais do padrão X.509 são estruturas de dados complexas que oferecem uma variedade de extensões, que podem assumir uma ampla gama de opções. Isto proporciona uma flexibilidade considerável, permitindo que o formato de certificado X.509 possa ser usado em muitas aplicações. Para construir uma PKI com base em certificados X.509, o grupo de trabalho PKIX (Public-Key Infrastructure X.509) desenvolveu um perfil da especificação X.509 versão 3 [Hunt, 2001]. Em sistemas PKI, a hierarquia de certificados é sempre baseada em uma

árvore de cima a baixo, com o certificado raiz no topo. O certificado raiz é auto assinado e representa a CA principal. Os certificados abaixo, das CAs intermediárias, são assinados pela CA principal. Uma CA intermediária possui autorização da CA principal para assinar certificados digitais.

A Figura 3.2 mostra um exemplo de um certificado digital gerenciado pela autoridade certificadora raiz brasileira (ICP-Brasil) e por uma CA intermediária (Secretaria de Receita Federal do Brasil). O certificado foi emitido pela autoridade de registo VALID Certificadora, e é utilizado para acessar os sistemas gerenciados pela Receita Federal e também assinar notas fiscais eletrônicas.

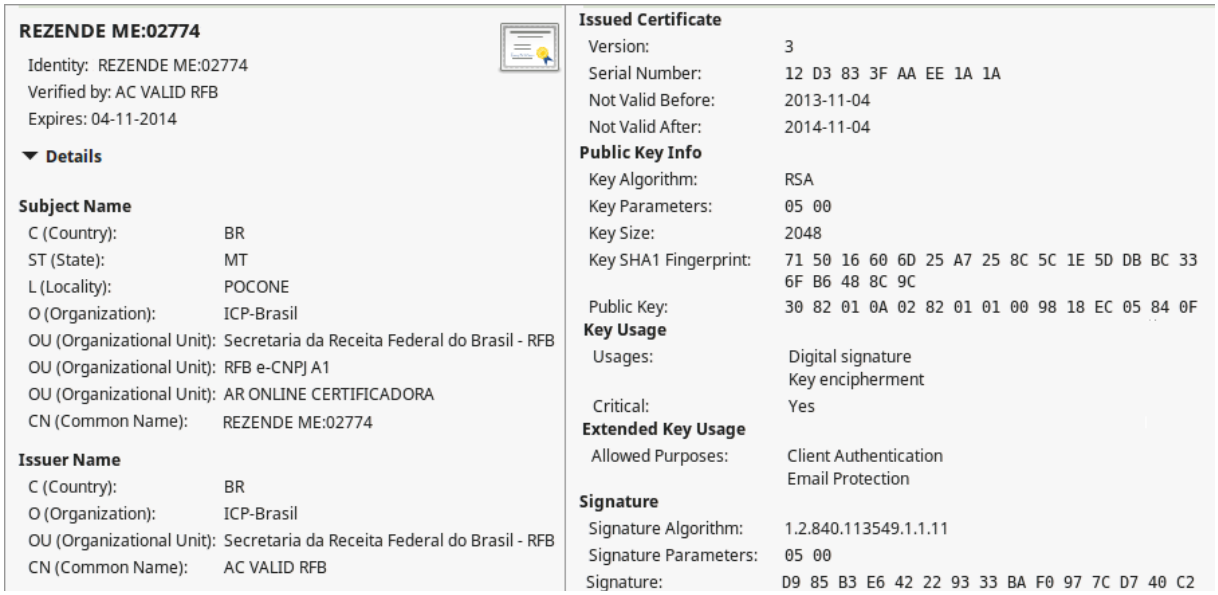


Figura 3.2: Exemplo de um certificado X.509.

## 3.2 Revogação de certificados na Internet

Um certificado digital tem um período de validade durante o qual ele é confiável. Após seu período de validade, um certificado torna-se inválido e não é mais confiável. Durante o período de validade do certificado, o emissor do certificado deve manter e fornecer informações sobre o status do mesmo. O status "revogado" indica que o período de validade do certificado foi prematuramente encerrado, portanto, não é mais confiável.

### 3.2.1 Lista dos certificados revogados

Uma Autoridade Certificadora (*Certification Authority* - CA) pode revogar um certificado por diversas razões, como: comprometimento da chave privada do titular; comprometimento da chave privada da autoridade certificadora, o que significa que todos os certificados emitidos pela mesma são potencialmente não-confiáveis e devem ser revogados; mudanças nas informações relativas ao titular do certificado; violação da política de segurança da autoridade certificadora pelo assinante; entre outras [Cooper et al., 2008]. Para divulgar as informações de revogação para os usuários dos certificados, alguns métodos foram implementados e serão apresentados a seguir.

Uma CA deve divulgar uma lista de revogação de certificados (*Certificate Revocation List - CRL*). Essa lista é uma estrutura de dados digitalmente assinada pela CA emissora dos mesmos, que contém: a data e a hora de sua publicação; o nome da CA emissora e o número de série dos certificados revogados que ainda não expiraram. Para poder confiar em um certificado, a aplicação deve verificar se seu número de série não consta da lista de revogação de certificados. Para tal, vários métodos diferentes podem ser empregados. A lista de revogação de certificados pode ser consultada pela aplicação do usuário, acessando um repositório disponibilizado pela CA e fazendo *download* da lista. A aplicação precisa saber quando a CA publicará uma nova lista para poder manter-se atualizada. Quando o titular de um certificado solicita uma revogação, ele deve aguardar até que a CA publique a próxima CRL. Toda CRL deve possuir a data e hora em que será publicada uma nova CRL. A CA poderá antecipar a sua atualização, mas, nunca deverá atrasar [Cooper et al., 2008]. A Figura 2.5 mostra a estrutura de uma CRL.

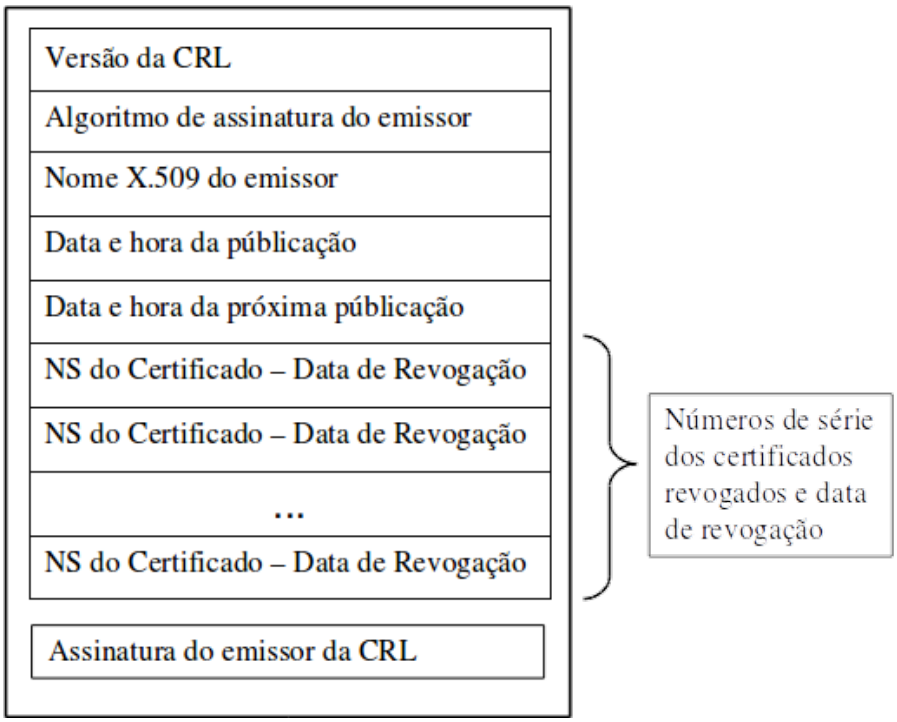


Figura 3.3: Estrutura de uma CRL, imagem adaptada de [Cooper et al., 2008].

Na CRL versão 3 foram introduzidas algumas extensões para auxiliar no seu funcionamento [Cooper et al., 2008]:

- *Número da CRL*: Esta extensão atribui a cada CRL um número que cresce monotonicamente. Esta extensão permite que os usuários determinem facilmente quando uma determinada CRL substitui outra CRL;

- *Pontos de distribuição da CRL*: Essa extensão combina informações de revogação de vários CAs em uma CRL. O objetivo global é reduzir o tamanho da CRL processada pelos usuários da CA. A CA pode fracionar a CRL de alguma forma e distribuir suas partes em diferentes pontos de distribuição. Desta forma, o usuário não precisa baixar uma CRL completa que contém informações que não são do seu interesse. Por exemplo, uma CA corporativa pode emitir uma CRL diferente para cada divisão da empresa. Um usuário que queira verificar um certificado para um empregado de uma divisão particular só precisa verificar a CRL da divisão. Há outras maneiras de efetuar a

fragmentação da CRL. Por exemplo, uma CA pode dividir informações de revogação devido ao motivo da revogação. Revogações de rotina, como as que ocorrem quando os dados dos empregados são alterados, podem ser armazenadas separadamente das revogações que são devidas a comprometimento de segurança;

- *Delta-CRL*: É outra forma de reduzir o tamanho da CRL. Em vez de emitir uma CRL cheia, a CA só anuncia uma lista de mudanças que ocorreram desde a última emissão de CRL. Os usuários atualizam seu próprio banco de dados de CRL com as informações do delta-CRL, de modo a manter uma CRL atualizada. Fazer o *download* e processar uma delta-CRL economiza banda e tempo de computação em relação ao processamento de uma CRL completa;
- *CRL indireta*: Esta extensão permite que uma CRL seja emitida a partir de um terceiro confiável, que não necessariamente emita certificados. A ideia subjacente é que o terceiro reúna informações de várias CAs e forneça informações de revogação para todas elas.

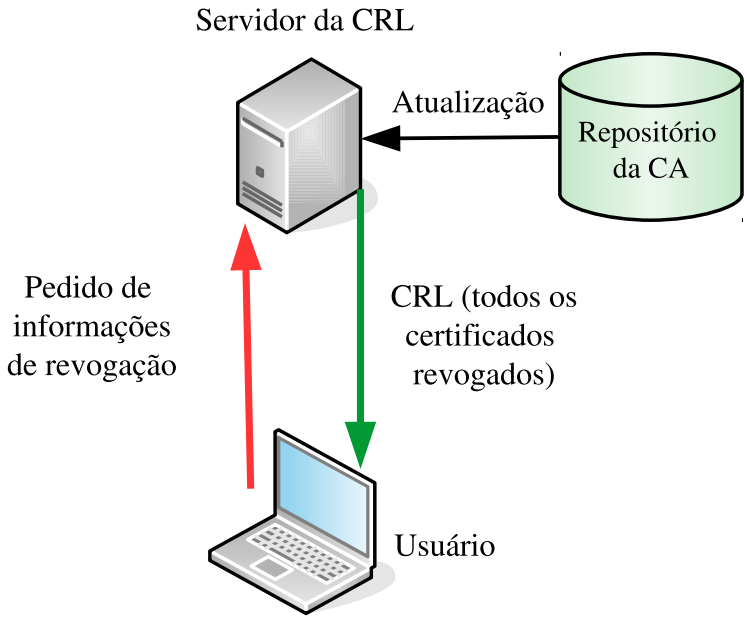


Figura 3.4: Processo de atualização da CRL na aplicação cliente.

Essas extensões foram propostas para amenizar o problema de crescimento do tamanho das listas de revogação. Dependendo da quantidade de certificados gerenciados por uma PKI as CRLs podem se tornar difíceis de se administrar.

A Figura 3.4 mostra a atualização de uma CRL na aplicação cliente. Uma CA emite uma CRL para um repositório periodicamente, e o usuário o acessa para recuperar a versão mais atualizada da CRL, na qual a aplicação precisa recupera a lista com todos os certificados de uma CA específica. O usuário não tem opção de escolher somente o certificado que deseja consultar, sendo possível recuperar mais informações do que o necessário.

### 3.2.2 OCSP

Para tentar sanar as deficiências da CRL o protocolo OCSP (*Online Certificate Status Protocol*) foi criado [Santesson et al., 2013]. Este protocolo permite que as aplicações possam

consultar o status de um certificado e obter três respostas possíveis para o status: bom, revogado ou desconhecido. Caso a resposta da consulta devolva o status bom, significa que o certificado não foi revogado. Considera-se que o certificado está dentro do seu período de validade, uma vez que não é recomendado enviar consultas de status de certificados vencidos, por meio desse serviço, para não gerar tráfego desnecessário. Se a resposta da consulta devolver o status revogado, significa que o certificado foi revogado. Se a resposta da consulta devolver o status desconhecido, significa que o sistema não reconhece o certificado especificado na consulta, que pode ter sido emitido por outra CA. As consultas efetuadas ao serviço de OCSP retornam menos informações que uma CRL, uma vez que ele só responde pelo certificado especificado na consulta, enquanto a CRL retorna uma lista que poder ter informações de mais de um certificado. Por outro lado, o cliente precisa estar conectado ao OCSP para realizar as consultas, enquanto a CRL funciona de forma *offline*.

A Figura 3.5 mostra como um usuário consulta o status de revogação de um certificado, utilizando OCSP. O usuário emite uma solicitação de status de um certificado especificado para um servidor OCSP, e recebe uma resposta dele. O OCSP deve obter a CRL periodicamente da CA.

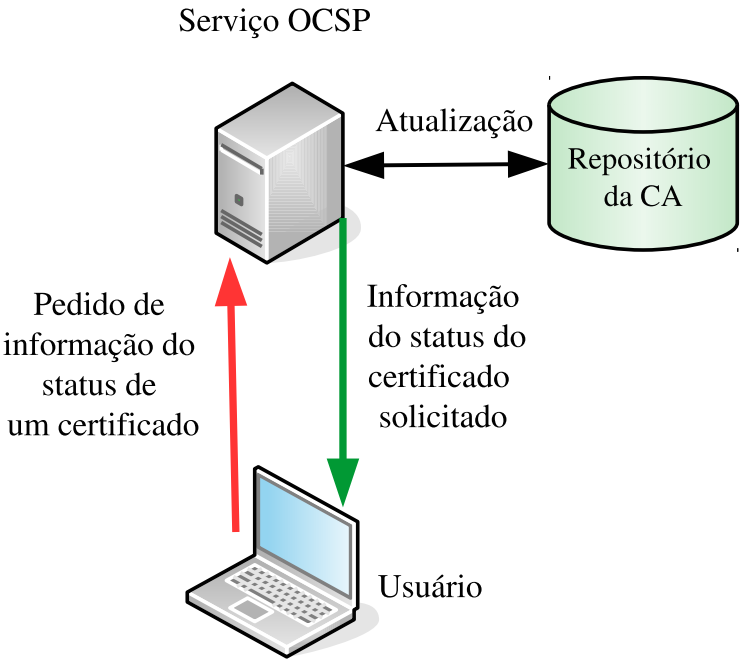


Figura 3.5: Consulta do status de um certificado.

Basicamente, existem dois tipos de formatos de mensagem para OCSP, pedido (OCSP *request*) e a resposta (OCSP *response*). A mensagem de pedido é produzida por um usuário, e enviado para o serviço OCSP.

Um pedido OCSP contém os seguintes dados [Santesson et al., 2013]:

- Versão do protocolo;
- Serviço requisitado;
- Identificador do certificado que precisa ser validado;
- Extensões opcionais que podem ser processadas pelo OCSP.

Após a recepção de um pedido, uma resposta OCSP determina se:

- A mensagem está bem formada;
- A resposta está configurada para fornecer o serviço solicitado;
- O pedido contém as informações necessárias para o respondedor.

Se qualquer uma dessas condições não for atendida, o respondedor OCSP produz uma mensagem de erro, caso contrário, retorna uma resposta. Há um tipo resposta padrão que o OCSP devem ser fornecer para os clientes, composta de:

- Versão da sintaxe da resposta;
- Identificador do respondedor;
- Quando a resposta foi gerada;
- Respostas para cada um dos certificados em um pedido;
- Extensões opcionais;
- Algoritmo de assinatura;
- Assinatura computada através de um *hash* da resposta.

Os clientes devem utilizar a extensão *Authority Infomation Access* do certificado X.509 para obter o endereço de acesso ao serviço de OCSP. A extensão *CRL Distribution Points* indica o endereço do servidor que armazena as CRLs. Primeiramente os clientes tentam obter a resposta através do OCSP. Se nenhuma resposta for obtida após um tempo limite e número de tentativas configurados, então ele tenta recuperar a CRL. Antes de pedir a um cliente OCSP para verificar o status do certificado, o aplicativo do cliente deve validar a assinatura em certificados em uma cadeia. Se a assinatura do certificado é inválida ou a aplicação não é capaz de verificá-la, uma verificação junto ao OCSP não deve ser requerida. A implementação OCSP pode apresentar um custo significativo para as CAs, porque as obriga a dar respostas a todos os clientes de um determinado certificado em tempo real. Por exemplo, quando um certificado é emitido para um site de alto tráfego, os servidores da CA são susceptíveis de serem atingidos por enormes volumes de solicitações OCSP consultando o status de certificados [Deacon e Hurst, 2007]. A Figura 3.6 mostra os campos *Authority Information Access* e *CRL Distribution Points* de um certificado X.509.

Através de uma extensão do protocolo TLS 1.2 [Eastlake et al., 2011], enquanto o cliente estabelece uma conexão TLS, também conhecido como *handshake*, ele pode solicitar e receber a resposta OCSP contendo o status de revogação do certificado digital utilizado pelo servidor. Isso é possível, pois, o servidor onde o serviço está hospedado e que possui o certificado digital a ser verificado, já efetua a solicitação da resposta OCSP para a CA que emitiu o certificado e envia essa resposta ao usuário. Essa técnica é chamada de grampeamento OCSP (*OCSP Stapling*). Embora pareça permitir que o operador do *site* possa controlar as respostas de verificação permitindo que um site fraudulento emita verificações falsas de um certificado revogado, as respostas grampeadas não podem ser forjadas, pois elas precisam ser assinadas diretamente pela autoridade de certificação e não pelo servidor do *site*. Sendo assim, caso o cliente recebe uma resposta grampeada inválida a conexão será abortada. Se o cliente não receber uma resposta grampeada, ele mesmo vai entrar em contato com o servidor OCSP para obter o

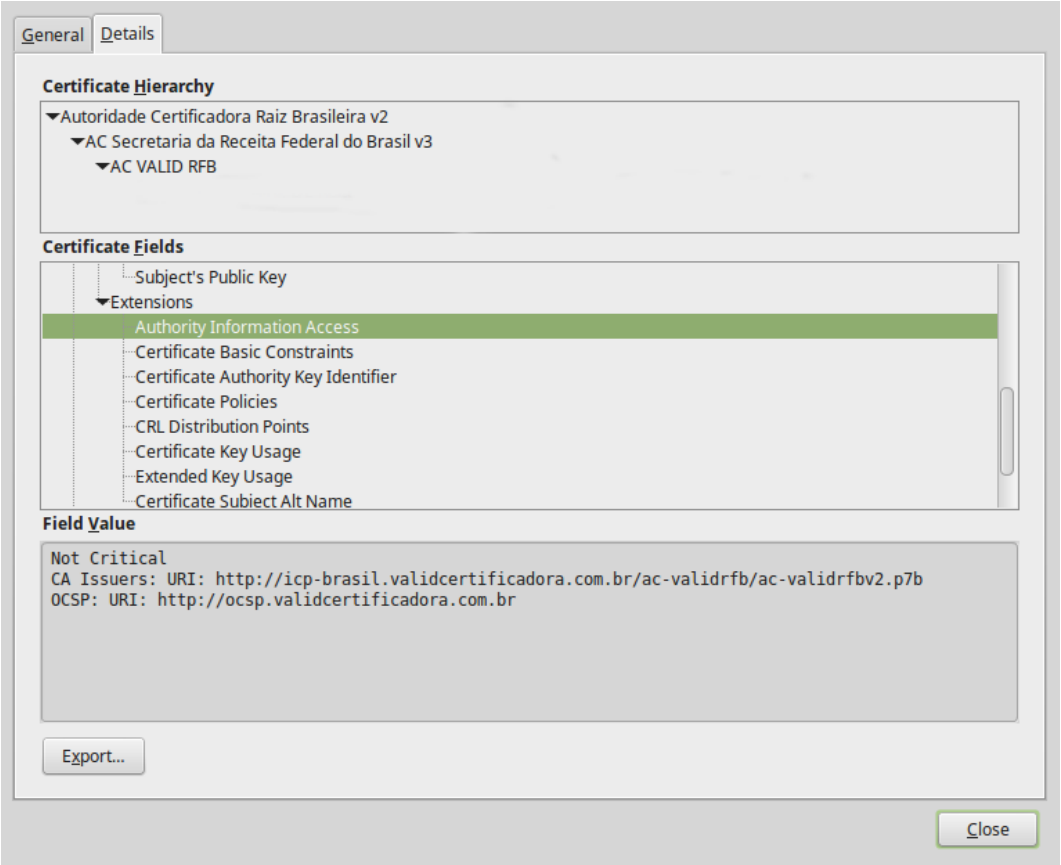


Figura 3.6: Extensão *Authority Information Access* do certificado X.509.

status do certificado. O único risco aumentado do grampeamento OCSP é que a notificação de revogação de um certificado pode ser adiada até a última resposta OCSP assinada expirar. Como resultado, os clientes continuam a ter a garantia verificável da autoridade de certificação que o certificado é atualmente válido (ou era muito recentemente), mas não precisa entrar em contato individualmente com o servidor OCSP. Esse mecanismo garante a privacidade do usuário e reduz o consumo de recursos de conectividade, principalmente em sistemas com grande fluxo de usuários.

### 3.2.3 Outras abordagens

Devido à complexidade do esquema global de nomeação da infraestrutura de chaves públicas (PKI) o sistema SPKI/SDSI foi proposto, com uma especificação mais simples, como alternativa ao X.509. O SPKI (*Simple Public-Key Infrastructure*) surgiu através da junção de propostas distintas. O SDSI (*Simple Distributed Security Infrastructure*) projetado por [Rivest e Lampson, 1996], é uma infraestrutura de chaves públicas que possui como principal característica o espaço de nomes local. O SPKI criado por [Ellison et al., 1999] surgiu com intuito de ser um modelo de autorização simples, flexível, bem definido e de fácil implementação. O SPKI surgiu com uma especificação mais simples como alternativa ao X.509, que conta com várias estruturas globais de implementação mais complexa. O SPKI/SDSI é citado por alguns autores somente como SPKI.

Em SPKI existe uma premissa de que um certificado é válido até a data do seu vencimento, após essa data ele é considerado expirado. Em SPKI existem quatro razões para um certificado ser invalidado [Clarke et al., 2001]:

1. O certificado expirou;
2. A chave privada do emissor foi comprometida;
3. A chave privada do solicitante foi comprometida;
4. O emitente emitiu o certificado incorretamente.

Nesse último caso o processo de emissão pode ser manual e suscetível a falhas. Por isso deve-se ter muito cuidado ao emitir um certificado, pois, sua revogação é uma situação muito indesejável e acaba causando situações incontornáveis. Visto que, desde a descoberta do problema até a execução de uma ação de mitigação, algum mau uso do certificado pode ter sido feito. Usuários do SPKI emitem certificados com período de validade extremamente curtos (em inglês *Short-Lived Certificates*) ou com permissões limitadas [Rivest, 1998].

A revogação de certificados em SPKI é um processo ligado à validação dos certificados. Três categorias de verificação *online* são possíveis[Ellison et al., 1999]: Lista de Certificados Revogados (CRL em inglês *Certificate Revocation List*), revalidação temporária (em inglês *Timed Revalidation*) e validação a cada utilização (em inglês *One-Time Revalidation*). Essas verificações são explicadas abaixo:

- *CRL* – A lista de certificados revogados é uma abordagem comum para a revogação dos certificados. Cada uma dessas listas especifica quais certificados não expirados foram revogados, e quando a próxima CRL será emitida. Uma CRL é considerada vencida após o seu prazo de validade e uma nova lista deverá ser produzida pelo seu emissor.
- *Timed Revalidation* - Enquanto a CRL é uma declaração negativa, a revalidação temporária é positiva. Como se fosse emitido um atestado de saúde para o certificado que atesta que o certificado pode ser considerado válido para o período indicado. Após o período é necessário fazer uma nova verificação para obter um novo atestado de saúde para um novo período.
- *One-Time Revalidation* - O usuário deve verificar a validade do certificado a cada uso.

Conforme descrito em [Jacobson et al., 2009], em NDN não obrigatório a utilização de um modelo de confiança específico. A confiança é estabelecida entre publicadores e consumidores de conteúdo, e o que é apropriado para um aplicativo pode não ser para outro. Os usuários são livres para reutilizar modelos existentes (por exemplo, PKI) para estabelecer a confiança em chaves, ou para definir novos modelos mais apropriados para sua aplicação. Um modelo particularmente adequado para o NDN é o SPKI/SDSI. Algumas abordagens de revogação em NDN, que serão apresentadas na próxima seção, são inspiradas nos métodos de revogação utilizados em SPKI. O SPKI possui uma estrutura mais simples que a PKI, que possui uma séria de processos que a torna complexa, conforme ilustrado na Figura 3.1. Outra característica do SPKI é a possibilidade de trabalhar com espaços de nomes locais que proporciona mais flexibilidade ante ao modelo hierárquico da PKI tradicional.

## 3.3 Revogação de certificados em NDN

Pode ocorrer que as chaves privadas utilizadas para assinar os conteúdos possam ser obtidas de maneira acidental ou maliciosamente por um usuário mal-intencionado. Desse modo, conteúdos falsos podem ser gerados com essas chaves como se fossem legítimos. Esse processo

pode uma série de ataques em NDN, pois, esses dados podem ficar armazenados no *cache* dos roteadores. Em NDN, a verificação da assinatura dos pacotes de dados permite identificar conteúdo falso, porém, a verificação é opcional para os roteadores e obrigatória para os clientes. O fato dos roteadores não serem obrigados a verificar assinaturas os torna vulneráveis a diversas categorias de ataques aos conteúdos, entre eles um ataque chamado *envenenamento de conteúdo* [Gasti et al., 2013]. Nesse ataque, um atacante pode utilizar uma chave comprometida para assinar conteúdos falsos e distribuí-los na rede. Devido a essa situação, o processo de verificação do status da chave torna-se de extrema importância para a validação dos conteúdos.

Devido ao processo de revogação estar associado ao gerenciamento de certificados, esse tema é pouco explorado em ICN de forma isolada. Alguns artigos que tratam do gerenciamento de confiança mencionam o processo de revogação de forma bem sucinta. [Zhang et al., 2011] propõem um sistema de gerenciamento de confiança utilizando criptografia baseada em identidade (*Identity-Based Cryptography* - IBC). Nessa proposta, os certificados são gerenciados por um sistema chamado PKG - *Private Key Generator*. Esse sistema não prevê revogação, uma vez que mensalmente as chaves privadas são atualizadas automaticamente.

[Yu, 2015] define uma abordagem de distribuição de certificados com ênfase nos produtores de conteúdo, apresentando uma proposta de revogação de assinaturas. O autor menciona vários desafios da gestão de chaves públicas em NDN e propõe que os aplicativos utilizem um mecanismo independente de provisionamento de certificados. Sua proposta adota um sistema de provisionamento de certificados, eliminando a necessidade do publicador distribuir sua chave pública. Nesse trabalho, a revogação de uma chave é realizada através de uma *declaração negativa*, também chamada de *atestado de suicídio da chave*, que deve ser feita pelo proprietário da chave e adicionada a um repositório. O objetivo do autor não era o processo de revogação de certificados, já que ele propõe solucioná-lo através da revogação da assinatura com duas abordagens. A primeira delas é pedir ao dono da chave privada que publique uma declaração negativa sobre a assinatura, e a segunda é a de manter a publicação de uma declaração positiva até a assinatura ser revogada ou expirar.

Alguns métodos para rejeitar conteúdos assinados com chaves que foram comprometidas são analisados em [Mauri e Verticale, 2013]. Um dos métodos, chamado de *proativo*, consiste na sincronização de repositórios de chaves. Os outros dois métodos são chamados de *reativos*, sendo que um deles é baseado em *timestamp*. Neste método, o requisitante envia um pacote de interesse solicitando uma chave, especificando um *timestamp* que indica o limite de validade. Quando um nó recebe o interesse, ele verifica se possui a chave e se seu *timestamp* é mais recente que o *timestamp* do interesse. Se a condição for satisfeita, o nó envia o pacote de dados correspondente que contém a chave. Caso contrário, o nó encaminha o interesse para o nó seguinte. Se nenhum nó tem a chave recente, o interesse é encaminhado até o detentor da chave original. O outro método reativo consiste em obter a chave diretamente do publicador, através de um parâmetro do pacote de interesse que indica que o dado deve ser obtido diretamente da sua fonte original. Os autores comparam o desempenho de obtenção das chaves em termos de atraso, taxa de transferência e taxa de acertos, e concluem que a solução baseada em *timestamp* oferece o melhor compromisso entre atraso e sobrecarga da distribuição do status da chave.

A Tabela 3.1 compara alguns aspectos das soluções de revogação proposta para NDN. As soluções são identificadas pelas siglas:

- *S1*: Recuperação da chave diretamente do publicador [Mauri e Verticale, 2013];
- *S2*: Recuperação baseada em *timestamp* [Mauri e Verticale, 2013] ;
- *S3*: Recuperação através de repositórios sincronizados [Mauri e Verticale, 2013];

- *S4*: Revogação de assinaturas através de declaração negativa [Yu, 2015];
- *S5*: Revogação de assinaturas através de declaração positiva [Yu, 2015];
- *S6*: Revogação através de atestado de suicídio da chave [Yu, 2015].

As características avaliadas em cada solução são as seguintes:

- *Distribuído*: Indica quais soluções funcionam de maneira descentralizada, ou melhor, funciona de forma cooperativa em vários equipamentos.
- *Uso do cache*: Indica quais das soluções fazem o uso do *cache* para obter um melhor desempenho.
- Tolerância a falhas: Indica se a solução permite que a consulta possa ser realizada através de outro local caso o servidor principal não esteja disponível.
- *Latência*: Indica a incidência de atraso na recuperação do dado: baixa, média ou alta.
- *Custo computacional*: Indica o nível de utilização de recursos computacionais: baixo, médio ou alto.

Tabela 3.1: Comparativo entre as soluções de revogação em NDN.

| Proposta | S1 | S2 | S3 | S4 | S5 | S6 |
|---|---|---|---|---|---|---|
| Distribuído | Não | Não | Sim | N/E | N/E | N/E |
| Uso do *cache* | Não | Sim | Não | N/E | N/E | N/E |
| Tolerância a falhas | Não | Não | Sim | N/E | N/E | N/E |
| Latência | Alta | Média/Alta | Média | Média | Média | Média |
| Custo computacional | Baixo | Baixo | Baixo | Alto | Alto | Médio |

(N/E) indica "não especificado"

## 3.4 Considerações finais

O objetivo deste capítulo foi apresentar uma visão geral das tecnologias envolvidas no processo de revogação certificados digitais na Internet e em ICN. A Seção 3.1 aborda a infraestrutura de chaves públicas (PKI), que é um conjunto de técnicas, práticas e procedimentos elaborados que deve suportar um sistema criptográfico para emitir, expedir, distribuir, revogar e gerenciar os certificados digitais. A Seção 3.2 aborda os métodos de revogação de certificados mais utilizados em PKI, entre eles o CRL e o OCSP. A CRL tem como vantagem funcionar *offline*, contudo, tem alto risco de ficar desatualizada. O OCSP disponibiliza uma consulta *online* do status do certificado, mas pode sofrer com problemas de disponibilidade. Apresenta também os métodos de revogação utilizados em SPKI. A Seção 3.3 mostra as abordagens sobre revogação em ICN. Uma delas é uma abordagem que faz parte de uma proposta de gerenciamento de certificados, todavia, mais voltado para os produtores de conteúdo. A outra proposta que apresenta uma abordagem na visão dos usuários, expondo três métodos e comparando o seu desempenho em latência, taxa de transferência e taxa de acertos. O capítulo seguinte apresenta uma proposta de um serviço de disseminação da informação do status de certificados digitais em NDN.

# Capítulo 4

# Serviço de consulta do status de chaves

Com base nas abordagens expostas no capítulo anterior, neste será apresentada uma proposta de um serviço de divulgação do status de certificados digitais em ICN inspirado no OCSP, todavia, adaptado para trabalhar nos padrões da NDN. A Seção 4.1 apresenta uma visão geral da proposta. A implementação da proposta resultou em um serviço de OCSP distribuído para NDN. A Seção 4.2 apresenta a arquitetura da proposta. A Seção 4.3 apresenta quais são as limitações da proposta.

## 4.1 Visão geral

Fazer com que os usuários gerenciem corretamente suas chaves criptográficas é um desafio de segurança. Em NDN assume-se que o gerenciamento de chaves se tornou uma prática comum para os usuários. Algumas práticas recomendam o uso de chaves com período de validade relativamente curto, com intuito de restringir a sua exposição. Quanto menos expostas, menor é o risco de ocorrer algum dano potencial em caso de comprometimento das mesmas. O período de validade limitado funciona bem para o modelo de segurança baseado em canais de comunicação, uma vez que eles possuem duração limitada. A cada conexão o usuário deve requisitar uma nova chave, evitando o processo de revogação. Todavia, a utilização de chaves com período de validade curto não seria adequado ao modelo de segurança centrado em dados da NDN. Uma vez que é necessário assinar novamente todos os conteúdos, cada vez que a chave utilizada para assiná-los for renovada. O mais indicado para um modelo centrado em dados de longa duração seria a utilização de chaves com período de validade longo. Contudo, com a exposição das chaves por um período maior de tempo, a implementação de um mecanismo de revogação se faz necessária. Quando uma chave for revogada, é preciso notificar os seus usuários de forma rápida e eficiente.

Diante desse cenário, apresentamos uma proposta de um serviço de divulgação do status de chaves. Esse serviço permite aos usuários consultar o status de uma determinada chave, antes de utilizá-la para validar um pacote de dados. A solução proposta visa atender aplicações que fornecem dados com longa duração. Consequentemente, essas aplicações farão utilização de chaves com um período de validade longo. Logo, essas aplicações podem possuir as chaves necessárias para validar os pacotes de dados instaladas na própria aplicação. Ainda assim, recuperar essas chaves e mantê-las armazenadas para utilização posterior. Para essas aplicações, a utilização de chaves com tempo de vida curto causaria uma carga de trabalho adicional para renová-las frequentemente. Como elas fazem uso das chaves por um período longo, não seria necessário renová-las toda vez que tiverem que validar um pacote de dados vindo do mesmo publicador. Ao invés disso, basta consultar o status da chave através do serviço de consulta

proposto. O serviço proposto poderá ser distribuído pela rede com intuito de reduzir o tempo de resposta e melhorar a disponibilidade da informação, reduzindo a dependência de um único ponto de acesso.

O mecanismo de funcionamento do serviço proposto é baseado no OCSP [Santesson et al., 2013]. O serviço receberá as consultas de informações do status de chaves e responderá com três padrões de respostas:

1. "*Válido*" - Indica uma resposta positiva, que a chave é conhecida pelo serviço e que não foi revogada.

2. "*Revogado*" - Indica que a chave foi revogada. Contudo, o motivo da revogação não é informado com no OCSP tradicional. O serviço considera somente a revogação permanente, e não a temporária.

3. "*Desconhecido*" - Indica que o serviço não conhece a chave solicitada.

A utilização do serviço proposto requer a implementação de dois passos adicionais no processo de validação de um pacote de dados. Os processos são ilustrados através de um fluxograma na Figura 4.1, e explicados abaixo:

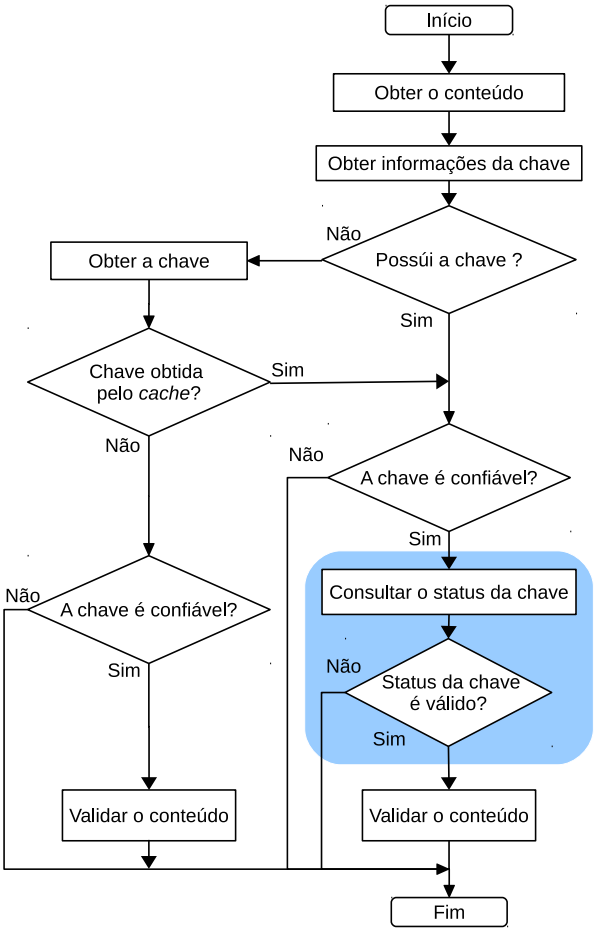


Figura 4.1: Fluxograma de validação de um pacote de dados.

1. Inicialmente, após recuperar o conteúdo a aplicação extrai as informações da chave que serão utilizadas para validá-lo.

2. Após identificar a chave, a aplicação verifica se já a possui. Caso não possua, então é necessário recuperá-la conforme o processo detalhado no Capítulo 2. Se a chave não foi obtida de um *cache* da rede, não é necessário consultar o seu status.

3. Em posse da chave, a aplicação verifica se ela é confiável para verificar o pacote de dados pelo modelo de confiança adotado.

4. A solução proposta introduz mais dois passos adicionais no processo de validação, que estão destacados na Figura 4.1.

    - O primeiro processo consiste na recuperação do status da chave utilizando o serviço de consulta proposto. Logo, a aplicação cliente deverá enviar um pacote de interesse para recuperar a informação do seu status.
    - Após recuperar o status, a aplicação verifica se ele é "*válido*", isto é, verifica se ela não foi revogada.
        - Caso o status seja "*Válido*", a chave poderá ser utilizada para validar o pacote de dados.
        - Caso o status seja "*Revogado*" ou "*Desconhecido*", uma nova chave deverá ser recuperada ou um processo de atualização de toda cadeia de certificação deverá ser executado.

O serviço de consulta proposto poderá ser executado em algum elemento da NDN, que responderá pelo prefixo (por exemplo, `"/ocsp1"`) definido para receber os interesses de consulta. Um pacote de interesse de consulta de status deverá enviado pelo cliente. O interesse especificado deverá conter o prefixo mais a chave a ser consultada. A estrutura interna do serviço de consulta é ilustrada na Figura 4.2. Os seguintes passos são executados pelo serviço após a recepção de um pacote de interesse:

1. Ao receber um pacote de interesse de consulta do status de uma chave (seta 1), primeiramente o ele precisa ser validado com o objetivo de confirmar se está utilizando as especificações corretas do serviço.

2. Após essa validação, uma consulta é efetuada na base de dados de informações de chaves (setas 2 e 3).

3. A partir do resultado da consulta, um pacote de dados com a resposta é construído (seta 4), assinado pelo serviço respondedor (seta 5) e encaminhado de volta para a rede (seta 6).

O processo de tratamento das consultas do status de um certificado pelo serviço proposto, é ilustrado no Algoritmo 1, com os seguintes passos:

1. O primeiro processo a ser executado pelo algoritmo é a validação dos dados de entrada para evitar inconsistências. Caso os dados estejam inconsistentes o pacote é descartado.

2. Após a validação, uma consulta será efetuada na base de dados que retornará a informação sobre o certificado identificado na entrada. Se a consulta retornar o status "*Válido*", então será atribuído a uma variável a informação do tempo em que o sistema recebeu fez contato pela última vez com o detentor das informações sobre esse certificado. Para os demais status, não é necessário especificar a informação de tempo. A utilização desse recurso será explicada na próxima seção.

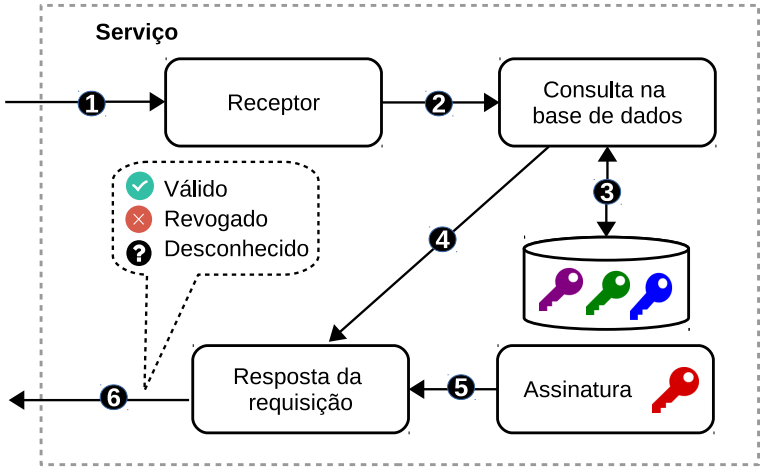


Figura 4.2: Estrutura interna do serviço.

3. O pacote de dados com a resposta é criado. O pacote carrega a identificação da chave, a resposta da consulta, o carimbo de tempo e outras meta informações (conforme especificação da NDN descrita em *MetaInfo* no capítulo 2).

4. Por fim, o pacote de dados de resposta é retornado e a execução do algoritmo é finalizada.

**Algoritmo 1:** MECANISMO DE RECUPERAÇÃO DO STATUS DE UM CERTIFICADO

```
   Entrada: Nome, Certificado
   Saída:  PctDadosResposta
1  início
2      se ValidaEntrada(Nome, Certificado) = Verdadeiro então
3          Resposta ← Consultastatus(Certificado)
4          se Resposta = "BOM" então
5              Tempo←UltimaResposta
6          fim
7          PctDadosResposta ← MontaPacote(Nome, Resposta, Tempo, Metainfo)
             retorna PctDadosResposta
8      senão
9          Descartar o pacote
10     fim
11 fim
```

A Figura 4.3 ilustra o funcionamento geral do serviço proposto. A arquitetura está dividida em: (i) o publicador do conteúdo, (ii) o serviço de consultas e (iii) a aplicação. Quando uma chave é gerada para ser utilizada por um publicador de conteúdo, os seguintes passos são executados desde a sua geração até a sua utilização para validação pela aplicação cliente:

1. Antes de servir o conteúdo para os clientes, o publicador deve primeiro distribuir as chaves que serão utilizadas para validar os conteúdos.

2. As chaves que serão utilizadas para validação são agregadas à aplicação (seta 1) e também enviadas para o serviço de consulta (seta 2).

3. A chave privada, que fica armazenada de forma segura pelo publicador, será utilizada para assinar os conteúdos (setas 3 e 4).

4. Quando os conteúdos são produzidos e assinados, os pacotes de dados deverão possuir a informação de qual chave é necessária para validá-los.

5. Ao instalar a aplicação de acesso aos conteúdos (seta 5), o cliente possuirá as chaves necessárias para validar os conteúdos que serão obtidos por ela.

6. Quando o cliente obtiver um conteúdo (seta 6) para o qual ele já possui a chave para validá-lo (setas 8 e 9), basta apenas obter a informação do seu status, fazendo uma solicitação ao serviço de consulta proposto (seta 7).

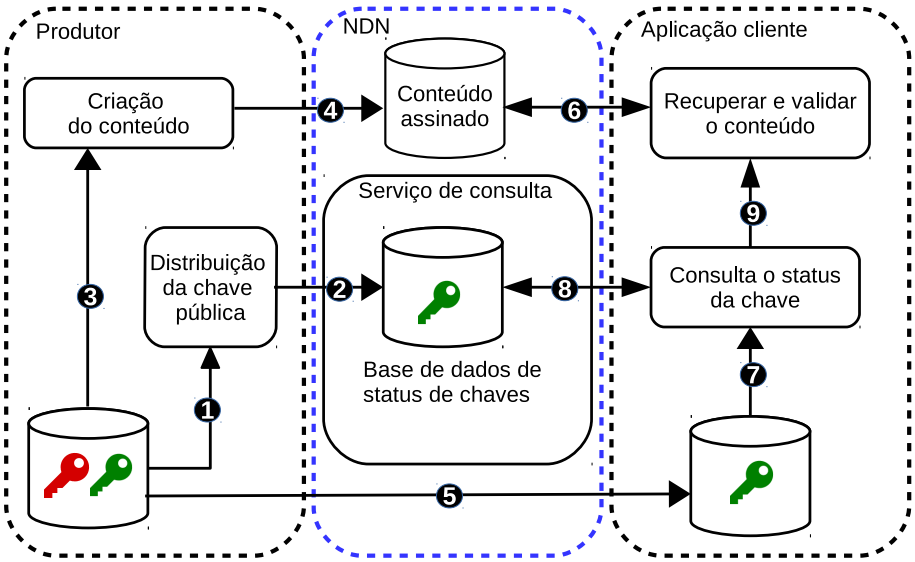


Figura 4.3: Arquitetura proposta.

O serviço de consulta proposto tem como objetivo fazer com que a informação do status dos certificados seja disponibilizada de forma rápida e segura para as aplicações clientes. Por se tratar de um serviço especializado, mecanismos de controle devem ser implementados para garantir a confiabilidade das informações fornecidas. Portanto, conforme especificado no funcionamento do serviço, os pacotes de resposta são assinados pelo serviço de consulta. A aplicação deve conter um conjunto de chaves necessárias para efetuar a validação das respostas emitidas. As ligações de confiança entre os publicadores, o serviço de consulta e a aplicação, devem ser estabelecidas pelo modelo adotado. O modelo de confiança proposto para NDN [Yu et al., 2015] foi adorado, por ser utilizado com padrão. Contudo, os usuários ficam livre para adotar outros modelos que melhor se adaptem às suas aplicações.

## 4.2 Arquitetura

Na seção anterior apresentamos o funcionamento do serviço de consulta de status de chaves/certificados proposto e o descrevemos. Ilustramos também, como as informações do status das chaves são disponibilizadas pelos publicadores, passando pelo serviço de consultas até chegar à aplicação cliente. Um dos objetivos da proposta também é melhorar a disponibilidade das informações de status das chaves. Uma das formas de melhorar a disponibilidade é reduzindo a dependência de um ponto único de acesso às informações. Isso pode ser feito, distribuindo várias cópias do serviço de consulta proposto pela rede. Além de melhorar a disponibilidade, é possível também reduzir a sobrecarga do serviço e reduzir o tempo de resposta. Uma vez que as

cópias do serviço podem ser posicionadas em locais mais próximos dos clientes. Para distribuir as informações do status das chaves para as diversas cópias do serviço espalhados pela rede, adotou-se uma solução de replicação de dados.

O cenário proposto, com várias cópias do serviço de consulta sendo distribuído por replicação, é formado pelos seguintes elementos: *Publicador*, *Coordenadores*, *Réplicas* e *Clientes*. O papel de cada um deles é descrito a seguir, conforme Figura 4.4:

- *Publicador*: (*P*) representa o detentor das chaves utilizadas para assinar e verificar os conteúdos. No cenário da solução proposta é denominamos de *Publicador*, mas pode ser também uma autoridade certificadora ou entidade de confiança dos produtores de conteúdo. É responsável gerenciamento das chaves e pelo processo de enviar atualizações do status das chaves para o serviço de consulta proposto;

- *Coordenadores*: (*C*) responsáveis por receber as informações do status das chaves enviadas pelo *Publicador* e encaminhá-las para as cópias do serviço de consulta (denominadas de *Réplicas*). Faz a função do servidor primário de replicação, responsável por manter as cópias do serviço de consulta sempre atualizadas.

- *Réplicas*: (*R*) recebem as informações de status das chaves dos *Coordenadores*, as mantém armazenadas internamente em uma estrutura de dados e responde as consultas efetuadas pelos *Clientes*.

- *Clientes*: (*Cl*) representam as aplicações clientes que necessitam validar os pacotes de dados. Para isso, precisam consultar se a chave armazenada na aplicação ainda é confiável para validar um determinado conteúdo.

O modelo de replicação adotado é baseado na replicação passiva [Budhiraja et al., 1993]. Nesse modelo os servidores primários são responsáveis pela replicação das informações e não respondem consultas. As consultas são processadas e respondidas pelos servidores secundários. Na arquitetura proposta o servidor primário será identificado como *Coordenador*. Para garantir o funcionamento da solução, o serviço pode conter mais de um *Coordenador*, conforme ilustrado na Figura 4.4. A função do *Coordenador* é manter as *Réplicas* atualizadas. Em caso de indisponibilidade do *Coordenador* principal, um segundo *Coordenador* deve assumir essa função. O *Publicador* possui o papel de um simples usuário. Que consiste em enviar as informações para o serviço de atualização e receber a sua confirmação.

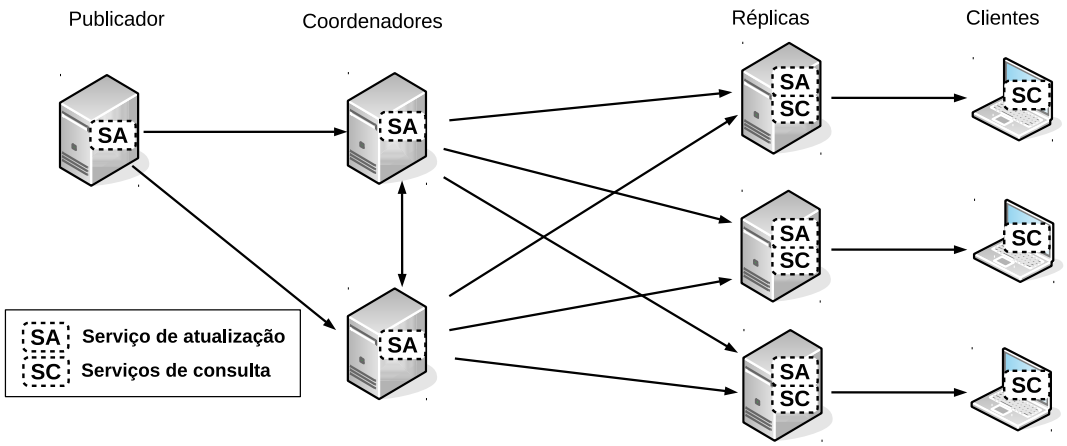


Figura 4.4: Fluxo da replicação das informações.

Para propagar a informação de status de chaves nos moldes da arquitetura NDN, foi adotado um processo baseado em troca de pacotes de interesse e dados. Essas trocas são ilustradas na Figura 4.5:

1. O *Publicador* envia um pacote de interesse para iniciar o processo. O nome do interesse deve identificar o publicador e a chave a ser atualizada.

2. O pacote de interesse é encaminhado para o *Coordenador* da replicação.

3. *Coordenador* envia um pacote de interesse para o *Publicador*, solicitando o status da chave mencionada no interesse de atualização.

4. Este interesse é respondido pelo *Publicador*, com um pacote de dados contendo o status da chave.

5. Ao receber o pacote de dados, o *Coordenador* envia um pacote de dados, correspondente ao primeiro pacote de interesse enviado pelo *Publicador*, confirmando que a atualização foi realizada com sucesso.

6. Este mesmo processo é efetuado entre o *Coordenador* e as *Réplicas*, para propagar a informação disponibilizada pelo *Publicador*.

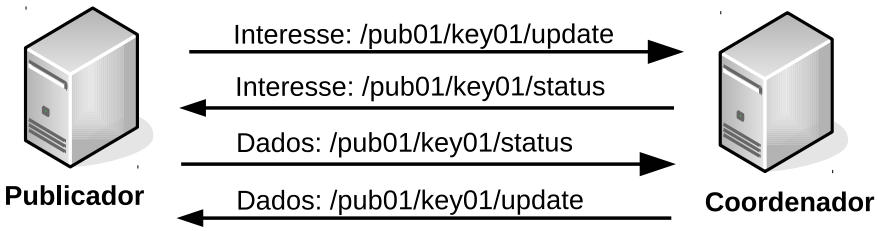


Figura 4.5: Troca de pacotes.

As troca de mensagens do processo de replicação proposto, ilustradas na Figura 4.6, seguem os seguintes passos:

1. O *Publicador* envia uma mensagem de atualização (UP1) para o *Coordenador*.

2. Ao receber a mensagem de atualização, o *Coordenador* solicita a atualização da chave especificada para o *Coordenador*. Para efetuar a atualização, o procedimento ilustrado na Figura 4.5) é realizado.

3. Após receber a atualização, os dados são armazenados pelo *Coordenador*. Em seguida, uma mensagem de atualização (UP1) é enviada para todas as *Réplicas*.

4. Após o *Coordenador* receber a confirmação de atualização (OK) da maioria das *Réplicas*, ele envia uma mensagem de confirmação (OK) para o *Publicador*, conforme ilustra a Figura 4.6 (replicação bem-sucedida).

5. Caso o *Publicador* não tenha recebido a confirmação do *Coordenador* até o *timeout*, ele envia novamente o pacote de interesse de atualização. Essa situação é ilustrada na Figura 4.6 (replicação mal sucedida).

- Esse procedimento tem como objetivo garantir que a maioria das *Réplicas* estejam atualizadas.
- As *Réplicas* que não confirmarem a atualização, continuarão recebendo solicitações de interesse do *Coordenador* até que todas estejam atualizadas.

A detecção de falha das *Réplicas* é feita através de *timeout*. O campo *InterestLifeTime* do pacote de interesse enviado para atualização é utilizado para controlar esse tempo. Uma abordagem para tratar as *Réplicas* desatualizadas será apresentada mais adiante.

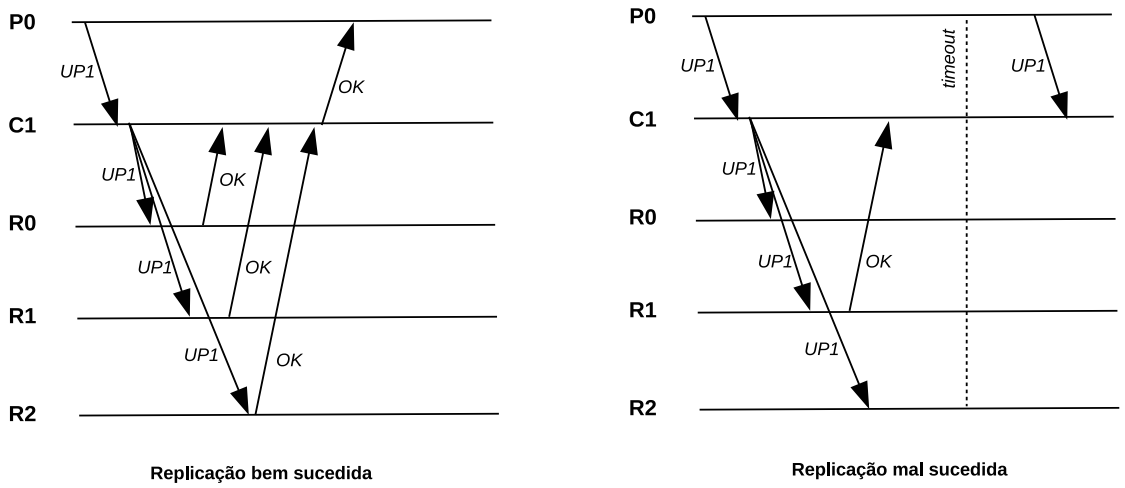


Figura 4.6: Replicação.

As operações de atualização das *Réplicas* sempre serão feitas por um único usuário (o *Publicador*) para o mesmo registro. A atualização consiste em uma operação de escrita referente à inclusão de um novo ou à atualização de um registro existente. Devido a esse perfil operacional, não ocorrerão transações concorrentes para operações de escrita sobre o mesmo registro. A consistência dos dados no processo de escrita está garantida, não sendo necessário utilizar controle de concorrência ou de ordenação de eventos. As operações de leitura sempre retornarão um único registro. Por mais que a solução possa receber um pacote com várias consultas, elas serão processadas individualmente.

O modelo de falhas adotado pela solução é do tipo *crash* sem recuperação [Budhiraja et al., 1993]. O processo de recuperação de falhas precisaria reconstruir a base de dados de informações de chave a partir do *Coordenador* para poder responder às consultas de forma consistente. Enquanto a *Réplica* não estiver atualizada ela não poderá responder consultas de forma consistente. O processo de recuperação poderá ser implementado e testado em um trabalho futuro.

Adotamos uma abordagem para tratar a situação de falhas de atualização ocasionadas por atraso ou falha nos *links* de comunicação, que consiste nos seguintes passos:

1. Periodicamente, a *Réplica* envia um pacote de interesse para o coordenador com o objetivo de monitorar a conectividade entre eles.

2. Caso o *Coordenador* receba esse interesse, ele responde à réplica com um pacote de dados de confirmação.

3. Ao receber a confirmação, a *Réplica* atualiza a informação de carimbo de tempo da troca de comunicação com o *Coordenador*.

4. Ao fazer uma consulta, o *Cliente* recebe também o carimbo de tempo da última troca de comunicação entre a *Réplica* e o *Coordenador* e poderá julgar se esse tempo é aceitável.

5. Caso o tempo de atualização não seja aceitável, o *Cliente* poderá solicitar essa informação de outra *Réplica*.

6. Na situação em que uma *Réplica* perde a comunicação com o *Coordenador*, o *Cliente* pode rapidamente identificar se ela se encontra desatualizada. Pois, através da informação de tempo especificada no pacote, é possível saber quando o *Réplica* teve o último contato com o *Coordenador*.

## 4.3 Considerações finais

Este capítulo apresentou uma descrição detalhada do serviço proposto. Inicialmente foi detalhado o funcionamento desse serviço e como ele se encaixa no processo de validação de um pacote de dados na arquitetura NDN. Em seguida, uma visão geral da arquitetura do serviço é apresentada, ilustrando como o ele é distribuído e atualizado através de replicação. Por fim, as suas limitações também são mencionadas. O capítulo seguinte discutirá o modelo de simulação utilizado para validar a proposta, as métricas que foram utilizadas e os resultados obtidos.

# Capítulo 5

# Validação da proposta

Neste capitulo o serviço de consulta proposto no capítulo anterior é avaliado. A validação é feita através de simulações, e são utilizadas as seguintes métricas: *tempo para atualização* das réplicas, *tempo de resposta* para recuperação da informação, *taxa de resposta das réplicas* e *taxa de informações atualizadas*. A Seção 5.1 apresenta o modelo de simulação adotado juntamente com a topologia utilizada. A Seção 5.2 descreve as métricas utilizadas para a avaliação. A Seção 5.3 apresenta os resultados da avaliação.

## 5.1 Modelo de simulação

Dada a indisponibilidade de uma plataforma ICN real para os experimentos, a proposta foi avaliada através de modelos de simulação. As simulações foram construídas utilizando o módulo ndnSIM versão 2.3 [Mastorakis et al., 2016], do simulador NS-3 (*Network Simulator*) sobre a topologia RocketFuel [Spring et al., 2002]. Essa topologia possui características de grandes *backbones* reais, conforme ilustrado na Figura 5.1. Foi utilizado o *template* NTT (AS2914) ilustrado na Figura 5.2, que é disponibilizado pelo simulador. O *template* possui 269 roteadores, 190 servidores e 461 clientes. Nas simulações foram utilizados cenários com 25, 50, 100, 200 e 400 clientes. A quantidade de réplicas rodando o serviço de consulta variou entre 5, 10, 20, 40 e 80. Essas quantidades foram definidas utilizando um fator de 5% de réplicas em relação à quantidade de clientes. Tendo em vista a quantidade de elementos disponibilizados pelo *template*.

Todas as configurações de cada canal de comunicação fornecidas pelo *template*, como taxa de transferência de dados e atraso, foram mantidas. A política de renovação de *cache* adotada em todos os cenários foi a LRU ( *Least Recently Used*). É a política padrão adotada no simulador que descarta os itens menos usados primeiro. Geralmente, essa política é mais eficiente do que as outras, porque, mantêm os dados acessados mais vezes nos últimos instantes.

## 5.2 Métricas de avaliação

Para esta avaliação serão utilizadas as seguintes métricas: tempo para atualização das réplicas, tempo para recuperação da informação, taxa de resposta das réplicas e taxa de informação atualizada. Essas métricas serão detalhadas a seguir:

**Tempo de atualização (*ut – update time*):** tempo decorrido (em milissegundos) desde o envio da solicitação de atualização do publicador ($t_{req}$, *request*) até a recepção da resposta de

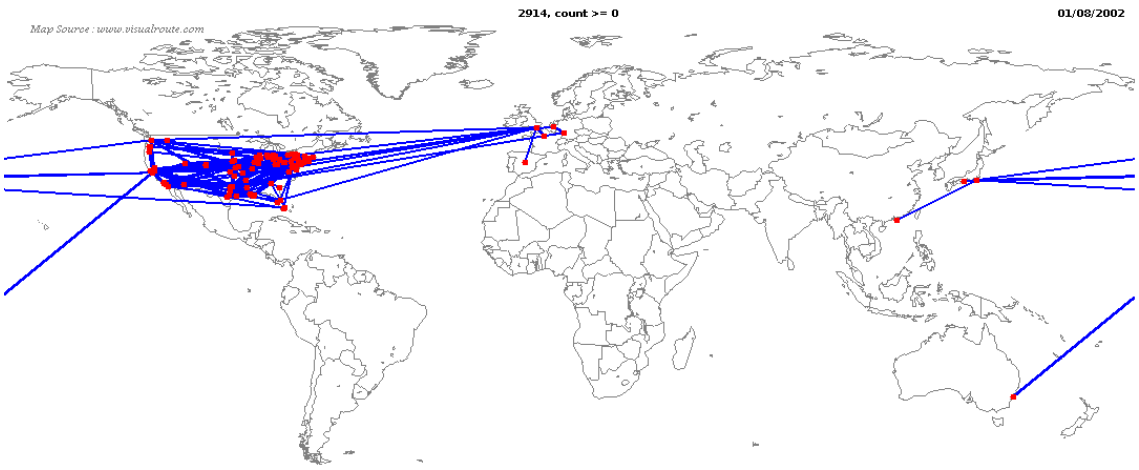


Figura 5.1: Topologia RocketFuel [Spring et al., 2002].

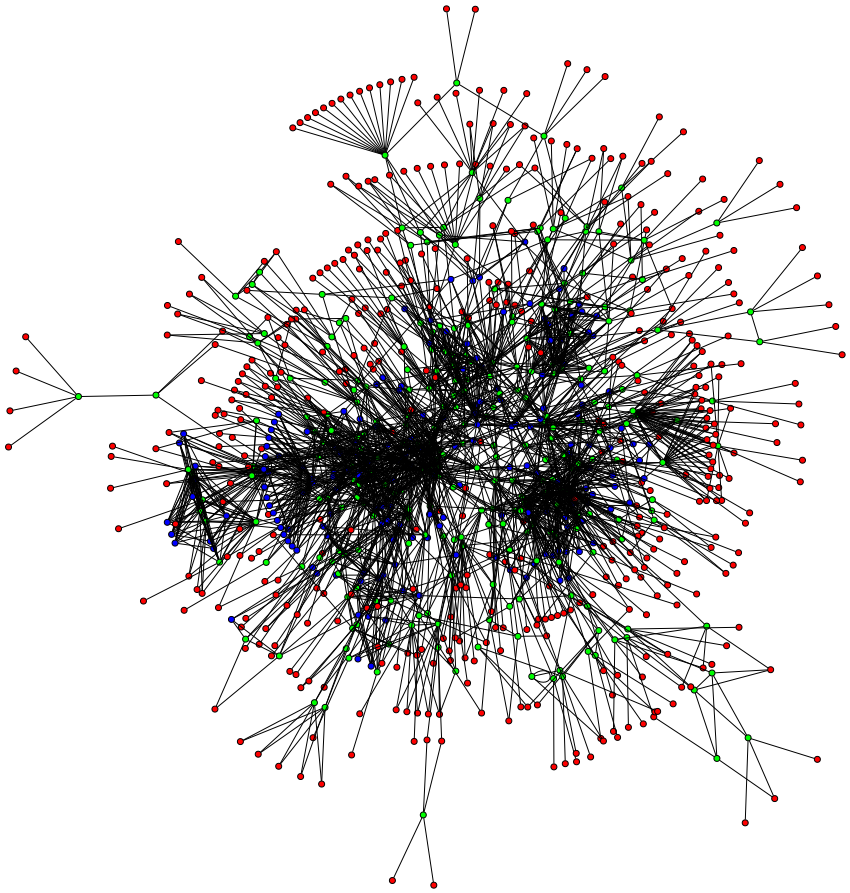

Figura 5.2: Topologia RocketFuel [Spring et al., 2002].

confirmação enviada pelo coordenador da replicação ($t_{rep}$, *reply*), conforme Equação 5.1.

$$ut = (t_{rep} - t_{req}) \tag{5.1}$$

Serão avaliados o menor tempo ($ut_{min}$), o tempo médio ($ut_{med}$) e o maior tempo ($ut_{max}$) necessário para atualizar todas as réplicas, considerando os valores obtidos em todas as rodadas de simulação.

**Tempo de resposta (*rt – response time*):** tempo decorrido (em milissegundos) entre a requisição do status de uma chave por um cliente ($t_{req}$, *request*) e a recepção da resposta correspondente ($t_{rep}$, *reply*), conforme Equação 5.2.

$$rt = (t_{rep} - t_{req}) \tag{5.2}$$

O tempo de resposta médio ($rt_{med}$) corresponde a média dos tempos de resposta de todos os clientes observados em todas as rodadas de simulação.

**Taxa de respostas das réplicas (*rr – response rate*):** razão das respostas obtidas através das réplicas sobre o número de solicitações de consultas. Quando uma resposta não é obtida através de uma réplica significa que ela foi obtida através do *cache*. A taxa de resposta corresponde à razão da quantidade de respostas obtidas ($r_{rep}$) sobre o número de solicitações de consulta ($n$), conforme Equação 5.3.

$$rr = \frac{\sum_{i=1}^{n} r_{rep}(i)}{n}\% \tag{5.3}$$

A taxa média de respostas ($rr_{med}$) corresponde à média dos valores obtidos em todas rodadas de simulação.

**Taxa de informações atualizadas (*ur – update rate*):** razão das respostas atualizadas sobre o número de solicitações de consultas. No intervalo de tempo em que uma réplica ainda não recebeu a atualização, ela pode fornecer uma resposta desatualizada, essa métrica tem objetivo de medir qual o percentual de respostas desatualizada consideram que uma atualização foi disparada no mesmo momento em que o cliente envia uma consulta. A taxa de respostas atualizadas corresponde à razão da quantidade de respostas obtidas ($r_{rep}$) sobre o número de solicitações de consultas ($n$), conforme Equação 5.4.

$$ur = \frac{\sum_{s} um_{i=1}^{n} r_{rep}(i)}{n}\% \tag{5.4}$$

A taxa média de respostas atualizadas ($ur_{med}$) corresponde a média dos valores obtidos nas 35 rodadas de simulação.

Para obtenção dessas métricas foi necessário criar uma aplicação customizada para o ndnSim. Essa aplicação registrava todas as atividades realizadas durante a simulação. Entre elas o tempo em que as solicitações eram enviadas e o tempo em que as repostas eram obtidas. Por fim, a quantidade de solicitações respondidas e a quantidade de respostas atualizadas.

## 5.3 Resultados

Nesta seção são apresentados os resultados obtidos pelas métricas descritas acima. Para cada quantidade de réplicas (5, 10, 20, 40 e 80) foram executadas 35 rodadas de testes com até 400 clientes. O coeficiente de variação das métricas observadas ficou abaixo de 10% em todas as rodadas de testes.

### 5.3.1 Tempo de atualização

O tempo de atualização consiste no tempo total (em milissegundos) desde o envio da solicitação de atualização pelo publicador até o envio de confirmação pelo coordenador da replicação. Na avaliação dessa métrica um conjunto de nós representando os publicadores, os coordenadores de replicação e as réplicas, foram escolhidos aleatoriamente.

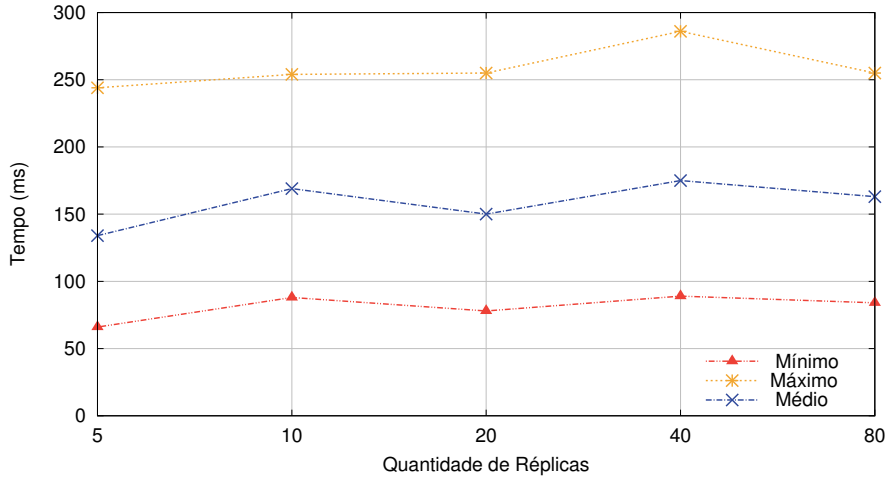


Figura 5.3: Tempo médio para atualização das réplicas.

A Figura 5.3 apresenta o tempo mínimo, médio e máximo necessário para atualizar todas as réplicas com o novo status de uma chave. Os resultados demonstram que a inclusão de mais réplicas não causa muita influência no tempo necessário para a atualização da informação de todas. As simulações não consideram possíveis falhas ou atrasos na atualização, como reenvio da solicitação para uma réplica que falhou ou demorou para responder, ou mesmo a não confirmação pela maioria das réplicas.

A Figura 5.4 apresenta o tempo mínimo, médio e máximo necessário para atualizar uma quantidade de até 400 réplicas. Observando os resultados percebê-ser que uma solução onde os clientes poderiam receber diretamente as atualizações, parece ser viável. Pois, é possível notar pelo gráfico que mesmo com um número maior de elementos, o tempo necessário para atualizar as réplicas, não sofre grandes oscilações. Contudo, pode-ser considerar que o coordenador da replicação deve conhecer todos os clientes que receberão a atualização. Sendo que, para efetuar a atualização de uma chave é necessário utilizar pelo menos quatro mensagens, considerando quando a replicação é bem sucedida. Isso aumentaria o número de mensagens trafegadas e ainda poderia gerar tráfego desnecessário, pois, estaria enviando informação de chaves que os clientes não utilizarão. Enfim, o serviço teria que conhecer a necessidade de cada cliente para enviar somente as informações necessárias sem desperdiçar recursos.

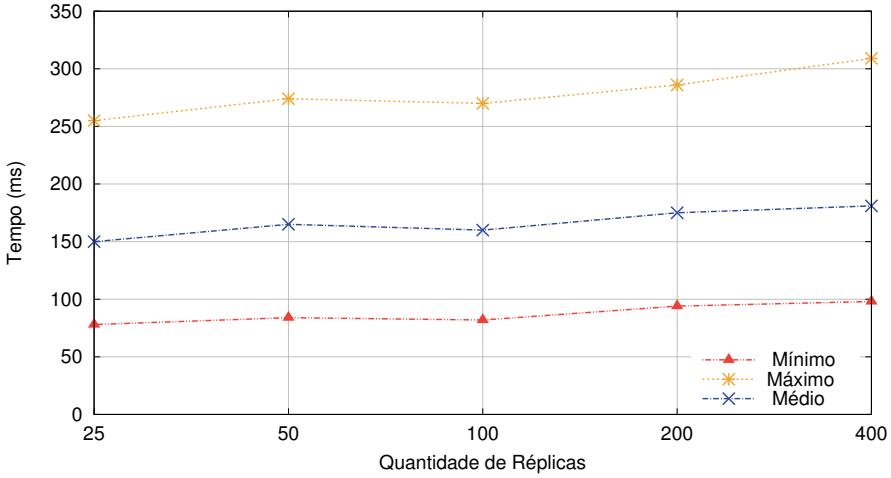


Figura 5.4: Tempo médio para atualização dos clientes.

### 5.3.2 Tempo de resposta

Os experimentos foram realizados com um número inicial de 25 clientes, sendo essa quantidade duplicada sucessivamente até chegar a um total de 400. Inicialmente foram realizados experimentos sem a utilização de réplicas, quer dizer, os clientes obtêm a informação do status das chaves diretamente do publicador. Os demais experimentos foram realizados partindo de 5 réplicas, sendo esse número duplicado até total de 80. Cada experimento com um mesmo conjunto de parâmetros foi executado 35 vezes.

A Figura 5.5 apresenta os tempos de resposta médios observados nos experimentos. Para o serviço de consulta sem réplicas, o tempo médio de resposta com 25 clientes foi de 239,4 milissegundos, reduzindo para 169,8 com 400. Ao adicionar as réplicas do serviço de consulta, o tempo médio diminuiu em todos os cenários. À medida que, mais clientes vão sendo adicionados, observa-se uma diminuição nos tempos médios de resposta. Esse efeito é previsível em NDN, pois, mais requisições para o mesmo conteúdo são agregadas às tabelas de interesses pendentes dos roteadores.

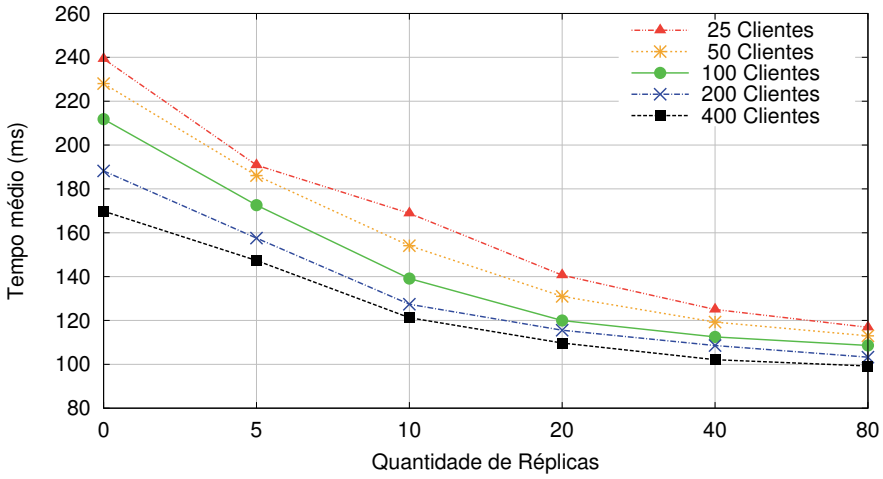


Figura 5.5: Tempos de resposta médios

A Tabela 5.1 detalha os percentuais de redução no tempo para obter o status de um certificado em relação ao tempo de resposta sem réplicas do serviço de consulta. Com a adição

de 5 réplicas, o tempo de resposta para obter o status de um certificado diminuiu em média de 13,2% a 21,3%, podendo atingir uma redução de até 53% com 80 réplicas. Observa-se em todos os cenários que as reduções não são significativas a partir de uma certa quantidade de réplicas. Portanto, a partir de uma determinada quantidade de réplicas não é mais vantajoso adicionar novas réplicas ao sistema. Por exemplo, no cenário com 400 clientes e 40 réplicas, a redução no tempo de resposta é de 39,8% e com 80 réplicas a redução passa a 41,5%, um ganho pequeno frente ao dobro de réplicas necessárias.

Tabela 5.1: Redução no tempo de resposta médio.

| Réplicas \ Clientes | 25 | 50 | 100 | 200 | 400 |
|---|---|---|---|---|---|
| 5 | 21,3% | 18,3% | 18,5% | 16,2% | 13,2% |
| 10 | 30,9% | 32,4% | 34,3% | 32,2% | 28,6% |
| 20 | 43,2% | 42,5% | 43,3% | 37,5% | 35,3% |
| 40 | 50,1% | 47,7% | 46,8% | 42,3% | 39,8% |
| 80 | 53,7% | 50,4% | 48,7% | 45,1% | 41,5% |

### 5.3.3 Taxa de resposta das réplicas

*Taxa de resposta das réplicas* é a razão entre o número de respostas obtidas através das réplicas e o número de solicitações de consultas. Uma vez que o pacote de dados com a resposta de uma consulta é armazenado no *cache* de algum roteador, esse roteador poderá responder a outras solicitações enquanto o *FreshnessPeriod* do pacote de dados não expirar. Nessa situação as réplicas tendem a responder menos solicitações, uma vez que elas podem ser atendidas pelos roteadores. Quando o solicitante não obtém a resposta através de uma réplica, significa que ela foi obtida através do *cache* de algum roteador.

Para obter essa métrica foi realizado simulações com 400 clientes escolhidos aleatoriamente, disparando consultas em cinco cenários diferentes, cada cenário com uma quantidade diferente de réplicas (5, 10, 20, 40 e 80). Foram realizadas 35 rodadas de simulação com um tempo total de 100 segundos. Cada cenário possuía três variações no *FreshnessPeriod* do pacote de dados: (i) o pacote não expira durante o tempo de simulação, (ii) o pacote expira uma vez durante a simulação e (iii) o pacote expira três vezes durante a simulação. A cada 10 segundos a quantidade de solicitações respondidas pelas réplicas é contabilizada. Foram calculados a média e o coeficiente de variação, onde o maior valor do coeficiente de variação observador foi de 1,5%. As médias das requisições atendidas pelas réplicas são mostradas em percentuais de zero a 100 por cento nos três gráficos seguintes, sendo um para cada variação do *FreshnessPeriod* do pacote de dados.

A Figura 5.6 apresenta a média da taxa de resposta (em percentuais de 0 a 100) das requisições atendidas pelas réplicas no cenário em que o pacote de dados expira durante o tempo de simulação. Neste cenário identificamos que, com o passar do tempo, as réplicas tendem a responder menos requisições, uma vez que uma parte das requisições podem ser atendidas pelo *cache*, pois, o *FreshnessPeriod* do pacote de dados não expira durante a simulação.

A Figura 5.7 apresenta a média da taxa de resposta (em percentuais de 0 a 100) das requisições atendidas pelas réplicas no cenário em que o pacote de dados expira uma vez durante o tempo de simulação. Neste cenário identificamos que com o passar do tempo as réplicas tendem a responder menos requisições, porém, após o pacote de dados expirar no *cache* o percentual de

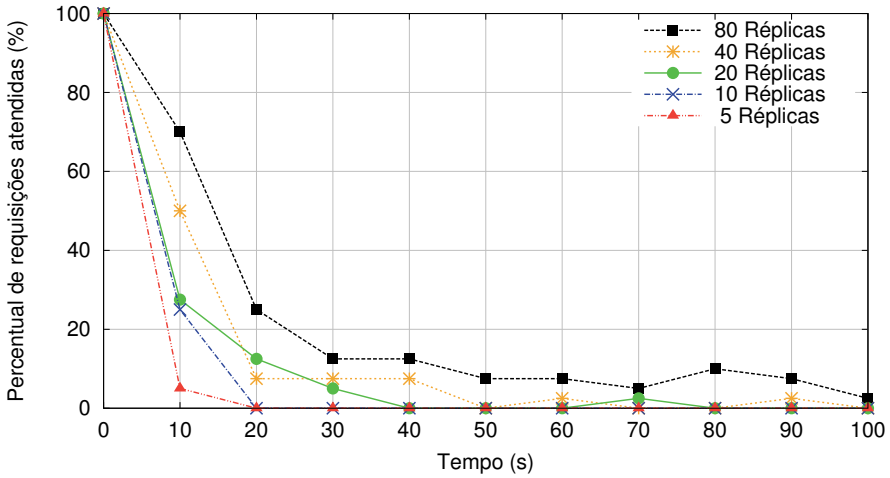


Figura 5.6: Taxa de resposta das réplicas sem renovação.

requisições atendidas pelas réplicas aumenta um pouco e depois começa a cair novamente. Uma vez que uma parte das requisições passam a ser atendidas novamente pelo *cache*.

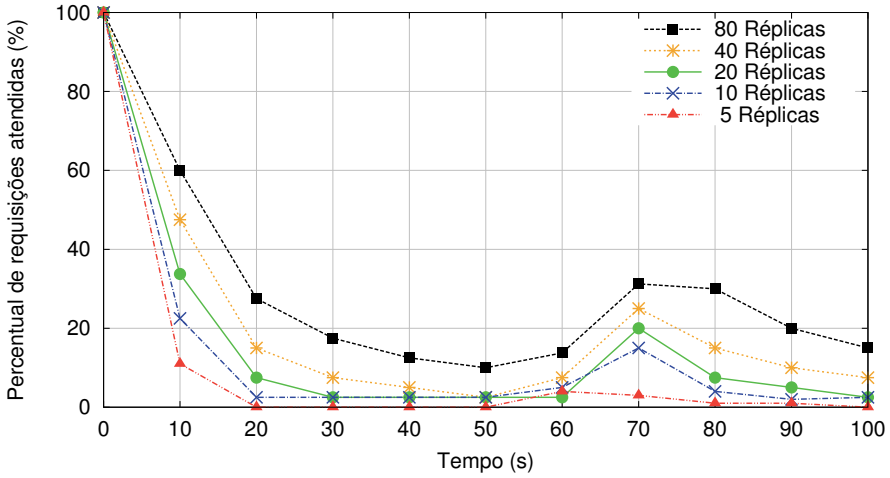


Figura 5.7: Taxa de resposta das réplicas com renovação a cada 50s.

A Figura 5.8 apresenta a média da taxa de resposta (em percentuais de 0 a 100) das requisições atendidas pelas réplicas no cenário em que o pacote de dados em expira três vezes durante o tempo de simulação. Neste cenário identificamos que com o passar do tempo as réplicas tendem a responder menos requisições. Todavia, após o pacote de dados expirar no cache, o percentual de requisições atendidas pelas réplicas aumenta um pouco e depois começa a cair novamente. Contudo, com uma queda um pouco inferior ao cenário anterior.

A Tabela 5.2 exibe a média da taxa de resposta em cada um dos três cenários simulados com as respectivas quantidades de réplicas. Os números possuem muita variação conforme a utilização das respostas em *cache*. Uma quantidade maior de réplicas não necessariamente aumenta a quantidade de informações disponibilizadas pelos *caches*, pois esse fator é influenciado pelo *FreshnessPeriod* do pacote. Quando o pacote de reposta expira, as réplicas tendem a responder mais solicitações de consulta. Todavia, ao realizar a média em todos os cenários essa taxa de resposta fica reduzida. A redução da taxa de reposta em um cenário com 80 réplicas, onde o pacote de resposta expira três vezes durante o tempo de simulação, é de 45% para 14%, ficando com uma média geral de 27%.

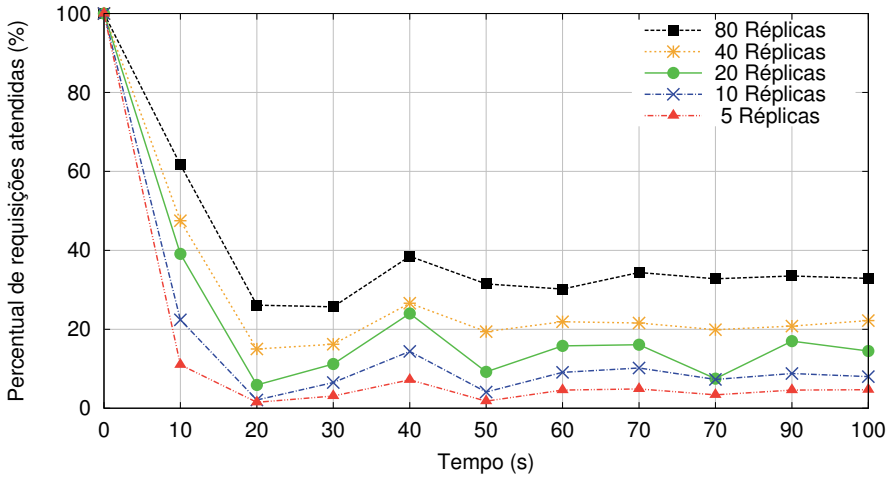


Figura 5.8: Taxa de resposta das réplicas com renovação a cada 20s.

Tabela 5.2: Taxa de resposta da Réplicas.

| Cenário \ Réplicas | 5 | 10 | 20 | 40 | 80 |
|---|---|---|---|---|---|
| i | 0,4% | 2,2% | 4,3% | 7,0% | 14,7% |
| ii | 1,8% | 5,7% | 8,0% | 13,4% | 22,7% |
| iii | 4,9% | 9,5% | 15,1% | 25,2% | 45,3% |
| Média | 2,3% | 5,8% | 9,1% | 15,2% | 27,6% |

### 5.3.4 Taxa de informações atualizadas

Tendo como base os resultados da métrica: tempo de atualização, onde o tempo máximo para atualizar 80 réplicas é de 255 milissegundos. Podemos concluir, que após o final do tempo de atualização as repostas obtidas pelos clientes através das réplicas serão respostas atualizadas. Essa métrica tem como objetivo obter a taxa de respostas atualizadas nas consultas realizadas pelos clientes no momento em que uma atualização de status de uma chave é disparada no sistema e ainda não foi finalizada. As consultas são lançadas aleatoriamente após uma atualização ser disparada pelo publicador, e antes do tempo para atualização ser encerrado. A Figura 5.9 mostra os resultados, que também estão apresentados na Tabela 5.3.

Tabela 5.3: Taxa de Informações Atualizadas.

| Réplicas \ Clientes | 25 | 50 | 100 | 200 | 400 |
|---|---|---|---|---|---|
| 5 | 7,8 % | 10,8 % | 11,1 % | 11,5 % | 12,8 % |
| 10 | 9,7 % | 11,2 % | 12,7 % | 16,1 % | 16,5 % |
| 20 | 10,6 % | 11,3 % | 13,1 % | 18,6 % | 19,3 % |
| 40 | 11,8 % | 12,6 % | 15,3 % | 23,3 % | 26,2 % |
| 80 | 14,2 % | 16,1 % | 22,7 % | 23,9 % | 28,5 % |

De acordo com a Tabela 5.3, durante a atualização no cenário com 400 clientes e 20 réplicas conseguimos uma taxa média de informações atualizada de 19,3%, com 40 réplicas aumentou para 26,2% e com 80 réplicas para 28,5%. Essa métrica indica que o aumento do número de réplicas diminui a possibilidade de um cliente obter uma resposta desatualizada.

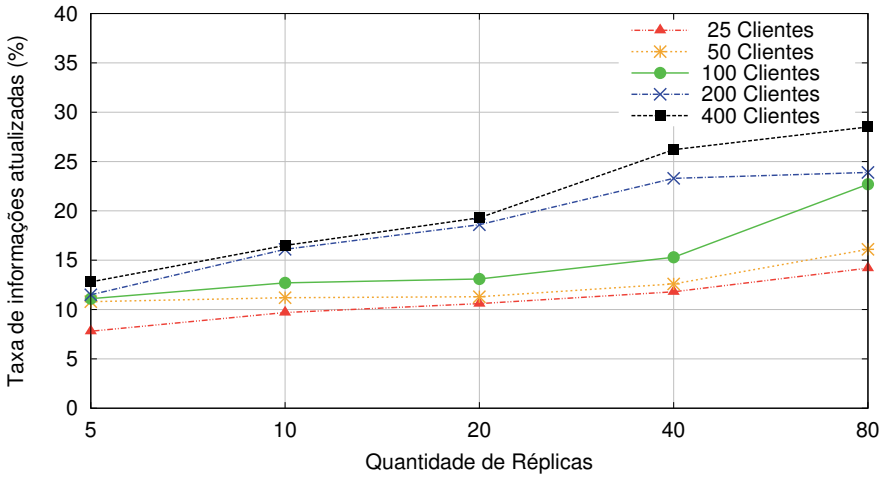


Figura 5.9: Taxa de informações atualizadas.

## 5.4 Considerações finais

O objetivo deste capítulo foi avaliar a solução proposta de divulgação do status de certificados em vários cenários. Na métrica *tempo de atualização*, os resultados demonstram que a inclusão de mais réplicas não causa muita influência no tempo necessário para a atualização da informação em todas as réplicas na rede. Na métrica *tempo de resposta*, demostra que com a utilização de réplicas os clientes conseguem recuperar de forma mais rápida o status de um certificado. Observa-se também que em todos os cenários que os ganhos não são significativos a partir de uma certa quantidade de réplicas. Portanto, a partir de uma determinada quantidade de réplicas não é mais viável adicionar novas réplicas ao sistema. Na métrica *taxa de resposta das réplicas*, a taxa de resposta varia de acordo com *FreshnessPeriod* do pacote de resposta. Quando o pacote de repostas expira, as réplicas tendem a responder mais solicitações de consulta. Contudo, ao observar a média em todos os cenários essa taxa de resposta fica reduzida. A métrica *taxa de informações atualizadas* demonstrou que quanto mais réplicas, menor é a possibilidade de um cliente obter uma resposta desatualizada.

# Capítulo 6

# Conclusão e trabalhos futuros

Este trabalho apresentou uma proposta de um serviço de consulta do status de chaves em redes ICN, inspirado no protocolo OCSP [Santesson et al., 2013]. Esse serviço é baseado em um esquema de replicação passiva, onde as réplicas são espalhadas pela rede e atendem as consultas dos clientes. O serviço de consulta proposto foi avaliado através de um modelo de simulação, usando o módulo ndnSIM do simulador NS-3 [Mastorakis et al., 2016] e o template NTT (AS2914) da topologia RocketFuel [Spring et al., 2002]. Os experimentos realizados demonstram que a arquitetura proposta foi eficiente na redução do tempo médio de obtenção da informação do status das chaves pelos clientes. Através do serviço de consulta distribuído, os clientes conseguem obter o status de um certificado mais rapidamente e com melhor disponibilidade. O serviço possui baixa complexidade, pois é baseado em um modelo simples de replicação, trabalhando com o modelo de falhas *crash* sem recuperação.

Através de simulações, foi possível avaliar as métricas propostas. Na métrica *tempo de atualização*, os resultados demonstram que a inclusão de mais réplicas não causa muita influência no tempo necessário para a atualização da informação em todas as réplicas na rede. A métrica tempo de resposta demostra que os clientes conseguem obter uma resposta mais rápida às consultas de status de certificados através das réplicas do que consultado diretamente o publicador. Observa-se também que em todos os cenários que os ganhos não são significativos a partir de uma certa quantidade de réplicas disponíveis para consulta. Portanto, a partir de uma determinada quantidade de réplicas não é mais viável adicionar novas réplicas ao sistema. Na métrica taxa de resposta das réplicas, a taxa de resposta varia de acordo com *FreshnessPeriod* do pacote de resposta. Quando o pacote de reposta expira, as réplicas tendem a responder mais solicitações de consulta. Contudo, ao realizar a média em todos os cenários essa taxa de resposta fica reduzida. A métrica *taxa de informações atualizadas* demonstrou que quanto mais réplicas, menor é a possibilidade de um cliente obter uma resposta desatualizada. Uma conclusão de forma geral é que para essa solução o *cache* pode influenciar no tempo de obtenção de uma resposta atualizada no momento da atualização. Em um cenário em um fator de réplicas por cliente seja maior, é mais indicado que os conteúdos expirem para que possam ser recuperado através das réplicas. Com exceção da informação de revogação, que é perpétua, que pode ficar de forma permanente nos *caches*.

Deve-se observar que em NDN é recomendado que todos os pacotes de dados devem ser assinados, a omissão deste processo cria uma vulnerabilidade no sistema. Devido a isso, na solução proposta as réplicas são obrigadas a assinar todos os pacotes de dados utilizando sua chave privada, cuja chave pública é instalada nas aplicações clientes. Caso essa chave privada seja comprometida, todo o sistema fica vulnerável. Também, quando o prazo de validade dessas chaves expirarem, será necessário atualizar todo o sistema e todos os clientes.

Uma parte deste trabalho foi publicada em um artigo no Workshop de Redes P2P, Dinâmicas, Sociais e Orientadas a Conteúdo do XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos de 2017 [Rezende et al., 2017].

Algumas aplicações foram desenvolvidas para obter as métricas de avaliação. Uma das aplicações utiliza uma técnica de replicação para distribuir as atualizações do status dos certificados para as réplicas de consultas. Essas duas aplicações trabalham na parte de retaguarda do sistema e são instaladas no publicador, no coordenador da replicação e nas réplicas de consulta. Outra aplicação servidora, responde as consultas solicitadas pela aplicação cliente, que foi incorporada ao processo de validação dos pacotes de dados. Essas aplicações são instaladas nas réplicas de consulta e nos clientes.

Como trabalhos futuros, pretende-se realizar outros experimentos em diferentes topologias, com uma quantidade maior de elementos e outros parâmetros de simulação. Também, efetuar testes de disponibilidade do serviço de consulta em caso de falha das réplicas, definindo quais são as classes de falhas que desejamos tolerar e avaliando se é possível aplicar um modelo de recuperação. Efetuar simulações para avaliar o comportamento da solução em aplicações que trabalham com transporte multi-caminhos. Por fim, estudar um método para validar as chaves utilizadas pelas réplicas para assinar os conteúdos.

# Referências Bibliográficas

[Ahlgren et al., 2012] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D. e Ohlman, B. (2012). A Survey of Information-Centric Networking. *IEEE Communications Magazine*, 50(7):26–36.

[Bhimani, 1996] Bhimani, A. (1996). Securing the Commercial Internet. *Commun. ACM*, 39(6):29–35.

[Budhiraja et al., 1993] Budhiraja, N., Marzullo, K., Schneider, F. B. e Toueg, S. (1993). The Primary-Backup Approach. *Distributed systems*, 2:199–216.

[Clarke et al., 2001] Clarke, D., Elien, J.-E., Ellison, C., Fredette, M., Morcos, A. e Rivest, R. L. (2001). Certificate Chain Discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322.

[Cooper et al., 2008] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. e Polk, W. (2008). Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280.

[Deacon e Hurst, 2007] Deacon, A. e Hurst, R. (2007). The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments. RFC 5019.

[Diffie e Hellman, 1976] Diffie, W. e Hellman, M. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

[Eastlake et al., 2011] Eastlake, D. et al. (2011). Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066.

[Ellison et al., 1999] Ellison, C., Frantz, B., Lampson, B., Rivest, R. L., Thomas, B. M. e Ylonen, T. (1999). SPKI Certificate Theory. RFC 2693.

[Freier, 1996] Freier, A. (1996). The SSL Protocol Version 3.0. RFC 6101.

[Gasti et al., 2013] Gasti, P., Tsudik, G., Uzun, E. e Zhang, L. (2013). DoS and DDoS in Named Data Networking. Em *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, páginas 1–7. IEEE.

[Ghodsi et al., 2011] Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B. e Wilcox, J. (2011). formation-centric Networking: Seeing the Forest for the Trees. Em *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, páginas 1:1–1:6, New York, NY, USA. ACM.

[Housley et al., 1999] Housley, R., Ford, W., Polk, W. e Solo, D. (1999). Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459.

[Hunt, 2001] Hunt, R. (2001). Technological Infrastructure for PKI and Digital Certification. *Computer communications*, 24(14):1460–1471.

[Jacobson et al., 2012] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M., Briggs, N. e Braynard, R. (2012). Networking Named Content. *Communications of the ACM*, 55(1):117–124.

[Jacobson et al., 2009] Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. e Braynard, R. L. (2009). Networking Named Content. Em *5th international conference on Emerging networking experiments and technologies (CoNEXT'09)*, páginas 1–12.

[Kohnfelder, 1978] Kohnfelder, L. M. (1978). *Towards a Practical Public-Key Cryptosystem*. Tese de doutorado, Massachusetts Institute of Technology.

[Koponen et al., 2007] Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S. e Stoica, I. (2007). A Data-Oriented (and Beyond) Network Architecture. Em *ACM SIGCOMM Computer Communication Review*, volume 37, páginas 181–192.

[Madson e Glenn, 1998] Madson, C. e Glenn, R. (1998). The use of HMAC-SHA-1-96 within ESP and AH. RFC 2404.

[Mastorakis et al., 2016] Mastorakis, S., Afanasyev, A., Moiseenko, I. e Zhang, L. (2016). ndnSIM 2: An Updated NDN Simulator for NS-3. Technical Report NDN-0028, NDN.

[Mauri e Verticale, 2013] Mauri, G. e Verticale, G. (2013). Distributing Key Revocation Status in Named Data Networking. Em *Meeting of the European Network of Universities and Companies in Information and Communication Engineering*, páginas 310–313.

[NDN, 2010] NDN (2010). Named data networking project. `http://www.named-data.net/`. Último acesso: 18/04/2017.

[NDN, 2014] NDN (2014). NDN specification documentation. `http://named-data.net/wp-content/uploads/2013/11/packetformat.pdf`. Último acesso: 25/05/2017.

[Polk et al., 2002] Polk, W., Housley, R. e Bassham, L. (2002). Algorithms and Identifiers for the Internet X. 509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3279.

[PURSUIT, 2010] PURSUIT (2010). FP7 PURSUIT Project. `http://www.fp7-pursuit.eu/PursuitWeb/`. Último acesso: 03/05/2017.

[Rezende et al., 2017] Rezende, D., Maziero, C. e Mannes, E. (2017). Distribuindo e consultando o estado de chaves públicas em redes centradas na informação. Em *Workshop de Redes P2P, Dinâmicas, Sociais e Orientadas a Conteúdo – Simpósio Brasileiro de Redes de Computadores (WP2P2+ SBRC)*, páginas 33–46, Belém PA.

[Rivest, 1998] Rivest, R. L. (1998). Can we Eliminate Certificate Revocation Lists? Em *International Conference on Financial Cryptography*, páginas 178–183.

[Rivest e Lampson, 1996] Rivest, R. L. e Lampson, B. (1996). SDSI - A Simple Distributed Security Infrastructure.

[Rivest et al., 1978] Rivest, R. L., Shamir, A. e Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126.

[SAIL, 2010] SAIL (2010). Sail project. `http://www.sail-project.eu/`. Último acesso: 03/05/2017.

[Santesson et al., 2013] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S. e Adams, C. (2013). X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560.

[Spring et al., 2002] Spring, N., Mahajan, R. e Wetherall, D. (2002). Measuring ISP topologies with rocketfuel. *ACM SIGCOMM Computer Communication Review*, 32(4):133–145.

[Turner et al., 2009] Turner, S., Housley, R., Polk, T., Brown, D. R. e Yiu, K. (2009). Elliptic Curve Cryptography Subject Public Key Information. RFC 4949.

[Yu, 2015] Yu, Y. (2015). Public Key Management in Named Data Networking. Technical Report NDN-0029, NDN.

[Yu et al., 2015] Yu, Y., Afanasyev, A., Clark, D., Jacobson, V., Zhang, L. et al. (2015). Schematizing Trust in Named Data Networking. Em *2nd ACM - International Conference on Information-Centric Networking*, páginas 177–186.

[Zhang et al., 2014] Zhang, L., Afanasyev, A., Burke, J., Jacobson, V., Crowley, P., Papadopoulos, C., Wang, L., Zhang, B. et al. (2014). Named Data Networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73.

[Zhang et al., 2011] Zhang, X., Chang, K., Xiong, H., Wen, Y., Shi, G. e Wang, G. (2011). Towards Name-Based Trust and Security for Content-Centric Network. Em *19th IEEE International Conference on Network Protocols*, páginas 1–6.