

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

HERSON HERACLES BARRETO E DAMASCENO

**DICOMAddress: modelo distribuído de endereçamento de imagens
para uma infraestrutura de informação telerradiológica**

João Pessoa
2018

HERSON HERACLES BARRETO E DAMASCENO

**DICOMAddress: modelo distribuído de endereçamento de imagens
para uma infraestrutura de informação telerradiológica**

Dissertação de Mestrado apresentada
ao Programa de Pós-Graduação em
Informática do Centro de Informática
da Universidade Federal de Paraíba
como parte dos requisitos para
obtenção do título de Mestre em
Informática.

Orientador: Prof. Dr. Gustavo Henrique Matos Bezerra Motta

João Pessoa

2018

D155d Damasceno, Herson Heracles Barreto e.

DICOMAddress : modelo distribuído de endereçamento de
imagens para uma infraestrutura de informação
telerradiológica / Herson Heracles Barreto e Damasceno.

- João Pessoa, 2018.

80 f. : il.

Orientação: Gustavo Henrique Matos Bezerra Motta.

Dissertação (Mestrado) - UFPB/CI/PPGI.

1. Infraestrutura de Informação. 2. Imagens Médicas. 3.
Named Data Networking. I. Motta, Gustavo Henrique Matos
Bezerra. II. Título.

UFPB/BC

FOLHA DE APROVAÇÃO

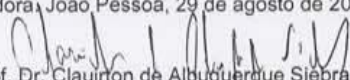


UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA



Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Herson Heracles Barreto e Damasceno, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 29 de agosto de 2018.

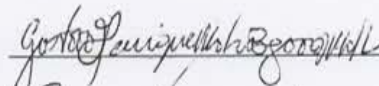
1 Aos vinte e nove dias do mês de agosto, do ano de dois mil e dezoito, às dez horas, no
2 Centro de Informática da Universidade Federal da Paraíba, em Mangabeira, reuniram-se os
3 membros da Banca Examinadora constituída para julgar o Trabalho Final do Sr. Herson
4 Heracles Barreto e Damasceno, vinculado a esta Universidade sob a matrícula nº
5 20161005699, candidato ao grau de Mestre em Informática, na área de "Sistemas de
6 Computação", na linha de pesquisa "Computação Distribuída", do Programa de Pós-
7 Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora
8 foi composta pelos professores: Gustavo Henrique Matos Bezerra Motta (PPGI-UFPB),
9 Orientador e Presidente da Banca, Fernando Menezes Matos (PPGI-UFPB), Examinador
10 Interno, Marco Antônio Gutierrez (USP), Examinador Externo à Instituição. Dando início aos
11 trabalhos, o Presidente da Banca, cumprimentou os presentes, comunicou aos mesmos a
12 finalidade da reunião e passou a palavra ao candidato para que fizesse a exposição oral do
13 trabalho de dissertação intitulado "DICOMAddress: modelo distribuído de endereçamento de
14 imagens para uma infraestrutura de informação telerradiológica". Concluída a exposição, o
15 candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "**aprovado**". A
16 banca deliberou que a versão final da dissertação deverá incluir as seguintes revisões: 1)
17 ampliação dos trabalhos relacionados para incluir artigos com propostas de soluções para o
18 problema da distribuição de imagens médicas, em particular, federação para DICOM; 2)
19 Apresentar de forma aprofundada o mecanismo de propagação de nomes em redes NDN
20 quando da mudança de localidade ou criação de objetos, incluindo o armazenamento
21 distribuído do banco de dados de associações; 3) Explorar possibilidades hierarquização do
22 espaço de nomes no NDN para aplicação no endereçamento de imagens médicas DICOM;
23 4) Realizar experimento testando o funcionamento do endereçamento quando da mudança
24 de localidade de objetos; e 5) Incluir discussão em relação aos trabalhos relacionados e a
25 tecnologia "block chain". O candidato tem o prazo de três meses, a partir de hoje, para
26 entregar a versão final da dissertação para anuência das revisões pelo orientador. Do
27 ocorrido, eu, Claurton de Albuquerque Siebra, Coordenador do Programa de Pós-
28 Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos
29 membros da banca examinadora, João Pessoa, 29 de agosto de 2018.

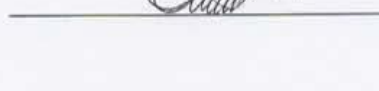

Prof. Dr. Claurton de Albuquerque Siebra

Prof. Dr. Gustavo Henrique Matos Bezerra Motta
Orientador (PPGI-UFPB)

Prof. Dr. Fernando Menezes Matos
Examinador Interno (PPGI-UFPB)

Prof. Dr. Marco Antônio Gutierrez
Examinador Externo à Instituição (USP)





RESUMO

DAMASCENO, Herson Heracles Barreto. **DICOMAddress: modelo distribuído de endereçamento de imagens para uma infraestrutura de informação telerradiológica.** 2018. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal da Paraíba, João Pessoa, 2018.

É sabido que a utilização da telerradiologia está cada vez mais presente em hospitais e clínicas e que estudos são localizados entre eles via URLs. É importante salientar que estes estudos precisam estar disponíveis por vários anos (décadas) e tais URLs devem permitir o acesso a eles no futuro, mesmo que a localização efetiva dos estudos tenha sido mudada. Este trabalho propõe um modelo de endereçamento de estudos (exames) de imagens médicas para uma infraestrutura de informação telerradiológica. Utiliza o protocolo *Named Data Networking* (NDN), que tem como premissa básica a nomeação de objetos para sua descoberta e recuperação, ou seja, o endereçamento deixa de ser baseado em localização e passa a ser apoiado no nome dado a cada objeto. O objetivo deste trabalho é propor o DICOMAddress, uma camada que faz o elo entre o DICOMFlow e a implementação do NDN, o NDN Forwarder Daemon, para a localização de objetos trafegados sem modificar o fluxo de trabalho. Ele pretende ser incorporado ao protocolo de distribuição de estudos médicos DICOMFlow, que é assíncrono, distribuído e utiliza URLs para a localização de estudos. Este modelo não pretende mudar a forma de transferência de imagens, e sim tem o objetivo de nomear estes exames para que possam gerar uma URL válida para a transferência destes. Para a validação do modelo, foi criado um protótipo onde foram efetuados experimentos em ambiente simulado a fim da validação do modelo proposto. Concluiu-se que o modelo proposto atende às necessidades do DICOMFlow e obteve êxito ao retornar a URL com a localização das imagens requisitadas.

Palavras-Chave: Infraestrutura de Informação. Imagens Médicas. *Named Data Networking*. Telerradiologia.

ABSTRACT

DAMASCENO, Herson Heracles Barreto. **DICOMAddress: Modelo distribuído de endereçamento de imagens para uma infraestrutura de informação telerradiológica.** 2018. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Universidade Federal da Paraíba, João Pessoa, 2018.

It is known that the use of teleradiology is increasingly present in hospitals and clinics and that studies are located between them via URLs. It is important to note that these studies need to be available for several years (decades) and such URLs must allow access to them in the future, even if the actual location of the studies has been changed. This work proposes a model for addressing studies (exams) of medical images for a teleradiological information infrastructure. It uses the Named Data Networking (NDN) protocol, whose basic premise is the naming of objects for their discovery and retrieval, that is, the addressing is no longer based on location and is now supported in the name given to each object. The objective of this work is to propose the DICOMAddress, a layer that makes the link between the DICOMFlow and the NDN implementation, the NDN Forwarder Daemon, for locating moved objects without modifying the workflow. It is intended to be incorporated into the DICOMFlow distribution protocol for medical studies, which is asynchronous, distributed and uses URLs to locate studies. This template is not intended to change the way images are transferred, but rather aims to name these exams so that it can generate a valid URL to transfer these images. For the validation of the model, a prototype was created where experiments were carried out in a simulated environment in order to validate the proposed model. It was concluded that the proposed model meets the needs of DICOMFlow and was successful in returning the URL with the location of the requested images.

Keywords: Information Infrastructure. Medical Imaging. Named *Data* Networking. Teleradiology.

Lista de Figuras

Figura 1 – Modelo Arquitetural do DICOMFlow	11
Figura 2 – Fluxo de comunicação em um cenário de uso	11
Figura 3 – Estruturas de dados do NDN	16
Figura 4 – Novo fluxo de comunicação com o modelo incluso	33
Figura 5 – Arquitetura do modelo proposto	36
Figura 6 – Fluxo de geração e envio de nomes	38
Figura 7 – Fluxo de recebimento e busca de Nome	39
Figura 8 – Fluxo de solicitação de URL	42
Figura 9 – Fluxo de resposta a solicitação de URL	43
Figura 10 – Exemplo de funcionamento do DICOMFlow	44
Figura 12 – Fluxo com localização alterada	48
Figura 13 – Fluxo da mensagem e pacotes <i>Interest</i> e <i>Data</i>	50
Figura 14 – Método da classe URIGenerator	54
Figura 15 – Classe URIGenerator	55
Figura 16 – Classe SendInterestService	56
Figura 17 – Definição dos <i>Jobs</i>	58
Figura 18 – Pacotes adicionados ao novo projeto	59
Figura 19 – Fluxo do teste preliminar de funcionamento do DICOMAddress	60
Figura 20 – Fluxo do segundo teste de funcionamento do DICOMAddress	61
Figura 21 – Funcionamento do Bash Ubuntu no Windows 10	63

Lista de Tabelas

Tabela 1 – Serviços do DICOMFlow e suas principais ações.	14
Tabela 2 – Detalhamento dos <i>Jobs</i> e suas ações.....	57
Tabela 3 – Configuração das máquinas do experimento	62

Lista de Siglas

BMS	Building Management Systems
CCN	Content-Centric Networking
CS	Content Store
DICOM	Digital Imaging and Communications in Medicine
DOI	Digital Object Identifier
DoS	Denial of Service
FIB	Forwarding Information Base
HTTP	Hypertext Transfer Protocol
ICN	Information-Centric Networking
IMAP	Internet Message Access Protocol
IoT	Internet of Things
IP	Internet Protocol
JAVA	Linguagem de Programação Orientada a Objetos
LOCKSS	Lots of Copies Keep Stuff Safe
NDN	Named <i>Data</i> Networking
PACS	Picture Archiving and Communication Systems
PIT	Pending <i>Interest</i> Table
REST	Representational State Transfer
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
UDP	User <i>Datagram</i> Protocol
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
WBAN	Wireless Body Area Network
XDS-I	Cross-Enterprise Document Sharing for Image
XML	Extensible Markup Language

Sumário

1. Introdução.....	1
1.1. Objetivos	4
1.1.1. Objetivo geral.....	4
1.1.2. Objetivos específicos.....	5
1.2. Justificativa.....	5
1.3. Metodologia.....	6
1.4. Organização do trabalho	7
2. Referencial Teórico.....	8
2.1. Infraestrutura de Informação	8
2.2. DICOMFlow.....	10
2.2.1. Protocolo de Serviços.....	13
2.3. <i>Named Data Networking</i>	15
2.3.1. Nomes	17
2.3.2. Roteamento e encaminhamento.....	18
2.4. Propagação de nomes	19
2.4.1. Roteamento baseado em OSPF.....	19
2.4.2. Roteamento de camada de enlace em NDN	20
2.5. NDN Forwarding Daemon	21
2.5.1. Faces.....	22
2.5.2. Tabelas.....	23
2.5.3. Estratégias de Encaminhamento	23
2.5.4. Armazenamento e Rede.....	24
2.6. Considerações finais.....	24
3. Trabalhos relacionados.....	25
3.1. Considerações finais.....	30
4. O modelo DICOMAddress	32
4.1. Arquitetura.....	35
4.1.1. URIGenerator	36
4.1.2. SendInterest	40
4.1.3. RouteRegister e PrefixRegister	40
5. Prototipação e testes	53
5.1. Experimento preliminar	59

6. Discussão	64
6.1. Limitações	65
6.2. Considerações finais	65
7. Referências Bibliográficas	67

1. Introdução

O compartilhamento de estudos (exames) de imagens médicas traz benefícios, tornando possível sua aplicação em diversos cenários, como por exemplo: terceirização de análise de imagens radiológicas, valendo-se da contratação de profissionais, sendo possível adquirir os serviços de especialistas de outros países, se beneficiando dos diferentes horários locais (MOTTA, 2014).

Dentre as muitas vantagens da telerradiologia, também é possível destacar a criação de relatórios médicos especializados à distância para pacientes em áreas de difícil acesso, os quais, em circunstâncias normais, seriam incapazes de tirar proveito destes exames. Casos de emergência também podem receber cobertura, uma vez que o processo pode ser executado rapidamente, em um dado momento. Além disso, há uma considerável redução de custos financeiros, incluindo, por exemplo, os relacionados aos médicos que precisam percorrer grandes distâncias dos grandes centros de saúde (FIGUEIREDO; MOTTA, 2013).

No entanto, para que haja o compartilhamento de imagens, faz-se necessária a adoção de alguns padrões. O padrão DICOM (*Digital Imaging and Communications in Medicine* – Comunicação de imagens digitais em medicina) especifica um protocolo para transferência, armazenamento e exibição de imagens médicas (PIANYKH, 2012). Um sistema que está intrinsecamente ligado ao DICOM é o PACS (*Picture Archiving and Communication Systems* – Sistema de comunicação e arquivamento de imagens). PACS são sistemas médicos que consistem necessariamente de hardware e software, construídos para gerenciar estudos de imagem médica digital. São basicamente compreendidos em: modalidades, que são aparelhos de obtenção de imagens (e.g. tomografia computadorizada, ressonância magnética), centrais de armazenamento de imagens, onde as imagens obtidas são arquivadas, e estações de trabalho, onde os radiologistas analisam as imagens. São padrões altamente integrados, focados no DICOM e interoperáveis (PIANYKH, 2012). Apesar de vários benefícios do padrão DICOM, é notado que esta estrutura foi projetada para uma arquitetura PACS local, isto é, sua criação não previa a utilização em redes de longa distância, mas sim a utilização em organizações com base em redes locais. Dessa forma, tornou-se uma tecnologia defasada, pois projetos modernos de

compartilhamento de imagens digitais, como a telerradiologia, precisam de soluções que integrem as estruturas DICOM existentes de uma forma flexível e escalável em redes de longa distância (PIANYKH, 2012).

Visando solucionar o problema de distribuição de imagens e dados médicos via Internet, surgiu o DICOMFlow (ARAUJO et al., 2016; LUCENA-NETO et al., 2016), que constitui uma Infraestrutura de Informação (AANESTAD et al., 2017), que é assíncrona, escalável e descentralizada, que efetua a comunicação entre entidades (e.g., hospitais, clínicas, radiologistas) que desejam compartilhar exames de imagens. Este processo ocorre via e-mail, que é utilizado para transportar um protocolo de serviços de gerência de exames de imagens no padrão DICOM. Desta forma, utiliza-se o e-mail para notificar o receptor da disponibilidade de um exame para realizar alguma ação (e.g., arquivar, emitir laudo), incluindo-se URL (*Uniform Resource Locator* – Localizador de Recurso Uniforme) de acesso e credencial de autorização. O receptor resgata o exame (dados brutos de imagens) via protocolo de transmissão de conteúdo definido na URL (e.g., HTTP/REST) e atendendo requisitos de segurança parte a parte. Ou seja, apenas metadados são trafegados pelo e-mail, utilizando-se da URL para resgatar o conteúdo dos exames.

A utilização de URLs para acesso a informações é prática comum na Internet, pois, através delas, um grande número de diferentes elementos pode ser obtido, como vídeos, áudios e arquivos. Porém, sua efetividade depende da constância (persistência) da capacidade de se localizar objetos (dados e serviços), pois a URL vincula recursos na Internet à sua localização atual na rede e ao caminho do objeto. Entretanto, quando um recurso da Internet é movido da sua localização original, a URL se torna inválida (LAWRENCE et al., 2001).

Investigar soluções para esse problema se torna cada vez mais importante com a crescente dependência dos sistemas e infraestruturas de informação (AANESTAD et al., 2017; BOWKER et al., 2010; MONTEIRO et al., 2013) de URLs para localização de objetos, como documentos, imagens e vídeos digitais. É relevante na área de saúde, particularmente para telerradiologia, que envolve a transferência de imagens e informações de pacientes entre locais distantes geograficamente (BARNEVELD BINKHUYSEN; RANSCHAERT, 2011). Uma forma de prover interoperabilidade é permitir a localização dos recursos onde eles

foram gerados, e a utilização de URLs para acesso a estes recursos é prática crescente no domínio da área de saúde. Como por exemplo, no padrão XDS-I (*Cross-Enterprise Document Sharing for Imaging*), o caminho para a localização de imagens e documentos médicos é armazenado no seu catálogo, onde posteriormente o usuário poderá ter acesso através de uma aplicação (VREELAND et al., 2016). Além disso, informações clínicas de pacientes devem ser localizáveis e acessíveis por longos períodos de tempo.

Por utilizar URLs para o acesso aos exames, podem ocorrer situações nas quais tais arquivos são movidos para outra localização. Por exemplo, quando um hospital se associa a uma rede de hospitais e, conseqüentemente, a localização dos exames é mudada (e.g. passam a residir em um *Datacenter* terceirizado), suas URLs ficarão inválidas, desabilitando o acesso dos parceiros às informações médicas.

Soluções comumente encontradas para o problema da persistência de URLs se baseiam na existência de elementos centrais, como nos casos do Digital Object Identifier (DOI) (PASKIN, 2000) e do sistema LOCKSS – Lots of Copies Keep Stuff Safe (LAWRENCE et al., 2001). Tais soluções mantêm um banco de dados de acesso centralizado para resolver os nomes dos objetos indicando sua localização e metadados suplementares, porém, por serem sistemas centralizados, ocorrem limitações para compartilhamento global de recursos através da Internet, que são necessários para comunicações entre sistemas de saúde. Sistemas distribuídos são importantes em aplicações de saúde, pois compartilham recursos tanto de forma local, quanto de forma global. São sistemas abertos, isto é, permitem o ingresso de novas entidades à infraestrutura e trazem consigo vantagens, como tolerância a falhas e escalabilidade (SAINI; YADAV, 2015).

Uma solução distribuída e sem controle central para este problema de persistência de acesso a objetos é a *Named Data Networking* (NDN), uma proposta de arquitetura de Internet que visa substituir o IP (*Internet Protocol*). Com o NDN, os objetos utilizariam um nome único, sendo possível o acesso a esses objetos apenas pelo nome, não importando a localização (SAXENA; ROORKEE, 2016; ZHANG et al., 2014). O NDN utiliza a nomeação hierárquica das informações, e, através desse nome único, seu acesso será sempre possível.

Ao invés de utilizar a tradicional arquitetura TCP/IP para acesso a objetos (dados e serviços) através do endereçamento de hosts (i.e., entregar pacotes de rede para um endereço de destino), possibilita o acesso através de um nome único que o identifique, de modo que o acesso será sempre possível, não importando sua localização física.

A localização dos objetos na rede ocorre através da troca de dois tipos de pacotes: *Interest* (interesse) e *Data* (dados). O consumidor, aquele que requer o objeto (e.g. usuário de rede, aplicação), atribui o nome do objeto desejado em um pacote *Interest* e o envia pela rede. Roteadores detentores de objetos (roteadores NDN) utilizam esse nome para reenviar o *Interest* até o real detentor do objeto. Uma vez que o *Interest* atinge um roteador que retém a informação solicitada, este irá retornar um pacote *Data*, que contém tanto o nome quanto o objeto, juntamente com uma chave assinada pelo detentor do conteúdo, que une o nome e os dados, tornando-os imutáveis (SAXENA; ROORKEE, 2016). O pacote *Data* seguirá em retorno pelo caminho feito pelo pacote *Interest* até o consumidor que fez a requisição.

Com a mudança na localização dos objetos, suas informações para acesso são propagadas na rede de forma distribuída, acarretando em uma tabela para alcance às informações atualizadas.

Ao verificar a natureza distribuída do NDN e do DICOMFlow, é possível ver características em comum, e com isso, projetar uma solução distribuída que permita o acesso a imagens médicas através de URLs, mesmo que a localização física dos dados seja alterada.

1.1. Objetivos

1.1.1. Objetivo geral

O objetivo geral deste trabalho é propor uma solução distribuída que permita o acesso a dados médicos via URLs persistentes trafegadas via DICOMFlow na Internet.

1.1.2. Objetivos específicos

- Estudar soluções atuais que sanem o problema de acesso a dados através de URLs, a fim de observar a razão de estas soluções existentes não serem adequadas para o DICOMFlow;
- Analisar situações nas quais o Named *Data* Networking foi empregado, visando explorar seus benefícios;
- Elaborar um modelo arquitetural da solução para acesso a dados médicos utilizando URLs persistentes;
- Desenvolver um protótipo do modelo arquitetural;
- Realizar experimentos visando a viabilidade técnica e operacional do protótipo.

1.2. Justificativa

Como a telerradiologia é amplamente utilizada, a visualização das imagens é realizada essencialmente por vias digitais e os exames precisam estar disponíveis por um longo período de tempo, faz-se necessária uma ferramenta que permita que o usuário possa facilmente acessar os dados de forma transparente e descentralizada, sem se importar em qual nó da rede estes dados estão alocados.

Como se tratam de informações clínicas, estas podem ser geradas em uma entidade (e.g. hospitais, clínicas), porém podem ser analisadas em outra entidade (e.g. radiologista). Também podem ser analisadas por diferentes profissionais com propósitos distintos (e.g. emissão de laudo, análise de histórico). Como são informações acessadas por diferentes organizações, preza-se pela persistência de acesso a estas informações, isto é, manter as entidades capazes de poder adquirir os documentos do paciente.

Também, caso uma das instituições que compõem o domínio de rede altere seu local de armazenamento (e.g. armazenamento de terceiros, armazenamento em *cloud computing*), ou a instituição encerre suas atividades, as URLs enviadas (e.g., pelo DICOMFlow) ficarão inválidas, de forma que as outras entidades ficarão impossibilitadas de acessar suas informações.

Por se tratar de um problema observado durante o desenvolvimento do DICOMFlow, este trabalho propõe uma solução que utiliza o *Named Data Networking* como tecnologia descentralizada para a localização de dados compartilhados entre entidades de saúde pertencentes a domínios de rede distintos para formação de uma infraestrutura de informação telerradiológica. Este modelo utiliza conceitos de NDN como solução, tais como o seu protocolo de roteamento.

Com isso, as informações poderão ser acessadas mesmo que o profissional não saiba onde elas estão armazenadas, pois as entidades se comunicarão de forma distribuída, buscando ponto-a-ponto a informação desejada.

1.3. Metodologia

A fim de realizar os objetivos propostos na seção 1.1, o desenvolvimento do trabalho foi realizado em etapas. Primeiramente, houve a revisão da literatura mais recente sobre o *Named Data Networking* para analisar trabalhos que empregaram tecnologia em situações distintas, em particular em sistemas de informação em saúde, a fim de verificar os benefícios da adoção do NDN como ferramenta. Em segundo lugar, foi elaborado um modelo arquitetural, com o objetivo de estabelecer um panorama sobre o problema e encontrar uma solução mais prática e adequada para o desenvolvimento do trabalho, buscando as ferramentas e tecnologias apropriadas. Em seguida, foi elaborado um protótipo inicial, com o intuito de verificar a sua funcionalidade e sua a viabilidade técnica e operacional.

Com a revisão da literatura, foi possível verificar situações onde o NDN foi aplicado de forma bem sucedida, também analisar as limitações encontradas na adoção da tecnologia, que foram base para a elaboração do modelo arquitetural. Neste modelo, foi traçada uma visão geral para o problema e as formas possíveis de resolvê-lo, como também as tecnologias que foram utilizadas na produção do trabalho. Este modelo é essencial, pois, através dele foi possível analisar a situação de forma mais sintética e pragmática. Em posse disto, foi iniciado o protótipo inicial, que, através dele, questões práticas foram testadas.

Em resumo, este trabalho foi realizado de acordo com seguintes atividades:

1. Levantamento bibliográfico: busca de artigos referentes às áreas *Named Data Networking*;
2. Análise do DICOMFlow e suas necessidades para a utilização de URLs persistentes;
3. Criação do modelo arquitetural, através dele, foi possível também a seleção de tecnologias empregadas na construção da ferramenta;
4. Elaboração do protótipo, a fim de buscar resultados preliminares;
5. Experimentação do protótipo: Preparação de experimentos de viabilidade técnica em ambiente simulado, para geração e análise de resultados.

1.4. Organização do trabalho

O restante do trabalho está organizado como segue:

Capítulo 2: Neste capítulo serão apresentados conceitos de infraestrutura de informação, a solução de distribuição de imagens DICOMFlow, alguns conceitos e seu funcionamento básico, como também uma explicação do *Named Data Networking*, seus componentes, sua utilização e seu funcionamento. Também será apresentado o NDN Forward Daemon, um componente essencial para a criação do modelo proposto.

Capítulo 3: Através dos trabalhos relacionados, será possível verificar que já existem casos práticos da utilização da tecnologia de nomeação de objetos, e que estes casos possuem resultados positivos.

Capítulo 4: Descreve o modelo proposto neste trabalho. Valendo-se da utilização do NDN Forward Daemon e do DICOMFlow, propõe-se a introdução da camada DICOMAddress para nomeação e pesquisa de objetos em uma rede.

Capítulo 5: Aqui será explicado de forma mais detalhada o seu funcionamento, criação do projeto, classes e aspectos técnicos da camada DICOMAddress

Capítulo 6: Este capítulo apresenta as considerações finais do trabalho.

2. Referencial Teórico

Este capítulo tem como principal propósito apresentar conceitos relacionados a infraestrutura de informação (II), explicar a solução de distribuição de imagens médicas DICOMFlow, seus componentes, camadas e seu fluxo de funcionamento. Também introduzir o conceito de nomeação de objetos através do *Named Data Networking*, seus componentes, funcionamento e apresentar o NDN Forwarder Daemon, a implementação do encaminhador de rede do NDN, que foi utilizada como base para a criação da camada DICOMAddress.

2.1. Infraestrutura de Informação

O termo "infraestrutura" se refere a grandes conjuntos de equipamentos coletivos necessários às atividades humanas, como edifícios, estradas, pontes, trilhos, canais, portos e redes de comunicação. Além de tijolos, argamassas, tubulações ou fios, a infraestrutura abrange entidades mais abstratas, como protocolos (humanos e informáticos), padrões e memória (BOWKER et al., 2010). Adicionada ao termo "informação", a infraestrutura refere-se genericamente a recursos e serviços digitais geralmente associados à Internet, a fim de suportar alguma atividade humana, como: serviços computacionais, mesas de ajuda e repositórios de dados (BOWKER et al., 2010).

Como é possível observar, as infraestruturas de informação formam uma unidade de projeto diferente quando comparadas às tradicionais classes de soluções de tecnologia da informação (TI). Essas classes de projeto podem ser definidas em sua ordem de crescente complexidade como: (1) recursos de TI, (2) aplicações, (3) plataformas e (4) IIs (HANSETH; LYYTINEN, 2010). As principais diferenças entre essas classes estão em sua complexidade geral, como elas se relacionam com seus ambientes de projeto e utilização, e como elas se comportam ao longo do tempo em relação a esses ambientes (HANSETH; LYYTINEN, 2010).

Recurso de TI é a possibilidade e/ou o direito do usuário ou de uma comunidade de usuários de realizar um conjunto de ações em um objeto ou

processo computacional. Um exemplo de recurso de TI seria um editor de texto. As aplicações consistem em conjuntos de recursos de TI. São desenvolvidas para atender a um conjunto de necessidades específicas de usuários dentro de um conjunto seleto de comunidades. Um exemplo de aplicação é a suíte Microsoft Office, em que o recurso de TI (editor de texto) está inserido. Plataformas diferem de aplicações devido a sua base de usuários ser heterogênea e crescente, ou seja, seu contexto de projeto não é determinado devido à constante generalização de seus recursos de TI. Um exemplo de plataforma é um sistema operacional (no qual os recursos de TI e as aplicações estão inclusos). Infraestruturas de informação são recursivamente compostas por recursos de TI, aplicações e plataformas, podendo ainda ser composta por outras infraestruturas (HANSETH; LYYTINEN, 2010).

Os sistemas de TI em saúde implementados, geralmente se destinam a conectar vários locais, seja dentro de uma organização ou além dela. Uma infraestrutura de informação não local e distribuída abrangerá múltiplos atores que podem ter diferentes necessidades e interesses que podem não estar alinhados. Para que uma infraestrutura de informação funcione, é necessário encontrar uma resolução de trabalho entre os múltiplos interesses locais e os interesses globais da rede como um todo (AANESTAD et al., 2017).

É percebido que as IIs surgem a partir de “bases instaladas”, isto é, ambientes construídos e pré-existentes, que muitas vezes são ignorados por outras conceituações de sistemas grandes e complexos. Os estudos das IIs enfatizam a durabilidade e o papel central das práticas, convenções, ferramentas e sistemas existentes, e esta base instalada tem um impacto fundamental na evolução das infraestruturas de informação (AANESTAD et al., 2017). Esta perspectiva enfatiza que a infraestrutura não cresce do novo: luta com a “inércia da base instalada” e herda forças e limitações dessa base (AANESTAD et al., 2017). Um exemplo de II bem sucedida é a Internet, pois o seu controle é difundido entre grupos diversos e a sua estrutura permite a inclusão e exclusão de componentes sem perturbar diretamente os outros usuários.

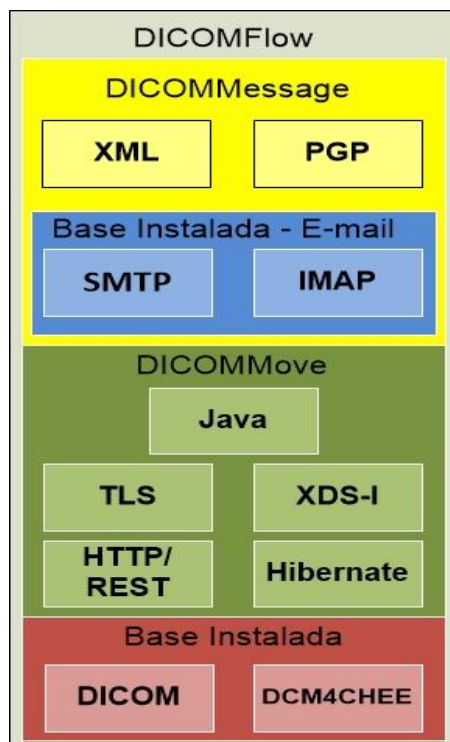
2.2. DICOMFlow

O DICOMFlow (ARAUJO et al., 2016; LUCENA NETO et al., 2016) é uma solução de distribuição de imagens médicas entre entidades de saúde (e.g., hospitais, clínicas, radiologistas) pertencentes a domínios de rede distintos visando a formação de uma infraestrutura de informação telerradiológica. Sua implementação de referência seguiu as definições presentes em um protocolo de serviços de gerência de exames de imagens, elaborado conforme os princípios da teoria de projetos para complexidade dinâmica em infraestruturas e informação (HANSETH; LYYTINEN, 2010), que adota a abordagem de reutilizar uma infraestrutura existente a fim de fomentar seu crescimento mediante um processo de alavancagem e adaptação. O resultado da implementação do protocolo foi um modelo que aproveita a infraestrutura PACS/DICOM (HUANG, 2011; PIANYKH, 2012) existente nas organizações de saúde e que possui uma arquitetura descentralizada, que prescinde de elementos centrais de controle, facilitando a adoção da solução e favorecendo o seu crescimento. A arquitetura do modelo é organizada em duas camadas (DICOMMessage e DICOMMove) que controlam a comunicação entre as entidades de saúde. A camada DICOMMessage, utilizando os protocolos de troca de e-mail SMTP (*Service Mail Transfer Protocol*) e IMAP (*Internet Message Access Protocol*), notifica as entidades da disponibilidade de um exame para realização alguma ação (e.g., arquivar, emitir laudo), com apenas metadados trafegados nas mensagens, dentre esses, a URL de acesso ao exame e credencial de autorização. As mensagens trafegadas são criptografadas e assinadas utilizando, respectivamente, os certificados digitais do receptor e do emissor. A camada DICOMMove, empregando tecnologia HTTP/REST e utilizando a URL de acesso, localiza fisicamente e transfere o exame para as entidades de destino. Ambas as camadas possuem mecanismos de segurança de informação especificados pelo protocolo. A Figura 1 ilustra o modelo arquitetural do DICOMFlow projetado em duas camadas.

A comunicação em duas vias distintas dá a opção para a entidade receptora de mensagem resgatar o exame imediatamente ou em um momento posterior, caracterizando um fluxo de comunicação assíncrono. A Figura 2 ilustra

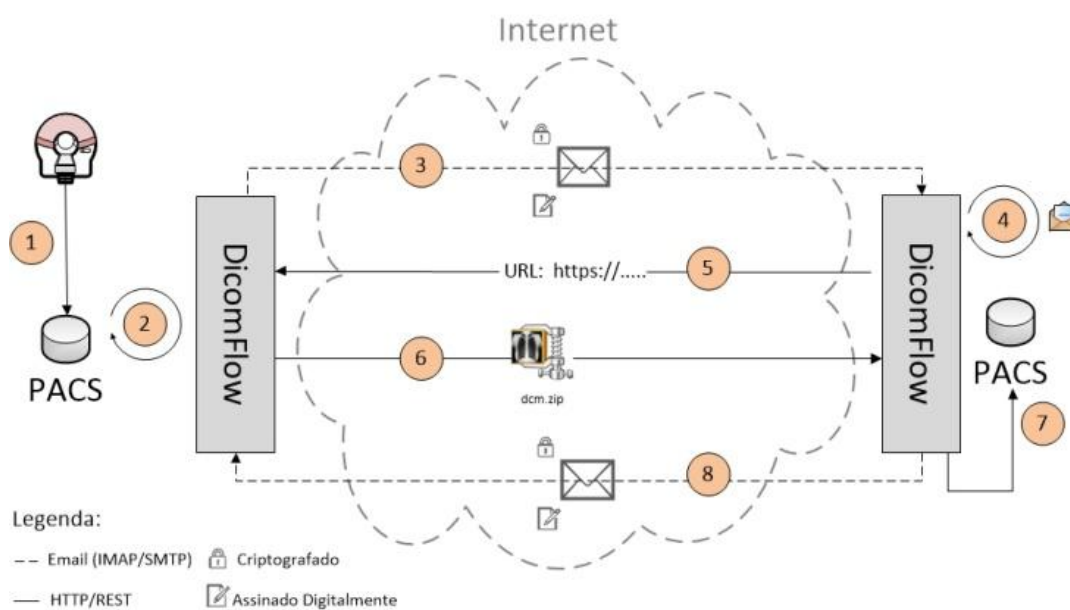
o ciclo de comunicação completo do DICOMFlow em uma solicitação de arquivamento em um provedor de armazenamento.

Figura 1 – Modelo Arquitetural do DICOMFlow



Fonte: (LUCENA NETO et al., 2016)

Figura 2 – Fluxo de comunicação em um cenário de uso



Fonte: Elaborada pelo autor

Segue a descrição das etapas do fluxo de atividades da Figura 2:

1. Em um departamento de radiologia de alguma organização de saúde, um paciente realiza um exame de imagem (e.g., uma tomografia computadorizada), cujos dados são enviados via protocolo DICOM para o servidor PACS local para armazenamento;

2. O serviço do DICOMFlow monitora regularmente o servidor PACS para detectar a chegada de novos exames a fim de determinar a ação a ser realizada com base em uma política de distribuição de exames/requisição de serviços. Tal política é definida em uma linguagem capaz de especificar políticas como: a) encaminhar exames de ressonância magnética do abdome para o radiologista Dr. X (e-mail é especificado) para emitir laudo; ou b) encaminhar todos os exames que chegam ao PACS para o provedor de armazenamento Y (e-mail é especificado) para fins de arquivamento permanente;

3. Seguindo a política, o serviço do DICOMFlow elabora e envia o e-mail correspondente para o destinatário com uma mensagem em XML (BRAY et al., 1997) contendo o serviço requisitado, informações básicas do paciente, dados de recuperação do exame e credenciais de acesso. A mensagem é assinada digitalmente pelo originador do exame e criptografada com a chave pública do destinatário com o padrão S-MIME utilizando certificados digitais;

4. No destinatário, o serviço do DICOMFlow monitora a respectiva caixa de e-mail em busca de mensagens com requisição de serviços. Ele lê a mensagem, descriptografa e valida a assinatura do remetente;

5. O destinatário envia uma solicitação de *download*, utilizando as credenciais e URL de acesso recebidas na mensagem;

6. O originador envia o arquivo via HTTPS/REST após a autenticação (baseada em certificado digital) e autorização de acesso (baseada nas credenciais).

7. Após o *download*, no caso do provedor de armazenamento, o exame é arquivado no PACS local do destinatário;

8. Um e-mail de confirmação é enviado para o remetente concluindo a transação de execução do serviço.

2.2.1. Protocolo de Serviços

O DICOMFlow especifica um conjunto de serviços básicos capaz de dotar a infraestrutura de maior capacidade de adaptação porque possibilita a definição de novos serviços a partir da composição de serviços mais simples, já implementados. Estes serviços especificam ações, informando as atividades que devem ser executadas pelo receptor da mensagem. Por exemplo, o serviço de armazenamento (**Storage**) possui uma ação de arquivamento (**Save**) para requisitar que um novo estudo seja arquivado pelo receptor da mensagem. As informações dos serviços e suas ações são estruturadas em linguagem de marcação XML e carregadas no corpo do e-mail. Foram estabelecidos os seguintes serviços (Tabela 1): **Storage**, responsável pela comunicação de requisições de transferência de estudos de imagens para arquivamento permanente em uma ou mais entidades externas àquela que os originou, assim como por comunicar os resultados obtidos do atendimento às requisições. **Sharing**, responsável por sinalizar aos interessados a existência e localização de novos estudos de imagens e por comunicar respostas a essas sinalizações. **Request** permite ao originador de um estudo requisitar a uma entidade externa (geralmente médicos radiologistas ou serviços de processamento) a realização de alguma operação tendo por base o estudo original, por exemplo, produzir e retornar um laudo. **Discovery** possibilita ao originador da mensagem descobrir se um determinado serviço (ou conjunto de serviços) pode ser atendido por uma entidade qualquer.

Tabela 1 – Serviços do DICOMFlow e suas principais ações.

Serviço	Ação	Descrição
Storage	Save	Requisição de armazenamento de estudo.
	Result	Resultado da operação de armazenamento.
Sharing	Put	Sinalização de estudo disponível para compartilhamento.
	Result	Resultado informando registro de metadados de compartilhamento.
Request	Put	Requisição para realizar operação em um estudo (e.g. laudo).
	Result	Resultado da ação (e.g. laudo efetuado e retornado ao requisitante).
Discovery	VerifyServices	Requisição das listas de serviços disponíveis.
	VerifyResult	Resultado contendo a lista de serviços.

Fonte: (LUCENA NETO et al., 2016)

Os serviços encaminhados via e-mail devem conter atributos com as informações necessárias para a transmissão, como URLs para acesso aos exames.

Por utilizar URLs para o acesso aos exames, podem ocorrer situações nas quais tais arquivos são movidos para outra localização, assim as URLs ficarão inválidas, desabilitando o acesso dos parceiros às informações médicas. Por exemplo, no cenário de uso apresentado na Figura 2, devido à assincronia do fluxo de comunicação, em algum momento após o envio de mensagem (3) e antes da solicitação de *download* (5), pode ocorrer de a organização de saúde se associar a uma rede de hospitais e, conseqüentemente, a localização dos exames ser mudada (e.g. passam a residir em um *Datacenter* terceirizado), quando o provedor de armazenamento Y efetuar a solicitação do *download* de um exame, utilizando a URL inválida, ocorrerá uma exceção no fluxo de comunicação que pode gerar sua interrupção ou seu bloqueio.

2.3. Named Data Networking

Uma solução para o problema de persistência de acesso a objetos ilustrado na seção anterior é a utilização do protocolo *Named Data Networking* (NDN), que é baseado nas redes centradas em conteúdo (*Content-Centric Networking*) (JACOBSON et al., 2012; OEHLMANN, 2013). O principal objetivo do NDN é mudar a semântica do serviço de rede. Ao invés de utilizar a tradicional arquitetura TCP/IP para acesso a um objeto (dados e serviços) através do endereçamento de hosts (i.e. entregar pacotes de rede para um endereço de destino), o NDN possibilita o acesso através de um nome único que o identifique, facultando o acesso ao objeto, não importando sua localização física.

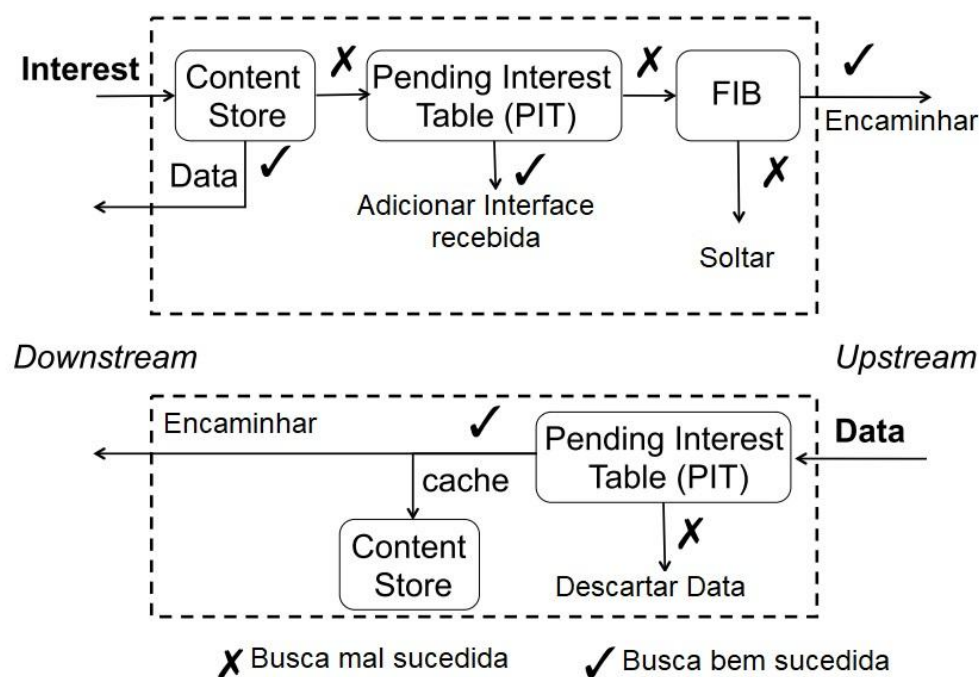
A definição da identificação única de um objeto é tratada pelo protocolo NDN com o uso nomes hierarquicamente estruturados que pode nomear qualquer objeto na rede - um arquivo, um fragmento de dados em um filme ou um livro (*chunk*), um comando para controlar objetos (*Internet of Things*) - Por exemplo, um vídeo de demonstração armazenado em um site denominado SiteA pode ter o nome */SiteA/videos/demo.mpg*, em que *'/'* delinea os componentes de nome em representações de texto, semelhantes às URLs. Essa estrutura hierárquica permite que as aplicações representem o contexto e as relações dos elementos de dados. Por exemplo, o segmento 3 da versão 1 de um vídeo de demonstração pode ser nomeado */SiteA/videos/demo.mpg/1/3* (ZHANG et al., 2014). Isso também permite a agregação de nomes, por exemplo, */SiteA/videos* pode corresponder a uma coleção de vídeos. Vale ressaltar que cada aplicação pode escolher o esquema de nomeação de seus objetos que se adapte a suas necessidades (JACOBSON et al., 2012).

Outro aspecto do NDN está relacionado à localização desses objetos em rede, efetuada através da troca de dois tipos de pacotes: *Interest* (interesse) e *Data* (dados). O consumidor, aquele que requer o objeto (e.g. usuário de rede, aplicação), atribui o nome do objeto desejado em um pacote *Interest* e o envia pela rede. Roteadores detentores de objetos (roteadores NDN) utilizam esse nome para reenviar o *Interest* até o real detentor do objeto. Uma vez que o *Interest* atinge um roteador que retém a informação solicitada, este irá retornar um

pacote *Data*, que contém tanto o nome quanto o objeto, juntamente com uma chave assinada pelo detentor do conteúdo, que une o nome e os dados, tornando-os imutáveis (SAXENA; ROORKEE, 2016). O pacote *Data* seguirá em retorno pelo caminho feito pelo pacote *Interest* até o consumidor que fez a requisição.

Para realizar as funções de encaminhamento de pacotes de interesse e dados, cada roteador NDN mantém três estruturas de dados: um Armazenador de Conteúdo (*Content Store* - CS) que é um cache temporário de pacotes *Data* que o roteador recebeu, uma Tabela de Interesses Pendentes (*Pending Interest Table* - PIT) que registra os nomes dos objetos solicitados nos pacotes *Interest* e uma Base de Informações de Encaminhamento (*Forwarding Information Base* - FIB) que encaminha os pacotes *Interest* para outros roteadores, como mostrados na Figura 3.

Figura 3 – Estruturas de dados do NDN



Fonte: Adaptado de Zhang et al. (2014)

A Figura 3 ilustra o fluxo de envio de um pacote *Interest* e o recebimento de um pacote *Data*. Quando um pacote *Interest* chega, o roteador NDN primeiro

verifica se o CS possui o objeto correspondente; caso positivo, será retornado o pacote *Data* contendo o objeto. Caso contrário, o roteador procura o nome na PIT, existindo uma entrada correspondente, a PIT grava apenas a interface de entrada do *Interest*. Senão, registra o nome transportado na Internet, juntamente com suas interfaces de entrada e saída e o encaminha para outros roteadores detentores de objetos com base nas informações da FIB, de acordo com a Estratégia de Encaminhamento adaptável. Esta pode decidir abandonar um *Interest* em certas situações, por exemplo, se todos os roteadores estiverem congestionados ou se o *Interest* for suspeito de ser parte de um ataque de DoS (*Denial of Service*).

Quando um pacote *Data* chega, o roteador NDN busca o nome correspondente na PIT e encaminha o pacote *Data* para todas as interfaces de entrada associadas ao nome. Em seguida, remove essa entrada PIT, e armazena o objeto no CS (JACOBSON et al., 2012; ZHANG et al., 2014).

Os pacotes *Data* sempre tomam o caminho inverso dos *Interest*, ambos não possuem endereços de *host* ou de interface; os roteadores encaminham pacotes *Interest* para detentores de objetos baseados nos nomes carregados e retornam pacotes *Data* para consumidores com base nas informações de estado PIT. Esta simetria de troca de pacotes *Interest/Data* induz um loop de controle *hop-by-hop* (de salto em salto – que não deve ser confundido com roteamento ou roteamento simétrico) e elimina a necessidade de qualquer noção de nós de origem ou destino na entrega de dados, diferente do modelo de entrega de pacotes IP de ponta a ponta (ZHANG et al., 2014).

2.3.1. Nomes

Embora roteadores reconheçam limites entre componentes em um nome, eles não atribuem nenhum significado aos nomes, ou seja, os nomes NDN são opacos à rede. O nome em um pacote NDN pode nomear vários objetos – um ponto final, um pedaço de dados em um filme ou um livro (*chunk*), um comando para ligar luzes (*internet of things*).

Essa decisão de projeto permite que cada aplicação escolha o esquema de nomeação que se adapta às suas necessidades e, portanto, a nomeação pode evoluir independentemente da rede. O projeto NDN assume nomes hierarquicamente estruturados, por exemplo, um vídeo produzido pela UFPB pode ter o nome `/ufpb/videos/demo.mpg`, em que `/` delinea os componentes de nome em representações de texto, semelhantes às URLs.

Essa estrutura hierárquica permite que as aplicações representem o contexto e as relações dos elementos de dados. Por exemplo, o segmento 3 da versão 1 de um vídeo de demonstração da UFPB pode ser nomeado `/ufpb/videos/demo.mpg/1/3`. Isso também permite a agregação de nomes, por exemplo, `/ufpb` pode corresponder a uma entidade autônoma originadora do vídeo. Nomes planos podem ser acomodados como caso especial, úteis em ambientes locais, porém delimitadores de nome hierárquicos (*namespaces*) são essenciais tanto no escalonamento do sistema de roteamento, quanto ao prover contexto necessário para os dados (ZHANG et al., 2014).

2.3.2. Roteamento e encaminhamento

O NDN roteia e encaminha pacotes baseados em nomes, o que elimina três problemas causados por endereços na arquitetura IP: escassez de espaços de endereços, NAT (*Network Address Translation*) transversal, que é a tradução de endereços IP e gerenciamento de endereços. Não há problema de insuficiência de endereços, já que o *namespace* é ilimitado. Não há problema de NAT transversal, já que o NDN elimina endereços públicos ou privados. Por fim, a atribuição e gestão de endereços não é mais necessária em redes locais (ZHANG et al., 2014).

O NDN pode usar algoritmos de roteamento convencionais, como o estado de enlace (*link-state*) e o vetor de distância (*distance vector*). Em vez de declarar prefixos IP, um roteador NDN declara prefixos de nome que cobrem os dados que o roteador está disposto a servir. O protocolo de roteamento propaga essas declarações através da rede, informando a cada roteador a construção da sua

própria FIB. Protocolos de roteamento convencionais, como OSPF e BGP (WANG et al., 2012), podem ser adaptados para rotear em prefixos de nomes, tratando nomes como uma sequência de componentes opacos e fazendo a correspondência de prefixo mais longo do componente de um nome em um pacote *Interest* contra a tabela FIB (ZHANG et al., 2014).

2.4. Propagação de nomes

Os algoritmos de encaminhamento adequados para Internet, i.e. estado de enlace (*link-state*) e vetor de distância (*distance vector*), podem ser utilizados para NDN com ligeiras modificações. O IP e o NDN usam a FIB para armazenar informações relacionadas ao roteamento. O IP pesquisa o endereço de destino no FIB para localizar o próximo salto e entregar os pacotes ao endereço de destino, não necessariamente pelo melhor caminho. O NDN pesquisa o prefixo do nome no FIB para encontrar o(s) próximo(s) salto(s) e busca dados, não necessariamente a cópia mais próxima. Os protocolos atuais de roteamento da Internet podem ser adaptados para uso no NDN alterando os tipos de mensagem (pacote *Interest*/pacote *Data*) e adicionando o encaminhamento de vários caminhos (SAXENA; ROORKEE, 2016; ZHANG et al., 2014).

2.4.1. Roteamento baseado em OSPF

Para suportar o roteamento baseado em nome no NDN, o protocolo de roteamento baseado em IP, OSPF, foi estendido para distribuir prefixos de nomes e calcular rotas para eles. Tal protocolo de roteamento dinâmico baseado em nome para NDN foi denominado OSPFN. O protocolo de roteamento OSPFN usa três módulos principais que são executados em paralelo em um nó: CCND (*Content Centric Networking Daemon*), OSPF Daemon (OSPFD) e OSPFN. O CCND manipula o encaminhamento de pacotes NDN. OSPFD inunda os anúncios de estado de link opaco (OLSA) para toda a rede. OSPFN constrói o nome OLSA e os fornece ao seu OSPFD local

O roteamento OSPFN funciona em sete etapas principais, conforme explicado: Etapa 1, quando um nó é inicializado, ele prepara o nome OLSA para cada prefixo de nome que deseja anunciar na rede. Etapa 2, o OSPFN fornece o OLSA ao seu OSPFD local para inundação em toda a rede. Etapa 3, quando o OSPFD de um nó recebe um OLSA, ele fornece OLSA atualizado ao seu OSPFN local. Etapa 4, Se o OLSA for novo e atualizado, o OSPFN extrairá o prefixo de nome junto com o ID do roteador do OLSA e mantém em uma tabela de prefixo de nome. Etapa 5, o OSPFN envia uma consulta ao OSPFD para localizar o (s) próximo (s) salto (s) para alcançar o (s) roteador (es) de origem de cada prefixo de nome. Etapa 6, depois de receber uma mensagem de consulta do OSPFN, o OSPFD verifica sua tabela de roteamento para a lista do próximo salto e o custo do caminho e envia as rotas de volta ao OSPFN. Etapa 7, o OSPFN prepara as entradas do FIB (prefixo do nome, próximo(s) salto(s) e custo do caminho) e atualiza o CCND (SAXENA; ROORKEE, 2016; ZHANG et al., 2014).

2.4.2. Roteamento de camada de enlace em NDN

O NDN utiliza o protocolo de roteamento de estado de enlace de dados nomeados (NLSR - *Named-data Link State Routing protocol*), que é um protocolo de roteamento *link-state* executado na parte superior do NDN, onde cada pacote *Data* usado para atualizações de roteamento é assinado pelo roteador de origem para suporte à autenticação. Existem quatro fatores-chave do protocolo de roteamento NLSR. A primeira é nomear os roteadores, links, processos, dados e chaves. Em segundo lugar, distribuição de chaves e confiança para eles. Em terceiro lugar, para puxar a disseminação de atualizações de roteamento em vez de empurrar as atualizações como no OSPF. Em quarto lugar, classificar as interfaces para suportar o encaminhamento de múltiplos caminhos.

O NLSR propaga o LSA para construir a topologia de rede e distribuir os prefixos de nomes. Cada nó mantém o banco de dados do estado do enlace (LSDB) para armazenar o LSA mais recente. Existem dois tipos de LSA: LSA de adjacência, que anuncia todos os links ativos que conectam um roteador NDN a seus vizinhos e prefixam LSA, que anuncia prefixos de nomes registrados.

Através desta adjacência LSA, cada nó NLSR encontra uma topologia de rede. Sempre que o NLSR encontrar alguma falha ou recuperação nos links conectados ou nos processos vizinhos, ele enviará LSA recente ou nova a todos os roteadores da rede. Ele também propaga novo LSA quando qualquer prefixo de nome é adicionado ou excluído. O NLSR usa os pacotes NDN *Interest Packets* e *Data Packets*, portanto, é necessário nomear os roteadores e as atualizações de roteamento em que um processo NLSR pode ser nomeado como / <network> / <site> / <router> / NLSR. Os roteadores também trocam os hashes do LSDB para detectar contradições e executar recuperações deles, o que é conhecido como sincronização *hop-by-hop*. Essa abordagem reduz o *overhead* em uma rede estável porque é necessário trocar apenas um hash em vez de muitos LSAs. O protocolo de roteamento NLSR usa o protocolo de sincronização CCNx ou o Sync para disseminar o LSA atualizado (ou recente) entre os roteadores vizinhos, onde ele permanece conectado ao repositório CCNx (ou Repo). Repo é uma aplicação que suporta comunicação com o CCND e mantém coleções de dados nomeados, nomeadas como slices. Essas fatias (fatia contém LSAs) são mantidas no Sync com outras fatias idênticas em nós vizinhos (SAXENA; ROORKEE, 2016; ZHANG et al., 2014).

2.5. NDN Forwarding Daemon

O *NDN Forwarding Daemon* (NFD) é a implementação de um encaminhador de rede que acompanha a evolução do protocolo NDN. Sua principal funcionalidade é encaminhar pacotes *Interest* e *Data* entre interfaces de rede. Para isso, abstrai os mecanismos de transporte de rede de baixo nível e mantém estruturas básicas como o *Content Store* (CS), o *Pending Interest Table* (PIT) e o *Forwarding Information Base* (FIB) (AFANASYEV et al., 2015).

O NFD foi projetado para dar suporte aos mais diversos experimentos utilizando a arquitetura NDN. Construído de forma modular e extensível, possibilita utilização em conjunto com diferentes protocolos, algoritmos e aplicações. O fluxo de encaminhamento de um pacote em um NFD se dá da seguinte maneira:

Os pacotes chegam ao NFD via Faces (uma generalização de “interface”), que lêem fluxos de entrada ou *Datagramas* através da API do sistema operacional, extraem os pacotes de camada de rede (*Interest*, *Data* ou *Nacks*¹) e os entregam para encaminhamento. Em seguida, o pacote é processado por uma estrutura denominada *pipeline* de encaminhamento, que define quais operações serão realizadas sobre ele.

No *pipeline* de encaminhamento (quando há o recebimento de um *Interest*), o pacote é devidamente registrado na *Pending Interest Table* (PIT), uma espécie de banco de dados de pacotes *Interest* pendentes. Logo após, uma consulta é feita ao cache local de pacotes (*Content Store* - CS) para verificar se há correspondência entre o pacote solicitado e os que estão armazenados temporariamente. Se houver, o pacote é retornado, caso contrário, o *Interest* deverá ser encaminhado. A partir daí uma estratégia de encaminhamento é aplicada sobre o pacote, decidindo de qual forma ele será passado adiante.

Por fim, o pacote é entregue a uma Face, que fará o empacotamento (dependendo do protocolo utilizado) e o envia novamente, utilizando as funcionalidades do sistema operacional.

Alguns módulos e características da implementação NFD são detalhados a seguir (AFANASYEV et al., 2015):

2.5.1. Faces

Um NFD, como um encaminhador de rede, move pacotes entre interfaces de rede. Pode se comunicar não apenas com interfaces de redes físicas, mas também em uma variedade de outros canais de comunicação (como túneis sobre TCP e UDP). Para isso, o NFD utiliza um mecanismo denominado Faces: uma generalização de uma interface de rede, onde pacotes podem ser enviados e recebidos. Uma Face pode ser, por exemplo, uma interface de rede física, um canal de comunicação entre um NFD e um nó remoto ou até mesmo um canal de comunicação entre um NFD e uma aplicação local.

¹ Nack – Negative Acknowledgement: sinal usado nas comunicações digitais para garantir que os dados sejam recebidos com o mínimo de erros

O processo de encaminhamento de pacotes em um NFD envia e recebe pacotes (*Interest*, *Data* e *Nacks*) através de Faces, que manipulam os mecanismos de comunicação (por exemplo, sockets) e ocultam as diferenças entre os protocolos de rede utilizados, do mecanismo de encaminhamento.

2.5.2. Tabelas

O módulo de tabelas fornece as principais estruturas de dados para um NFD. A *Forward Information Base* (FIB) é usada para encaminhar pacotes *Interest* para fontes potenciais de dados correspondentes. É quase idêntico a um IP, exceto que permite uma lista de Faces de saída em vez de uma única. O *Content Store* (CS) é um *cache* de pacotes *Data*. Os pacotes *Data* que chegam a um NFD são colocados neste *cache* o maior tempo possível, a fim de satisfazer *Interests* futuros que solicitam os mesmos dados. A *Pending Interest Table* (PIT) mantém o controle de *Interests* encaminhados camada acima em direção a fontes de conteúdo, de modo que os pacotes *Data* podem ser enviados para seus solicitantes. Também contém *Interests* recentemente satisfeitos para detecção de loop e medições.

2.5.3. Estratégias de Encaminhamento

Em um NFD, as estratégias de encaminhamento fornecem a inteligência para tomar decisões sobre se, quando e para onde encaminhar pacotes *Interest*. Diferentes aplicações necessitam de diferentes comportamentos de encaminhamento. Por exemplo, um aplicativo de recuperação de arquivos deseja obter o conteúdo de uma fonte com a maior largura de banda, um aplicativo de bate-papo de áudio necessita de uma conexão com o menor atraso, e uma biblioteca de sincronização de dados quer difundir pacotes *Interest* a todas as Faces disponíveis para alcançar seus objetivos. As diferentes necessidades de encaminhamento fazem com que existam diferentes estratégias de encaminhamento no NFD. Existem, também, APIs disponíveis para o desenvolvimento de novas estratégias de direcionamento, de acordo com a necessidade da aplicação.

2.5.4. Armazenamento e Rede

Como cada pacote NDN *Data* contém um nome e uma assinatura, ele é significativo independentemente de quem solicitou ou de onde é recuperado. Assim, um roteador pode armazenar em cache pacotes de dados recebidos em seu Armazenador de Conteúdo e usá-los para satisfazer solicitações futuras. O Content Store é análogo à memória buffer em roteadores IP, mas os roteadores IP não podem reutilizar um pacote após encaminhá-lo para seu destino, enquanto os roteadores NDN podem. NDN trata armazenamento e canais de rede de forma idêntica em termos de recuperação de dados. Para arquivos estáticos, NDN consegue a entrega de dados quase ideal. Mesmo o conteúdo dinâmico pode se beneficiar do armazenamento em cache no caso de multicast (por exemplo, teleconferência em tempo real) ou retransmissão após uma perda de pacote.

2.6. Considerações finais

Baseado no que foi apresentado neste capítulo, foi possível conhecer sobre a necessidade do compartilhamento de imagens médicas, como também verificar a importância de como estas imagens precisam estar disponíveis para sua aquisição. Também foi possível constatar que há a necessidade de um tratamento para a manutenção destes links ativos para acesso às imagens. Ao falar sobre a nomeação de objetos numa rede e a utilização do NDN, é uma solução que mantém o acesso aos objetos mesmo que estes estejam em um lugar não conhecido do usuário que requisita este objeto. Esta tecnologia se mostra importante, pois sua natureza distribuída corresponde com a natureza distribuída do DICOMFlow, sendo uma solução adequada para lidar com os pontos levantados no capítulo.

3. Trabalhos relacionados

Para que os objetivos sejam alcançados, e que seja possível contornar as limitações presentes nos modelos supracitados, é importante observar projetos de pesquisa em que a solução NDN foi adotada, e, através destes analisar suas características a fim de identificar fatores de sucesso presentes em suas soluções e suas limitações, que possam impossibilitar a sua utilização em larga escala. Também é importante analisar projetos que utilizam a base instalada PACS/DICOM que pretendem realizar o compartilhamento de recursos, como também a utilização de *blockchain* com o mesmo objetivo.

O projeto VirtualPACS – *A Federating Gateway to Access Remote Image Data Resources over the Grid* (SHARMA et al. 2009) apresenta um kit de ferramentas chamado VirtualPACS a fim de federar vários recursos de dados de imagem, de modo que os clientes possam interagir com eles como se estivessem armazenados em um PACS centralizado. Com a utilização deste kit de ferramentas, os usuários são capazes de executar operações DICOM padrão (consulta, recuperação e envio) em bancos de dados de imagens distribuídas.

A federação de fontes de dados de imagem remota requer: (1) uma representação de dados comum entre as origens de dados; (2) ferramentas que podem envolver fontes de dados heterogêneas para que os dados de imagem armazenados em uma fonte de dados sejam expostos usando a representação de dados comum e acessados por meio de interfaces bem definidas; (3) um mecanismo para descobrir fontes de dados no ambiente; (4) um mecanismo para interagir com fontes de dados seguras e não seguras e mover resultados de uma solicitação eficientemente de múltiplas fontes de dados para solicitante / consumidor de dados; e (5) um mecanismo para traduzir um pedido DICOM para um pedido que pode ser passado para as fontes de dados para execução, e os resultados do pedido de volta para os objetos DICOM. Para atender a esses requisitos, a plataforma VirtualPACS foi construída em uma arquitetura em camadas, consistindo de três camadas: Mediação, *middleware* e apresentação.

A plataforma VirtualPACS atualmente não existe mais da forma em que foi concebida, pois foi comprada por uma organização, se tornando uma plataforma privada.

O trabalho *Federated Service-Based Authentication Provisioning for Distributed Diagnostic Imaging Systems* (SHARGHI et al. 2015) pretende tratar dois problemas: (1) a falta de uma interface adequada com infraestruturas comuns, que acabou tornando os sistemas PACS proprietários como ambientes fechados e impedindo a integração e interoperação contínuas com infraestruturas comuns. (2) Em relação a segurança, não há mecanismo de autenticação federada para sistemas PACS e cada sistema PACS controla a identidade do usuário localmente. Desta forma, foi proposto um *middleware* baseado em serviços para sistemas de diagnóstico por imagem distribuídos, empregando diferentes tipos de agentes para obter um mecanismo de autenticação federada para tais sistemas.

O *middleware* proposto apresenta duas camadas de serviços: 1) Serviços de comunicação com recursos de transmissão que são responsáveis pela troca de mensagens e 2) Serviços comuns que fornecem as funcionalidades necessárias que são freqüentemente usadas pelos usuários do sistema. Esses serviços comuns são responsáveis por: autenticação de usuários do PACS com base em suas identidades registradas; tomar decisões de controle de acesso; monitorar o uso de usuários e recursos; atualização da política do sistema e gerenciamento de metadados. O *middleware* fornece um serviço de gerenciamento de identidades federadas, que permite aos usuários autenticarem-se com infra-estrutura que inclui um conjunto comum de políticas para todos os aplicativos participantes, a fim de compartilhar com segurança e sem problemas os recursos.

O trabalho *Decentralizing privacy: Using blockchain to protect personal data* (ZYSKIND et al. 2015) propõe a utilização de *blockchain* para a construção de uma plataforma de gerenciamento de dados pessoais com foco em privacidade, evitando o contato dos dados com terceiros. O usuário instala um aplicativo que usa a plataforma para preservar sua privacidade. Ao se inscrever pela primeira vez, uma nova identidade compartilhada (usuário, serviço) é gerada e enviada,

juntamente com as permissões associadas, para o *blockchain* em uma transação. Após isso, tanto o serviço quanto o usuário podem agora consultar os dados usando uma transação com o ponteiro (chave) associado a ela. O *blockchain* então verifica se a assinatura digital pertence ao usuário ou ao serviço. Para o serviço, suas permissões para acessar os dados também são verificadas. Por fim, o usuário pode alterar as permissões concedidas a um serviço a qualquer momento emitindo uma transação com um novo conjunto de permissões, incluindo a revogação do acesso a dados armazenados anteriormente.

Os *blockchains* são tão fortes quanto o tamanho da rede. Quanto maior a rede, mais segura e anti-frágil ela se torna. Isso significa que é necessário uma vasta rede com muitos usuários para que ela se torne ideal. Se um *blockchain* não for uma rede robusta com uma grade de nós amplamente distribuída, será mais difícil obter seu benefício total.

O trabalho *Securing Building Management Systems Using Named Data Networking* (SHANG et al., 2014) defende a utilização de NDN como protocolo para *Building Management Systems*. BMSs são sistemas de software/hardware que realizam o controle, monitoramento e gerenciamento de aquecimento, ventilação e ar condicionado, iluminação, água, acesso físico e outros componentes de um prédio. O trabalho foca na distribuição segura de dados de sensores, que é um elemento fundamental desses sistemas e controle de acesso correspondente para usuários finais, aplicativos e dispositivos. A priori, os BMSs utilizavam protocolos próprios entre dispositivos, porém, nos últimos anos foram adaptados para serem executados através de Ethernet e IP. Ou seja, em vez de executar diretamente acima da camada física, os quadros de dados são encapsulados em pacotes TCP ou UDP e distribuídos por uma rede IP. Nos casos tradicional e IP/Ethernet, o controle de acesso é quase sempre conseguido através de isolamento físico ou lógico (por exemplo, usando VLANs e firewalls). No entanto, o uso de TCP/IP para BMS também enfrenta problemas significativos, como configuração de rede complexa para atender às metas de comunicação e segurança; a exigência de implantar middleware para conectar namespaces de aplicativos aos endereços IP, números de portas e outras designações usadas para encaminhar e encaminhar dados; e a falta de mecanismos apropriados de

segurança a nível de pacotes para a comunicação máquina a máquina. Os autores defendem a adoção de NDN, pois ele fornece funcionalidades, como roteamento e encaminhamento utilizando a base de nomes de dados definida pelo aplicativo, ao invés de endereços de host numéricos e não intuitivos. Suporte de segurança inerente à camada de rede, através de assinaturas por pacotes e criptografia opcional, em oposição a confiar em soluções de camada superior ou isolamento físico do sistema. Arquitetura solicitação/resposta embutida na camada de rede, mais adequada a acesso de dados do sensor e controle autenticado, evitando a utilização de protocolos na camada de aplicativo e cache na rede e suporte para armazenamento persistente distribuído, podendo reduzir cargas de consulta em dispositivos com restrições de recursos, evitando ataques de negação de serviço (DoS).

Já o projeto Navigo (GRASSI et al., 2015) sugere uma aplicação de encaminhamento de pacotes em redes veiculares utilizando NDN, onde os pacotes de Interesse são encaminhados para a localização geográfica indicada nos nomes, sem a necessidade de um protocolo de roteamento. Por ser uma solução automotiva, é dependente da localização, isto é, os dados produzidos e consumidos por eles estão ligados a locais específicos. Exemplos de tais aplicações vão desde a obtenção de atualizações do tráfego rodoviário em uma determinada rua para a busca de um espaço de estacionamento disponível.

Zhang et al. (2016) sugerem uma aplicação de *Mobile Health (mHealth)* utilizando NDN ao invés de TCP/IP. *Open mHealth* aproveita os dispositivos móveis do público (e.g. smartphones e tablets) para estender intervenções baseadas em evidências além do alcance dos cuidados tradicionais e, assim, melhorar a gestão e prevenção de doenças.

Ambas as tecnologias usam dados nomeados para interoperabilidade - *Open mHealth* como um ecossistema de aplicativos e NDN ao nível da arquitetura de rede. Também procuram capacitar os proprietários de dados para exercer controle sobre seus dados sem depender de um determinado canal seguro, tipo de armazenamento ou meio de comunicação.

Foi construído um protótipo de ecossistema *mHealth* chamado NDNFit, projetado e implementado para explorar esses alinhamentos e investigar como NDN pode ser mais eficaz do que TCP/IP para certos tipos de aplicações modernas. Como resultados, foram verificados os potenciais benefícios da arquitetura NDN para a saúde móvel. Especificamente, seu projeto sugeriu o poder da rede por meio de dados nomeados por aplicativo, a utilidade de convenções de nomenclatura para simplificar a arquitetura do aplicativo. O projeto também demonstrou o poder de proteger dados diretamente, o que remove dependências e restrições decorrentes da dependência de camadas de transporte subjacentes para segurança.

Naik et al. (2017) utilizam o protocolo baseado em redes centradas em informações (ICN – *Information Centric Networking*) (TROSSEN; SARELA; SOLLINS, 2010) para o monitoramento dos sinais vitais dos pacientes a partir de redes sem fio de sensores de corpo. Desenvolveram um protocolo de gerenciamento e padronização de dados biomédicos, análise de vizinhança, análise de dados e protocolo de visualização baseado na ICN, onde este trabalho é focado em manter estes dados seguros de ataques à rede.

Primeiramente, foi utilizado um framework de Rede Interna de Área do Corpo Sem Fio (I-WBAN – *Internal Wireless Body Area Network*), que constrói uma estrutura centrada na informação para aplicativos WBAN. Nesta arquitetura, a recuperação de dados é de nível inicial e controlada pelos receptores e não pelos remetentes. Todos os dispositivos de corpo vizinhos podem ser conectados com links sem fio para construir um sistema de informações completo. Quando um nó quer obter um determinado pedaço de dados (*chunk*), ele envia o pacote *Interest*, que contém o nome deste dado. Então, o sistema I-WBAN pesquisa os dados e os retorna. Cada dispositivo do corpo está equipado com um novo mecanismo de encaminhamento CCN (*Content Centric Networking* – Rede Centrada em Conteúdo) que permite a entrega de dados centrados em informação.

Este mecanismo de encaminhamento inclui os três componentes principais utilizados no NDN: *Content Store* (CS), *Pending Interest Table* (PIT) e *Forwarding Information Base* (FIB). O CS é a memória *buffer*, que visa armazenar em *cache*

os pacotes de dados que chegam com maior frequência. A PIT controla as solicitações *Interest* reencaminhadas para as fontes de conteúdo, de modo que os pacotes de dados relevantes devolvidos possam ser enviados camada abaixo para seus solicitantes, respectivamente. A FIB é usada para encaminhar pacotes *Interest* para potenciais fontes de dados correspondentes. Ao alavancar os recursos CCN, os nós I-WBAN adotam um novo modelo de protocolo em camadas, que é apresentado com o encaminhamento baseado em nomes, e não em localização, utilizando a estrutura de nomes hierarquicamente estruturados em NDN. Os nomes contêm componentes amigáveis para humanos e são independentes da localização ao contrário da abordagem IP. A estrutura de nomes do NDN permite que os aplicativos controlem a transmissão de dados entre produtores e consumidores mais flexível. No entanto, os mesmos dados gerados pelo produtor podem ser interessantes para mais de um consumidor ao mesmo tempo (e.g., uma organização e um médico).

Portanto, o NDN permite que nós intermediários possam salvar uma cópia dos dados transmitidos. Então, os consumidores podem recuperar os dados gerados de qualquer nó da rede que tenha uma cópia deles sem a necessidade de transmitir o *Interest* ao produtor. Tais mecanismos são benéficos para os sistemas IoT (internet das coisas) devido à redução da carga de tráfego na rede. Por fim, é visto que ICN altera a arquitetura da internet de centrada em *hosts* para centrada em informação. A informação é altamente distribuída e armazenada em cache em toda a arquitetura ICN. Seu foco principal no conteúdo ao invés de *hosts* fornece uma entrega de conteúdo escalável e eficiente, onde assinante ICN acessa o conteúdo de uma localização diferente.

3.1. Considerações finais

Observando os trabalhos apresentados, é percebido que as soluções que adotam PACS/DICOM, apesar de compartilhadas, não são ideais, pois são privadas e necessitam de uma infraestrutura fortemente acoplada. Já a utilização

de *blockchains* requer redes bastante robustas e com um número grande de usuários para que suas propriedades sejam alcançadas.

Ao analisar os trabalhos apresentados que utilizam NDN, é sugerida nova proposta ao invés rede TCP/IP, oferecendo soluções baseadas em NDN. Estas soluções demonstram sucesso em suas aplicações e não causam uma ruptura no fluxo de funcionamento para os usuários.

Por ser um conceito recente, é visto que a aplicação do NDN está mais focada em objetos (IoT) e equipamentos móveis, o que não exclui a sua utilização em redes fixas também.

4. O modelo DICOMAddress²

Tomando como base o funcionamento dos protocolos, as diretrizes para definição de nomes e roteamento de objetos do protocolo NDN, o modelo aqui proposto sugere a implementação do encaminhador de rede NDN *Forward Daemon* (NFD) em conjunto com a utilização da implementação de referência do DICOMFlow a fim de gerar uma solução que seja capaz de localizar exames médicos distribuídos nas diferentes localidades das comunidades participantes.

Para melhor explicar o funcionamento do modelo, foi criado um cenário que exemplifica a sua necessidade e utilização: uma comunidade que adota o DICOMFlow como protocolo de transmissão de exames imagens médicas (i.e., um conjunto de clínicas e hospitais que são entidades participantes do mesmo grupo, e, através dele, podem compartilhar exames entre si) encontra-se em um mesmo grupo, enviando e recebendo notificações via e-mail para acesso às informações. Como o DICOMFlow tem natureza assíncrona, o usuário decide baixar as informações quando desejar, de acordo com sua necessidade. Para que isto seja possível, a URL enviada via e-mail precisa estar ativa, mesmo que a localização das informações não seja mais a mesma de quando a notificação foi enviada. Caso ocorra o desligamento de uma entidade da comunidade, ou então que haja uma mudança no armazenamento dos arquivos de alguma entidade, as URLs já trafegadas se tornarão indisponíveis para posterior acesso aos arquivos pelos parceiros da comunidade, desta forma rompendo o fluxo do compartilhamento de informações clínicas.

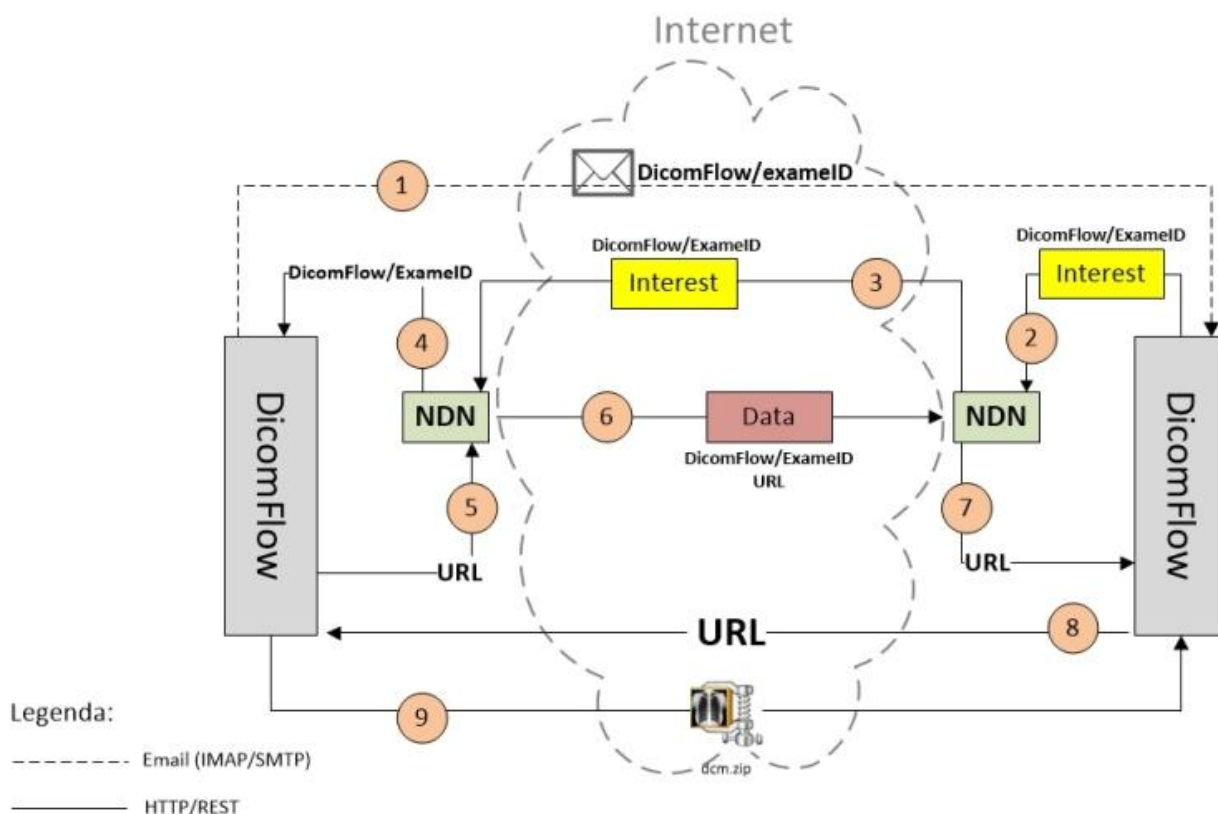
Como solução, o modelo sugere a utilização de URLs ativas. Para que isto ocorra, é necessário um mecanismo de encaminhamento às informações dissipadas na comunidade, i.e., a utilização do NFD como estratégia para localizar os outros nós e retornar a localização atual dos exames de forma transparente ao usuário. O NFD irá buscar nas camadas acima a localização nova do exame e retornará uma URL com a localização atual. Após retornar a URL

² <https://github.com/DicomFlow/DicomAddress>

com a localização atual dos exames, o usuário poderá ter acesso às informações, que serão transmitidas via HTTP/REST, de acordo com o DICOMFlow.

Este modelo não pretende mudar a forma como os *downloads* dos exames são feitos pelo DICOMFlow, e sim utilizar o NDN para encontrar a URL ativa do local físico do exame através do nome único do exame. Então, a URL é o objeto que deve ser encontrado pelo modelo proposto e, após ser encontrada, utilizada pelo DICOMFlow no seu fluxo normal de *download* do exame. Dessa forma, apenas um novo fluxo é adicionado ao processo de distribuição de imagens e, assim, as características da infraestrutura do DICOMFlow são mantidas. A Figura 4 ilustra o ciclo completo de comunicação do DICOMFlow com o novo fluxo de endereçamento incluso (etapas 2, 3, 4, 5, 6 e 7).

Figura 4 – Novo fluxo de comunicação com o modelo incluso



Fonte: Elaborada pelo autor

A caixa intitulada NDN na Figura 4 representa o modelo proposto. Então, o novo fluxo segue da seguinte maneira:

1. Logo após a aquisição de um novo exame, o DICOMFlow originador elabora e envia o e-mail correspondente para o destinatário com uma mensagem em XML contendo o serviço requisitado. A URI (o seu nome único) enviada ao destinatário é formatada nas normas utilizadas de acordo com a geração de nomes escolhida no NDN (e.g. UFPB/1.1.8.2343435.11212.224.33434);

2. Após a leitura da mensagem, o DICOMFlow do destinatário prepara um pacote *Interest* e o encaminha ao NDN do destinatário;

3. O NDN do destinatário verifica o pacote e o redireciona através da estratégia de encaminhamento adotada;

4. O pacote chega ao NDN originador do exame, encaminhando o nome para o DICOMFlow originador;

5. Neste passo, o DICOMFlow originador retorna a localização ativa do exame;

6. O NDN originador então cria um pacote *Data* com a URL para acesso e o envia pelo caminho inverso do *Interest*;

7. Ao receber o pacote *Data*, o NDN do destinatário recupera a URL e repassa para o DICOMFlow do destinatário;

8. O DICOMFlow do destinatário envia uma solicitação de *download*;

9. Por fim, o originador envia o arquivo via HTTPS/REST.

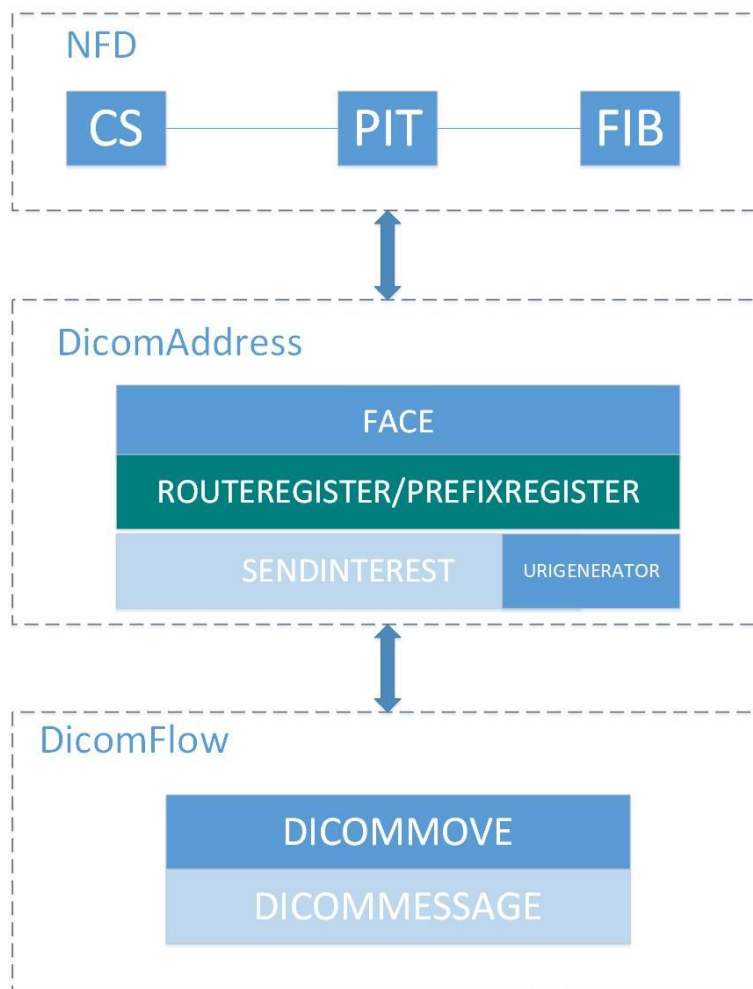
É interessante verificar que a URI enviada da entidade geradora para a entidade receptora é o seu nome único no formato NDN, onde, sua formatação seguirá as regras do modelo proposto (que serão apresentadas abaixo). Através desta URI, será possível gerar uma URL para acesso a estas informações compartilhadas via DICOMFlow.

Por não enviar os objetos diretamente e sim os pacotes *Interest* buscarem a localização atualizada destes, é visto que o sistema irá funcionar com ou sem a adoção do modelo proposto, desta forma, serviços já implementados não sofrerão ruptura, e, logo também poderão adotar o modelo proposto de forma gradativa, já que esta nova estratégia é transparente ao usuário.

4.1. Arquitetura

Através da análise do novo fluxo, foi possível definir alguns macro requisitos necessários para o seu correto funcionamento, a gerência de nomes de exames, a gerência de comunicação entre DICOMFlow e NDN *Forwarding Daemon* e a gerência de envio e recebimento de pacotes. Uma nova camada (DICOMAddress) composta por quatro componentes (**URIGenerator, RouteRegister, PrefixRegister e SendInterest**) foi adicionada entre essas duas arquiteturas. O DICOMAddress funcionará como um *gateway*, que é uma solução capaz de contornar possíveis problemas de associação entre entidades heterogêneas (EDWARDS et al., 2007), ou seja, esta camada facilitará a associação entre o DICOMFlow e o NDN *Forwarding Daemon*. Ainda, é possível compor os elementos da nova camada sem prejudicar o funcionamento das duas arquiteturas através do uso das interfaces da camada DICOMMove (ARAUJO et al., 2016) e das interfaces do NFD (AFANASYEV et al., 2015). A Figura 5 demonstra a arquitetura do modelo proposto com a adição das novas camadas.

Figura 5 – Arquitetura do modelo proposto



Fonte: Elaborada pelo autor

Na Figura 5, é possível ver que o DICOMAddress se encontra na camada acima do DICOMFlow, funcionando como *gateway* para que os serviços DICOMFlow possam se comunicar com os serviços do NFD. Desta forma, o DICOMAddress promove uma solução de endereçamento de nomes sem gerar ruptura no funcionamento do DICOMFlow.

4.1.1. URIGenerator

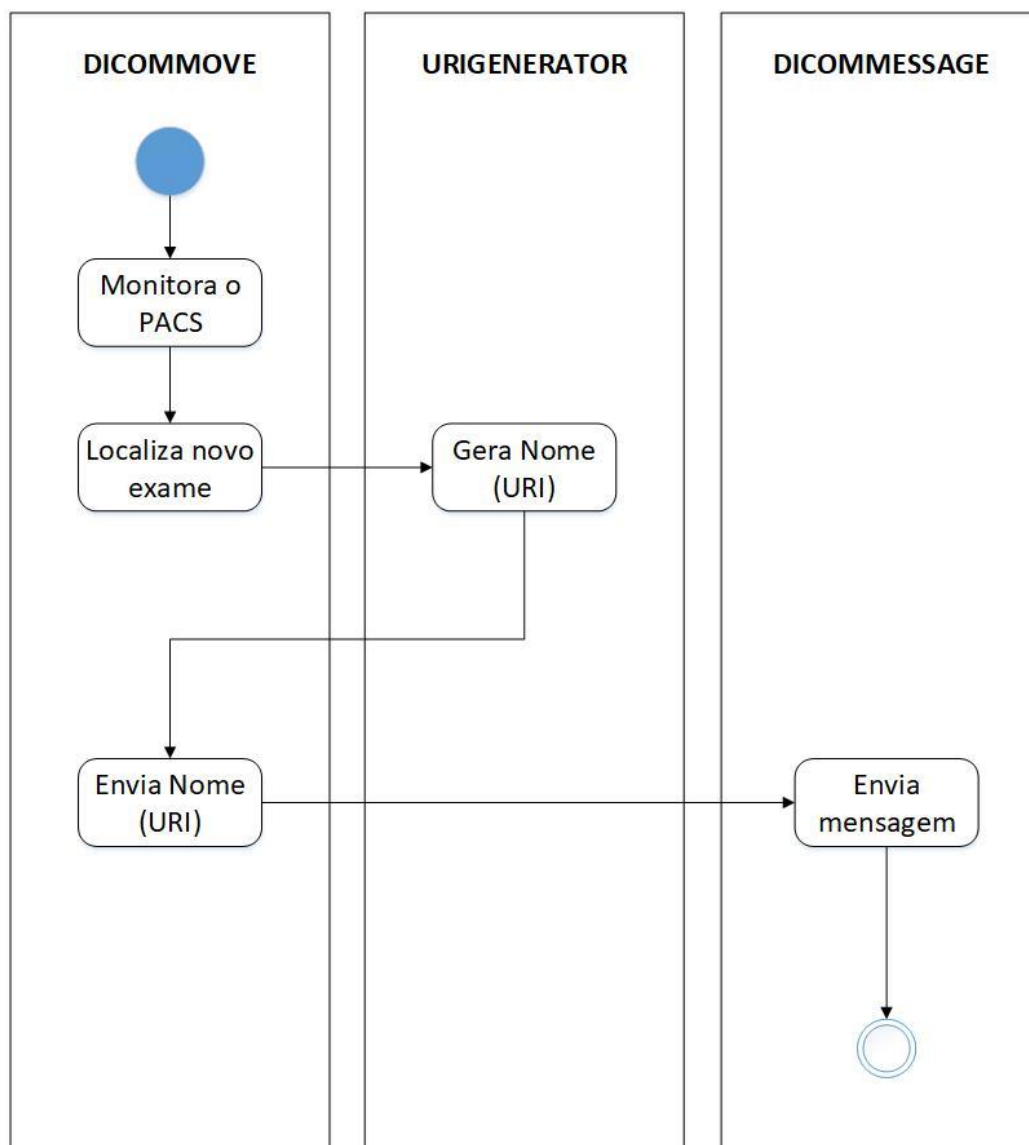
Esse componente atende ao macro requisito de gerência de nomes de exames. O DICOMAddress adota o seguinte esquema de geração de nomes, utilizando a ordem hierárquica de elementos presentes na infraestrutura de informação telerradiológica: **/dominio/idPaciente/modalidade/idDICOM**. O

prefixo **/dominio** identifica o conjunto de clínicas e hospitais que são entidades participantes do mesmo grupo, e, através dele, podem compartilhar exames médicos entre si. O **/idPaciente** indica o código do paciente, **/modalidade** referencia o tipo de exame (e.g. tomografia computadorizada, ressonância magnética) e por fim o **/idDICOM** referencia o identificador DICOM único do exame. Por exemplo, uma tomografia computadorizada gerada no hospital central da UFPB do paciente de código 1234 tem nome **/HospitalCentralUFPB/1234/ct/1.1.8.2343435.11212.224.33434**.

É importante destacar que a utilização desta hierarquia foi a fim de simplificar o entendimento deste componente, porém, a hierarquia de geração de nomes é flexível, podendo ser definida de acordo com a entidade que irá adotar o DICOMAddress, da forma que lhe seja mais adequada. Esta hierarquia foi utilizada para facilitar a busca de exames de um paciente específico ou de um conjunto de exames.

A implementação do DICOMMove possui uma interface de geração de URLs que fornece os dados necessários para a sua criação e recebe a URL formada e a encaminha nas mensagens. Logo, através dessa interface, foi possível definir uma estratégia de geração de nomes no URIGenerator que recebe os dados necessários (i.e. seu nome único do exame) e gera uma URL para acesso ao exame. A Figura 6 mostra o fluxo de geração de nomes e a sua execução ocorre durante a etapa 1 do fluxo de comunicação ilustrado na Figura 4.

Figura 6 – Fluxo de geração e envio de nomes



Fonte: Elaborada pelo autor

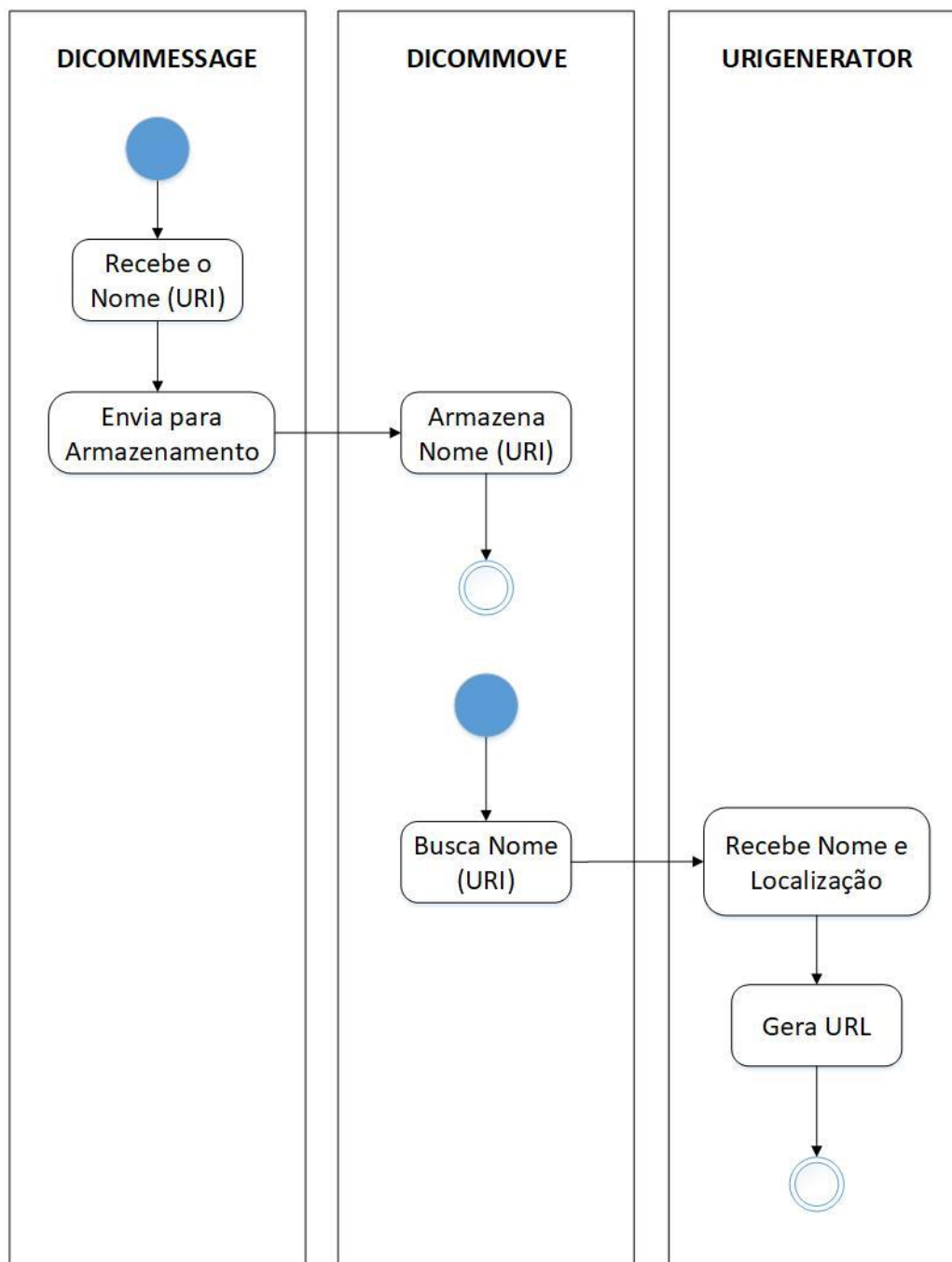
A Figura 6 apresenta este fluxo de geração de nomes, onde, ao invés de gerar a URL de acesso, o DICOMMove irá se comunicar com o URIGenerator a fim de gerar seu nome único (URI) e enviá-lo via DICOMMessage, onde o e-mail com este nome será transmitido aos participantes da rede de entidades.

Por exemplo, a Clínica de Imagens da UFPB gerou um exame de tomografia computadorizada, com identificador 1.2.840.10008.1.2. Seu DICOMMove irá monitorar seu PACS, encontrar este exame e gerar seu nome único através do módulo URIGenerator (e.g.

ClinicaDeImagensUFPB/1.2.840.10008.1.2) a fim de ser enviado para as outras entidades participantes via DICOMMessage.

O URIGenerator também lida com a geração de URLs, i.e., receber o nome único do exame e elaborar uma URL ativa para acesso ao exame.

Figura 7 – Fluxo de recebimento e busca de Nome



Fonte: Elaborada pelo autor

A Figura 7 demonstra o recebimento do nome pelo DICOMMessage da entidade receptora e o armazenamento em seu DICOMMove. Como o DICOMFlow funciona de forma assíncrona, após receber o nome do exame, este é armazenado em seu DICOMMove e pode não ser utilizado imediatamente. Quando o usuário decidir adquirir os exames, esta URI será utilizada para buscar a localização atual de onde os exames estão armazenados, e, após sua localização encontrada, o módulo URIGenerator irá gerar uma URL para acesso a estes exames.

Por exemplo, depois que a Clínica de Imagens UFPB enviou o exame de tomografia computadorizada de nome ClinicaDelmagensUFPB/1.2.840.10008.1.2 às entidades participantes da sua rede, o Hospital UFPB decide acessar e armazenar estes exames. Então ele utiliza este nome que foi recebido a fim de encontrar a localização atual do exame requerido. Após ser encontrado, o módulo URIGenerator irá gerar uma URL (e.g. <http://www.clinicaufpb.br/1.2.840.10008.1.2>) para fornecer acesso a estas informações, onde, utilizando sua estrutura do DICOMFlow será possível a obtenção destes exames.

4.1.2. SendInterest

Este componente atende ao macro requisito de gerência de pacotes. Através dele, será possível enviar pacotes *Interest*, contendo o nome único gerado pela URIGenerator, processar este nome, gerar a URL atualizada do exame requisitado e enviar como pacote *Data* ao solicitante.

4.1.3. RouteRegister e PrefixRegister

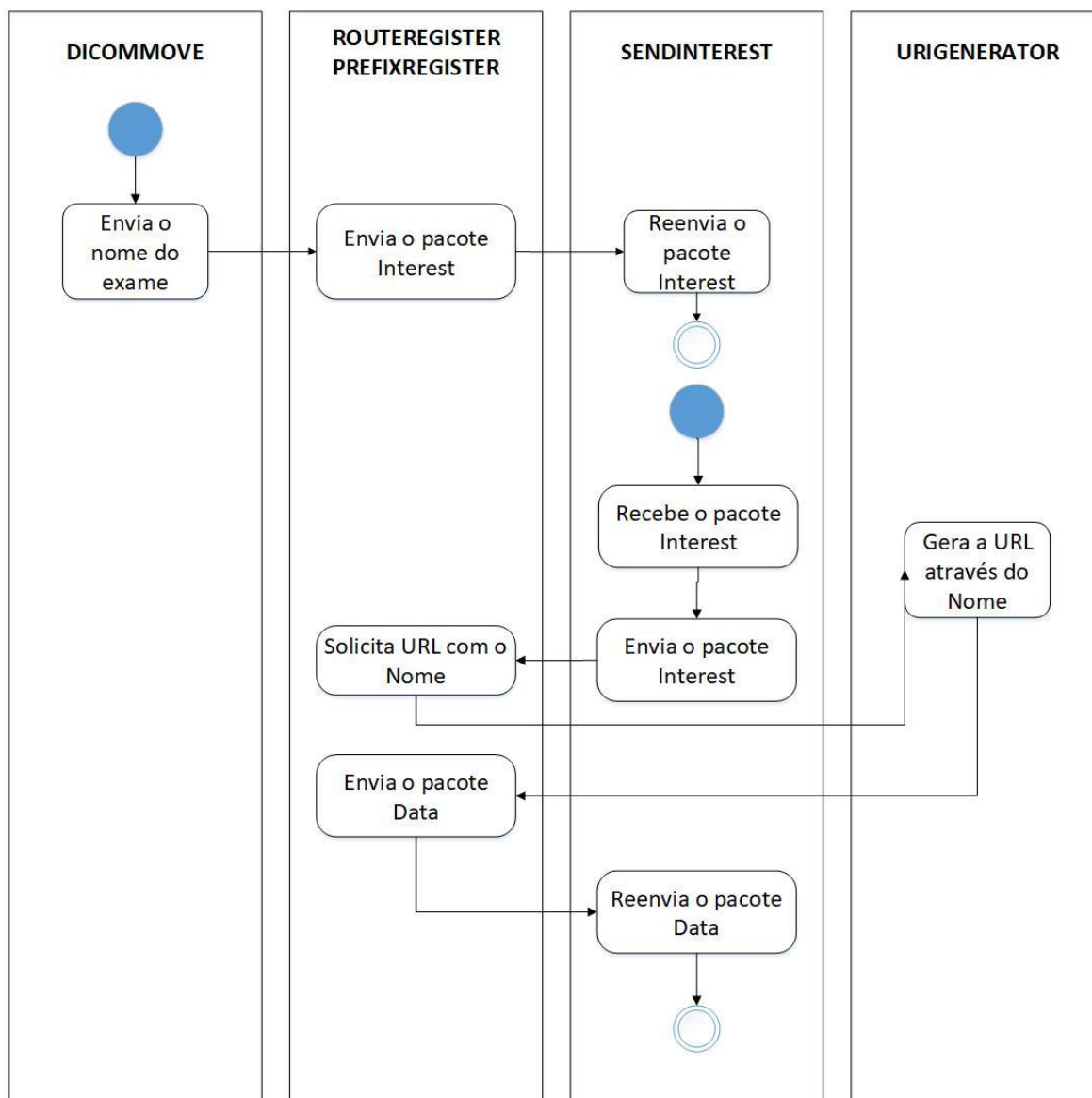
Este componente atende ao macro requisito de gerência de comunicação entre DICOMFlow e NDN *Forwarding Daemon*. Ele registra prefixos e rotas que atendem a interface *Faces* existente na implementação NFD, tornando possível receber e enviar pacotes *Interest* e *Data* configurados no SendInterest. Através

dele, será possível se comunicar com o DICOMMove, enviar um pacote *Interest* com o nome único do exame e receber o pacote *Data* com a URL atual deste.

Os componentes RouteRegister e PrefixRegister também funcionam como forma de registrar qual domínio é responsável por atender quais requisições de busca de dados, por exemplo, o domínio ClinicaDelmagensUFPB será responsável por encontrar e atender requisições de busca de nomes que possuam este prefixo, assim como o domínio HospitalUFPB será responsável por atender requisições com este prefixo. Estes componentes que registram os prefixos funcionam como forma de propagar os dados, i.e., as entidades participantes da rede serão notificadas em qual domínio buscar os dados. Caso um domínio não exista mais, estes componentes também registrarão os novos domínios responsáveis pelos nomes já existentes.

A Figura 8 mostra os fluxos de solicitação de URL através de um pacote *Interest* e sua execução ocorre durante as etapas 2, 3, 4 e 5 do fluxo de comunicação ilustrado na Figura 4. A Figura 9 de mostra o fluxo de resposta da solicitação de URL.

Figura 8 – Fluxo de solicitação de URL



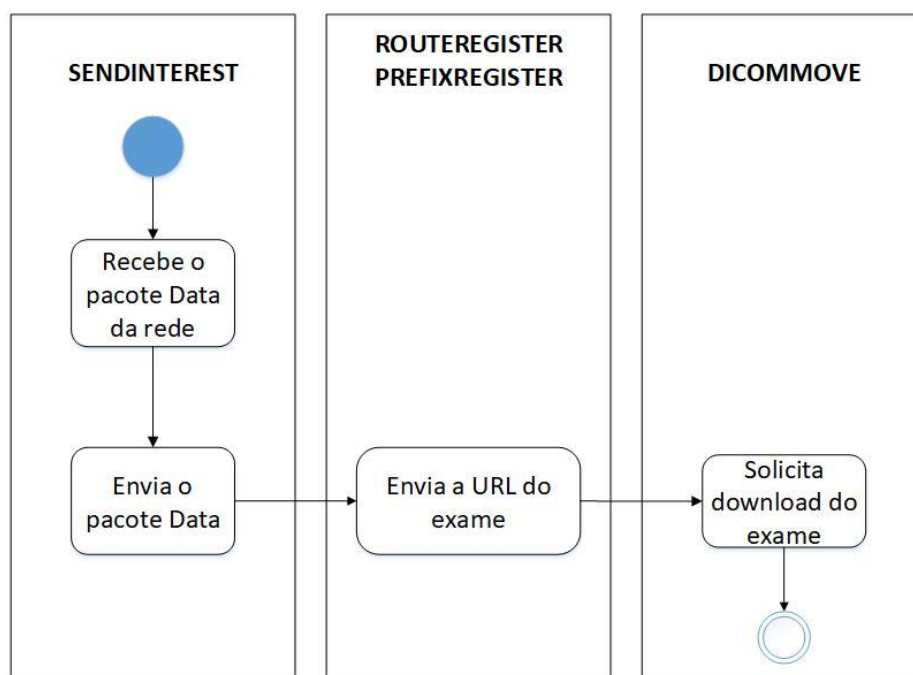
Fonte: Elaborada pelo autor

A Figura 8 demonstra o fluxo de solicitação de URL, que ocorre no lado do receptor, que deseja adquirir as informações. O RouteRegister e PrefixRegister juntamente com o SendInterest tratam do envio e recebimento de pacotes *Interest* e *Data*. O RouteRegister e PrefixRegister se encarregam da comunicação, que fazem o elo entre o DicomMove e o SendInterest. O SendInterest recebe o pacote *Interest* com o nome único e o RouteRegister e PrefixRegister solicitam uma URL através do URIGenerator, onde receberão e poderão enviar o pacote *Data* contendo esta URL.

Por exemplo, depois de receber o nome do exame, o Hospital UFPB deseja armazená-lo em seu PACS local. Então será criado um pacote *Interest* (utilizando o módulo SendInterest) a fim de buscar este exame, que vai ser enviado para o domínio ClinicaDelmagensUFPB (utilizando os módulos RouteRegister e PrefixRegister), que possui este exame. Após ser localizado, o módulo URIGenerator irá criar uma URL para acesso do exame, que retornará como um pacote *Data*, que irá fazer o caminho inverso até chegar no DICOMMove do Hospital UFPB.

É importante salientar que caso este exame tenha sido movido, os módulos RouteRegister e PrefixRegister irão buscar a localização atual das informações, para depois solicitar uma URL de acesso ao URIGenerator.

Figura 9 – Fluxo de resposta a solicitação de URL



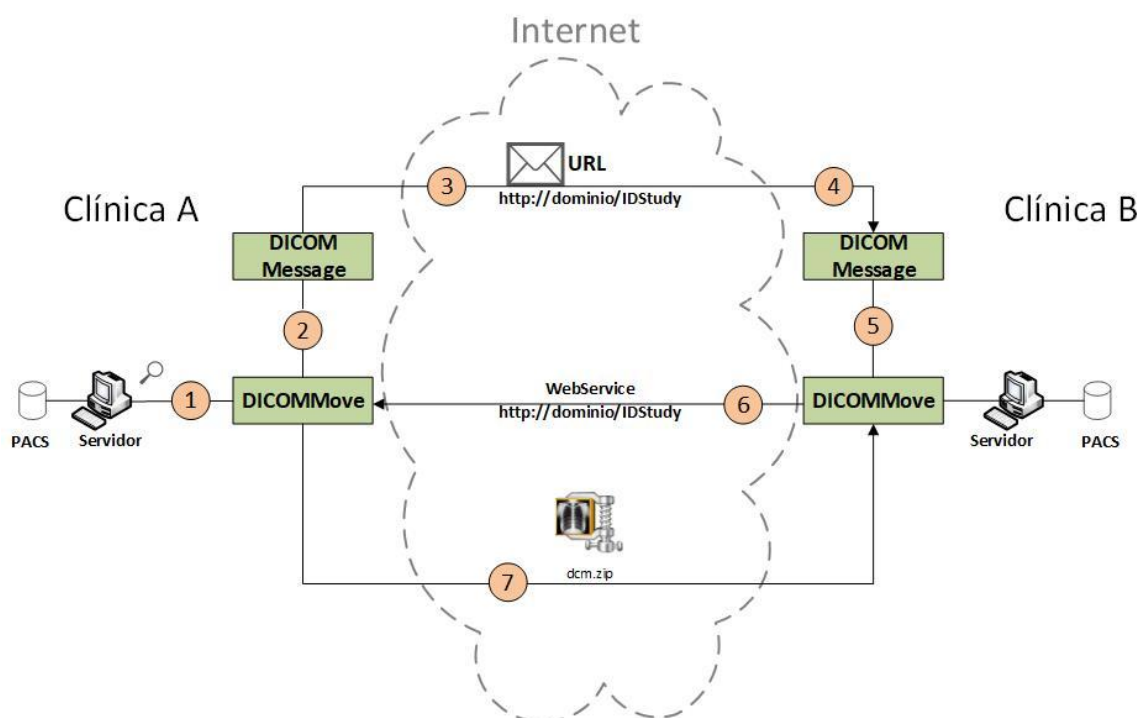
Fonte: Elaborada pelo autor

A Figura 9 mostra o fluxo de reposta da solicitação de URL através de um pacote *Data* e sua execução ocorre durante as etapas 6 e 7 do fluxo de comunicação ilustrado na Figura 4. É possível ver que o SendInterest irá receber o pacote *Data* contendo a URL, enviando via RouteRegister e PrefixRegister ao DICOMMove que poderá solicitar o *download* do exame normalmente.

Este modelo pretende não modificar o fluxo operacional do DICOMFlow, visa utilizar tecnologias *open source*, como os protocolos adotados e fazer utilização da base instalada.

Abaixo será mostrado o funcionamento simplificado do DICOMFlow original, sem a camada do DICOMAddress. Este funcionamento descreve serviços implementados no projeto DICOMFlow. Na Figura 10, o fluxo de envio de mensagem, recebimento e transferência de arquivos.

Figura 10 – Exemplo de funcionamento do DICOMFlow



Fonte: Elaborada pelo Autor

Acima, as Clínicas A e B possuem seus servidores PACS (que contém informações de diretórios e bancos de dados), e se comunicam via DICOMFlow, seguindo estes passos:

1. O DICOMMove da Clínica A estará monitorando seu servidor PACS, utilizando um serviço chamado *PersistentService*, a fim de encontrar algum dado novo;

2. Ao encontrar um novo exame, é utilizado o DICOMMessage para enviar uma mensagem para a Clínica B, utilizando o serviço *MessageService*;
3. Esta mensagem, no formato XML, contém a localização destes novos dados que são acessados via URL, esta é criada pelo serviço *URLGenerator*;
4. O DICOMMessage da Clínica B irá receber a mensagem e extrair a URL;
5. Esta URL será repassada ao seu DICOMMove;
6. O DICOMMove irá acessar esta URL diretamente, via *WebService*, o serviço que pode solicitar o *download* do exame, que controla o módulo *FileService*, responsável por criar e ler o arquivo do estudo que está armazenado no seu PACS;
7. A Clínica B, por fim, receberá o exame e poderá armazenar no seu PACS.

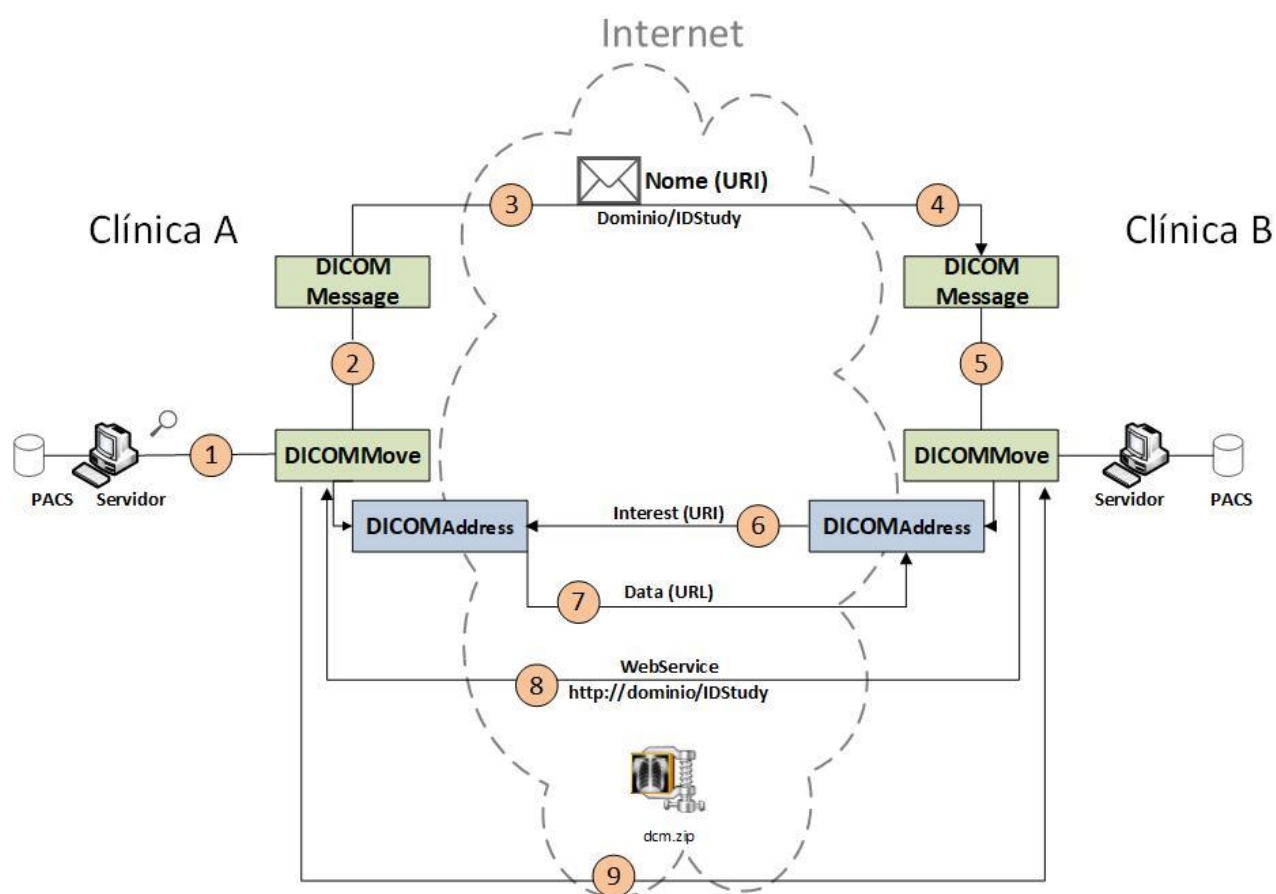
Com o objetivo de ilustrar as etapas da requisição e recebimento de imagens médicas via DICOMFlow, é importante salientar que todas as políticas de segurança de envio de imagens que conferem ao DICOMFlow estão omitidas. Para melhor ilustrar o fluxo de funcionamento do DICOMFlow, segue um exemplo, que será explicado utilizando a Figura 10:

A Clínica de Imagens UFPB está em uma rede de hospitais e deseja se comunicar com o Hospital UFPB. Por ambos estarem na mesma rede e utilizando o DICOMFlow como forma de compartilhar seus exames, a Clínica de Imagens UFPB gera uma tomografia computadorizada e armazena em seu PACS. Como o seu DICOMMove estará constantemente verificando o seu PACS (passo 1), ao encontrar este novo exame de tomografia computadorizada, o seu DICOMMessage irá enviar uma mensagem aos hospitais e clínicas participantes da sua rede (passo 2). Esta mensagem possui uma URL para acesso a este exame recém-gerado (passo 3). O Hospital UFPB recebe a mensagem em seu DICOMMessage (passo 4), onde poderá extrair e armazenar esta URL em seu DICOMMove (passo 5). Quando o Hospital UFPB desejar recuperar este exame de tomografia, o seu DICOMMove irá acessar esta URL (passo 6), que irá se comunicar com Clínica de Imagens UFPB (originadora do exame) e fornecer uma

cópia que será enviada ao Hospital UFPB (passo 7). Finalmente, o Hospital UFPB poderá receber o exame e armazenar em seu PACS.

A seguir, será mostrado o mesmo fluxo com a adição da camada DICOMAddress na Figura 11.

Figura 11 – Fluxo do DICOMFlow com a adição da camada DICOMAddress



Fonte: Elaborada pelo Autor

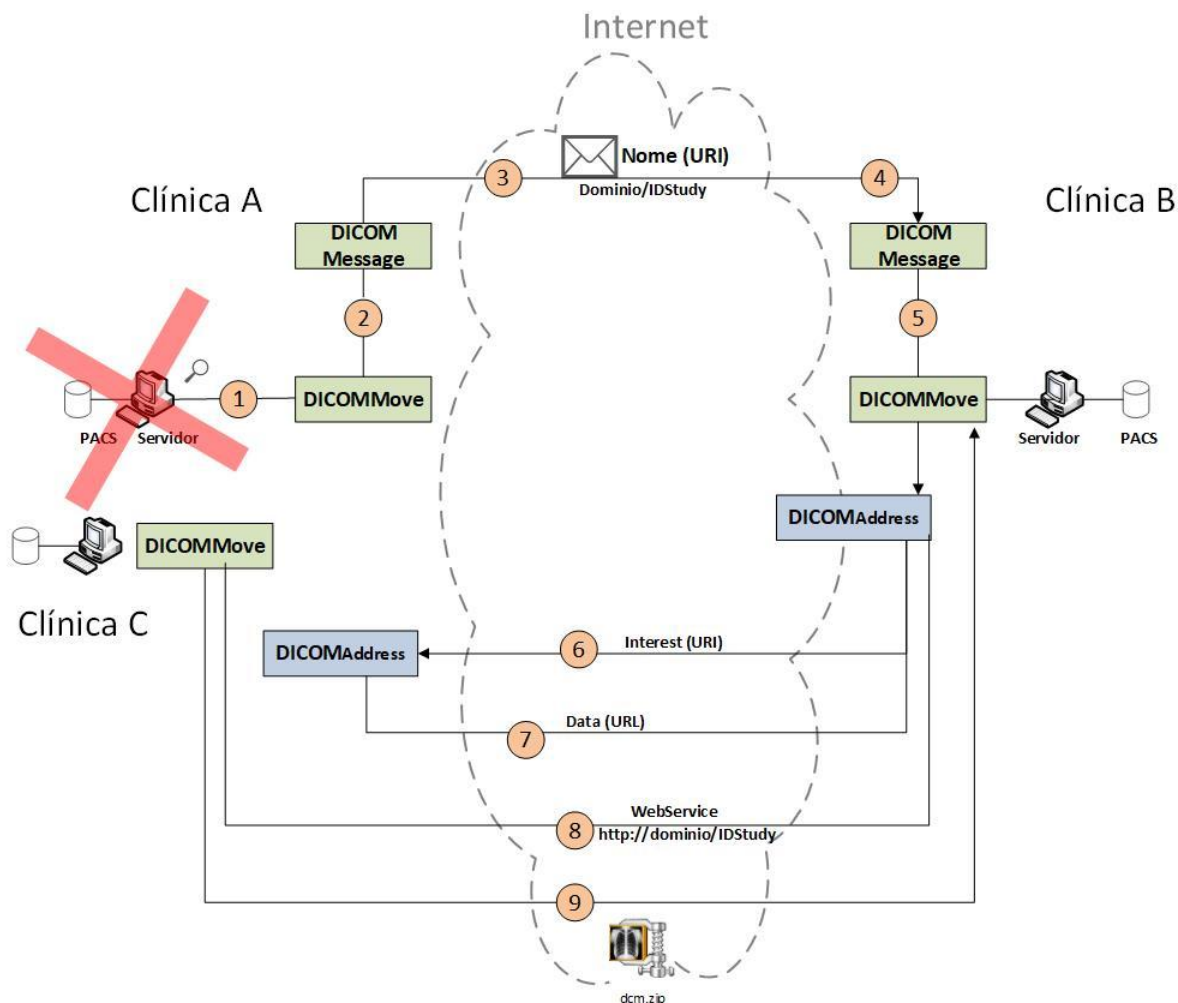
1. A clínica A, que monitora seu servidor PACS encontra um novo exame médico;
2. Então, é enviada uma mensagem via DICOMMessage a fim de alertar a Clínica B;

3. Nesta mensagem em XML, agora será enviado o nome único deste exame (ex.: clinica1/1.8.88.888.88);
4. O DICOMMessage da Clínica B irá receber esta mensagem e extrair o nome;
5. O nome será enviado ao seu DICOMMove;
6. Após isso, o DICOMMove irá se comunicar com a camada DICOMAddress, que irá criar um pacote *Interest*, que possui o nome único do exame (URI) a fim de se comunicar com o DICOMAddress da Clínica A, que irá encontrar este exame e gerar uma URL da sua localização atual;
7. É gerado um pacote *Data*, que será enviado à Clínica B contendo a URL atualizada;
8. A Clínica B poderá solicitar o *download* do exame via *WebService* ao DICOMMove da Clínica A;
9. Por fim, a Clínica B poderá receber e armazenar o exame normalmente em seu PACS.

Como o DicomFlow é utilizado de forma assíncrona, isto é, a Clínica B poderá acessar os exames posteriormente, há a possibilidade dos exames não estarem localizados onde a URL enviada apontava. É interessante salientar que não haverá uma nova forma de transmitir os exames, e sim de como reaver a localização e URL atualizadas deste exame, tornando a forma de utilização do usuário bastante similar à utilização sem a camada DICOMAddress. Também não haverá mudança na segurança do DICOMFlow, visto que o envio das informações contará com o módulo DICOMMove, que já provê a segurança necessária para estes dados.

A Figura 12 ilustra o próximo exemplo, em que há busca de uma imagem na qual sua localização inicial foi alterada. Caso a transação do exemplo anterior ocorresse normalmente até o passo 6 (busca pelo nome) o exame estivesse na Clínica C:

Figura 12 – Fluxo com localização alterada



Fonte: Elaborada pelo Autor

1. A clínica A, que monitora seu servidor PACS encontra um novo exame médico;
2. Então, é enviada uma mensagem via DICOMMessage a fim de alertar a Clínica B;
3. Nesta mensagem em XML, agora será enviado o nome único deste exame (ex.: clinica1/1.8.88.888.88);
4. O DICOMMessage da Clínica B irá receber esta mensagem e extrair o nome;
5. O nome será enviado ao seu DICOMMove;

6. Após isso, o DICOMMove irá se comunicar com a camada DICOMAddress, que irá criar um pacote *Interest*, que possui o nome único do exame (URI) a fim de buscar este exame. Após verificar que o exame não se encontra na Clínica A, utilizando a estratégia de encaminhamento, irá se comunicar com o DICOMAddress da Clínica C, que detém o exame, e gerar uma URL da sua localização atual;

7. É gerado um pacote *Data*, que será enviado à Clínica B contendo a URL atualizada;

8. A Clínica B poderá solicitar o *download* do exame via *WebService* ao DICOMMove da Clínica C;

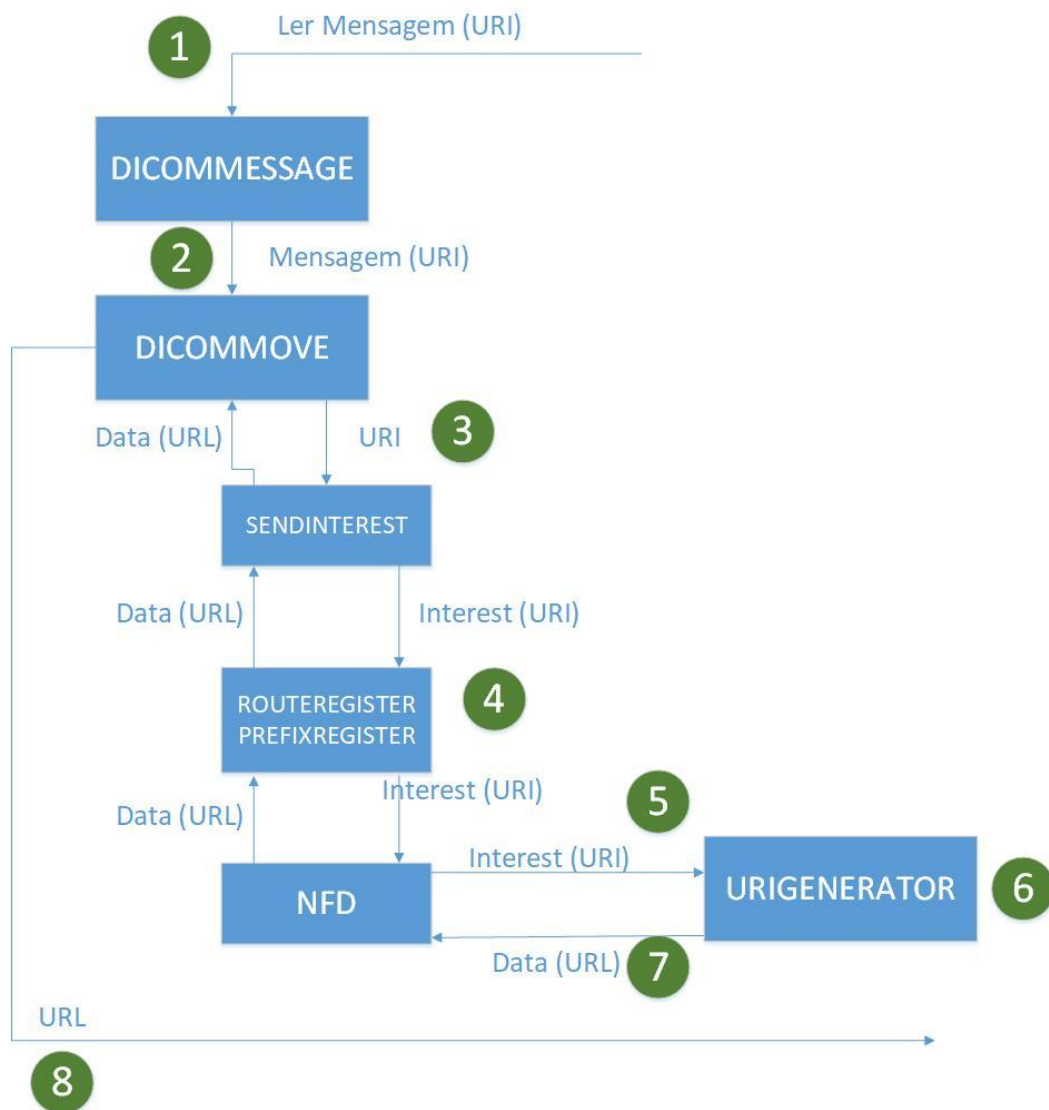
9. Por fim, a Clínica B poderá receber e armazenar o exame normalmente em seu PACS.

Retomando o exemplo anterior, entre a Clínica de Imagens UFPB e o Hospital UFPB, em que ambos estão em uma rede de hospitais. A Clínica de Imagens UFPB gera uma tomografia computadorizada e armazena em seu PACS. Como o seu DICOMMove estará constantemente verificando o seu PACS (passo 1), ao encontrar este novo exame de tomografia computadorizada, o seu DICOMMessage irá enviar uma mensagem aos hospitais e clínicas participantes da sua rede (passo 2). Esta mensagem (passo 3) contém o nome único gerado por este, através do módulo URIGenerator (e.g. clinicaufpb/ 188.088.88). O Hospital UFPB receberá esta mensagem e irá extrair este nome único do exame (passo 4) e armazenar esta URI em seu DICOMMove (passo 5). De posse deste nome único, a camada DICOMMove do Hospital UFPB irá se comunicar com o seu módulo DICOMAddress e criar um pacote *Interest*, a fim de buscar este exame. Este pacote está destinado ao DICOMAddress da Clínica de Imagens UFPB, que será recebido e irá fazer a busca da localização do exame de tomografia gerado. Após encontrado, o módulo URIGenerator irá enviar esta URL (e.g. www.ufpb.com.br/1283.8888) de volta ao DICOMAddress do Hospital UFPB (passo 7), onde, por fim este vai poder solicitar o *download* do exame utilizando a URL (passo 8) e recebê-lo (passo 9) para poder armazenar em seu PACS.

Para melhor ilustrar as etapas para a busca e retorno das informações, a

Figura 13 detalha o fluxo que a mensagem percorre na cama DICOMAddress.

Figura 13 – Fluxo da mensagem e pacotes *Interest* e *Data*



Fonte: Elaborada pelo Autor

A

Figura 13 demonstra de forma detalhada o envio da mensagem. Seguem os passos:

- 1 – O DICOMMessage irá ler a mensagem que contém o seu nome único (URI);
- 2 – Enviar esta mensagem ao DICOMMove,
- 3 – Conduzir ao *SendInterest*, que gerencia o envio e recebimento de pacotes;
- 4 – Os componentes RouteRegister e PrefixRegister, irão se comunicar com o NFD e transmitir esse pacote;
- 5 – O NFD irá receber o pacote e enviar ao componente URIGenerator;
- 6 – O URIGenerator irá converter este Nome (URI) para uma URL válida e atual, de onde os exames podem ser adquiridos;
- 7 – O pacote *Data* contendo a URL será enviado fazendo o caminho inverso;
- 8 – De posse da URL, o DICOMMove poderá acessar o exame e fazer seu *download*.

Com base nos exemplos anteriores, caso o Hospital UFPB recebesse a mensagem da Clínica de Imagens UFPB com o exame de tomografia computadorizada e armazenasse em seu DICOMMove, quando quisesse acessar este exame, a mensagem seguiria este curso:

Primeiramente, a mensagem seria lida e enviada ao DICOMMove (passos 1 e 2), onde seu nome (URI) seria extraído, depois esta URI seria utilizada como forma de buscar o exame de tomografia computadorizada (passo 3), que seguiria pelo módulo *SendInterest* sendo gerado um pacote *Interest*, este pacote seguiria pelos módulos RouteRegister e PrefixRegister (passo 4), onde será possível localizar o exame utilizando o módulo NFD (passo 5). Caso este tenha sido mudado da sua localização original, estes módulos funcionam de forma que seja possível buscar a localização atual, em que o novo prefixo registrado (domínio) será responsável por encontrar os dados (e.g. o domínio ClinicaDeImagensUFPB não existe mais, porém o novo domínio DepositoDeImagensUFPB é o responsável por responder às requisições do domínio anterior). Depois de

encontrado o exame, o módulo URIGenerator irá gerar a URL para acesso das informações (passo 6). Respondendo à solicitação do pacote *Interest*, é gerado um pacote *Data* com esta URL que fornece acesso ao exame (passo 7). Este pacote irá seguir o fluxo inverso, até chegar ao módulo DICOMMove, onde, por fim, será possível a aquisição do exame normalmente (passo 8).

Verifica-se que a adição da camada DICOMAddress em um ambiente que emprega o DICOMFlow não causa uma grande ruptura no seu fluxo de funcionamento, pois, a maioria das práticas (ou processos) que a nomeação de objetos utiliza, é transparente ao usuário.

Este capítulo apresentou o modelo básico da camada DICOMAddress, mostrando como a adoção desta irá influenciar no fluxo do DICOMFlow, como também como utilizar um nome único para identificação do exame e transformá-lo em uma URL válida, onde o usuário poderá ter acesso se valendo das mesmas políticas de segurança adotadas pelo DICOMFlow.

5. Prototipação e testes

Neste capítulo será mostrado como o DICOMADDRESS foi desenvolvido, seu funcionamento detalhado, os módulos que foram criados para sua execução e os testes criados a fim de validar o seu funcionamento.

Foi usada a implementação de referência do DICOMFlow³ e a implementação de referência do NDN para Java⁴ e foi criado um novo projeto. Nele, foram importados ambos os projetos e suas respectivas bibliotecas para o desenvolvimento do módulo DICOMAddress. Foram importadas as bibliotecas `jndn`, `jndn-management` e `jndn-mock` via Maven. Foram utilizadas as bibliotecas fornecidas pela implementação do NDN em Java, porém o NDN possui implementações em outras linguagens. Foi utilizada a implementação em Java, pois o DICOMFlow também foi desenvolvido em Java.

No projeto do DICOMMove, foram criados os pacotes `job.ndn` e `service.ndn`, a fim de adicionar a camada DICOMAddress. As classes de serviço de busca de objetos foram desenvolvidas utilizando *threads* Java, que são classes que executam instruções de forma independente (e simultânea) à execução de outras classes, isto significa que é possível fazer a busca de exames de maneira paralela.

Para desenvolver o módulo DICOMAddress, foi necessária a construção de quatro classes base: **RouteRegisterService**, **SendInterestService**, **PrefixRegisterService** e **URIGeneratorService**. Estas classes fornecem o funcionamento básico da camada apresentada no capítulo anterior, provendo gerência de comunicação, rota, envio de pacotes e geração de URLs. A seguir, será apresentada a definição de cada uma destas classes:

RouteRegisterService: Esta classe tem o objetivo de registrar a Rota para comunicação entre os componentes da rede;

³ <https://github.com/dicomflow>

⁴ <https://github.com/named-Data/jndn>

SendInterestService: Esta classe é responsável pelo envio de pacotes *Interest* e *Data*. Através dele, as URIs e URLs serão trafegadas entre os componentes da rede;

PrefixRegisterService: Esta classe tem o objetivo de registrar o prefixo utilizado na URI do exame, através dele,

URIGeneratorService: Esta classe irá obter a URI do exame e através dele, será possível gerar a nova URL, que disponibilizará as informações requeridas.

As Figuras Figura 14 e Figura 15 demonstram a classe URIGenerator com seus métodos.

Figura 14 – Método da classe URIGenerator

```
@Override
public String getURI(String studyIuid) {
    return getPrefix() + "/" + studyIuid;
}

@Override
public String getStudyIuid(String uri) {

    StringTokenizer tokenizer = new StringTokenizer(uri, "/");

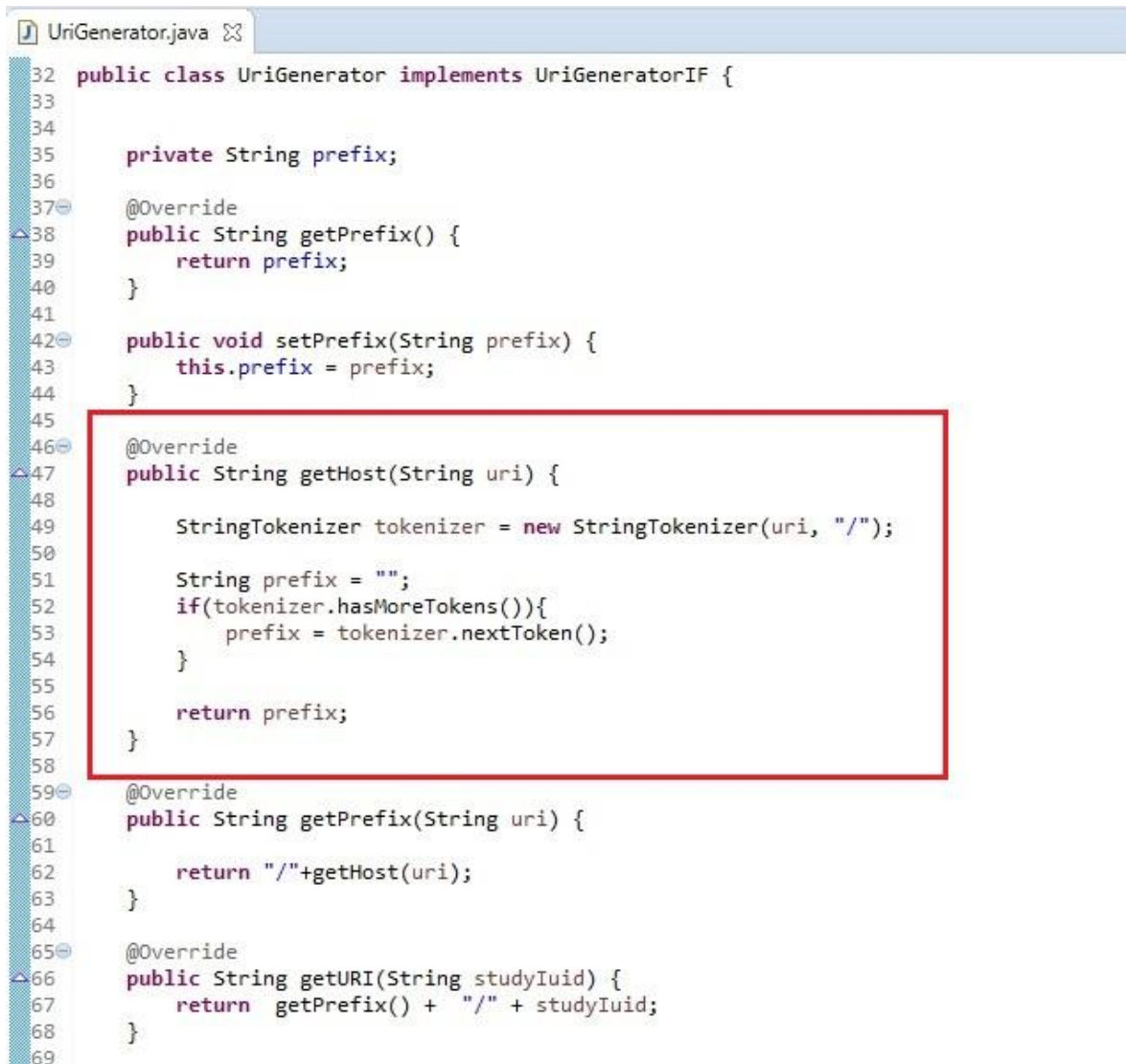
    String studyIuid = "";
    while(tokenizer.hasMoreTokens()){
        studyIuid = tokenizer.nextToken();
    }

    return studyIuid;
}
```

Fonte: Elaborada pelo autor

Na Figura 14 é possível ver os métodos **getStudyIuid** e **getURI**. O método **getStudyIuid** irá utilizar a URI a fim de encontrar o ID DICOM do estudo. E o método **getURI** irá concatenar o seu prefixo com o ID Dicom para gerar o nome do exame.

Figura 15 – Classe URIGenerator



```
32 public class UriGenerator implements UriGeneratorIF {
33
34
35     private String prefix;
36
37     @Override
38     public String getPrefix() {
39         return prefix;
40     }
41
42     public void setPrefix(String prefix) {
43         this.prefix = prefix;
44     }
45
46     @Override
47     public String getHost(String uri) {
48
49         StringTokenizer tokenizer = new StringTokenizer(uri, "/");
50
51         String prefix = "";
52         if(tokenizer.hasMoreTokens()){
53             prefix = tokenizer.nextToken();
54         }
55
56         return prefix;
57     }
58
59     @Override
60     public String getPrefix(String uri) {
61
62         return "/" + getHost(uri);
63     }
64
65     @Override
66     public String getURI(String studyIuid) {
67         return getPrefix() + "/" + studyIuid;
68     }
69 }
```

Fonte: Elaborada pelo autor

Na Figura 15 é possível ver no método **getHost** que a classe URIGenerator tem a possibilidade de obter um domínio (getHost) utilizando a URI como parâmetro, isto é, através do nome do exame (e.g. clinica1/1.8.88.888.88), será possível identificar “clinica1” como domínio do exame.

Figura 16 – Classe SendInterestService

```

14 public class SendInterestService implements SendInterestServiceIF{
15
16     private int maxAttempts;
17
18     class Counter implements OnData, OnTimeout {
19
20         public int callbackCount_ = 0;
21         public String return_ = "";
22
23         public void onData(Interest interest, Data data) {
24             ++callbackCount_;
25             System.out.println("Got data packet with name " + data.getName().toUri());
26             ByteBuffer content = data.getContent().buf();
27
28             for (int i = content.position(); i < content.limit(); ++i)
29                 return_ += content.get(i);
30         }
31
32         public void onTimeout(Interest interest) {
33             ++callbackCount_;
34             System.out.println("Time out for interest " + interest.getName().toUri());
35             return_ = "";
36         }
37     }
38 }

```

Fonte: Elaborada pelo autor

Na Figura 16 é possível observar que os métodos **onData** e **onTimeout** para busca de pacotes *Interest* são satisfeitos quando recebem um pacote *Data* com o nome correspondente. Desta forma, quando os pacotes *Interest* são criados, eles podem ser satisfeitos por encontrar o dado pesquisado, ou podem ser encerrados por esgotamento de tempo (assim encerrando a *thread* que faz a busca destes dados).

Para automatizar o DICOMMove, existe um conjunto de classes que executam uma ação em tempos determinados, chamados de *Jobs*, que são definidos em arquivo de configuração, este arquivo também determina a frequência de quando serão executadas estas ações.

Como o DICOMFlow tem seu funcionamento através de *Jobs*, também foram criados *Jobs* específicos a fim de automatizar o módulo DICOMAddress, que são: FindNDNStorages, PrepareNDNStorages, ProcessNDNStorages, ProcessPendingNDNStorages, SendNDNStorages, StoreNDNStudies, StorePendingNDNStudies e VerifyPrefixRegisterService.

Abaixo, a Tabela 2 irá detalhar cada *job* e respectiva sua ação.

Tabela 2 – Detalhamento dos *Jobs* e suas ações

Nome do <i>Job</i>	Ação
FindNDNStorages	Busca de novos estudos de imagens médicas
PrepareNDNStorages	Prepara o armazenamento dos novos estudos de imagens médicas
ProcessNDNStorages	Processa registros abertos a fim de gerar URLs
ProcessPendingNDNStorages	Processa registros abertos pendentes a fim de gerar URLs
SendNDNStorages	Envio de estudos de imagens médicas
StoreNDNStudies	Armazenamento de estudos de imagens médicas
StorePendingNDNStudies	Armazenamento de estudos de imagens médicas pendentes
VerifyPrefixRegisterService	Verifica o registrador de prefixos NDN

Fonte: Elaborada pelo autor

Esses *Jobs* estão registrados no arquivo de configuração. A Figura 17, demonstra o arquivo de configuração ApplicationContext.xml onde define os *Jobs* necessários para a automatização do DICOMAddress.

Figura 17 – Definição dos *Jobs*

```

<!-- JOBS PARA VERIFICAR O REGISTRADOR DE PREFIXOS NDN -->
<ref bean="verifyPrefixRegisterServiceCronTrigger"/>

<!-- JOBS PARA ATUALIZACAO DE ACCESSES E SERVICE PERMISSIONS -->
<ref bean="findAccessCronTrigger"/>

<!-- JOBS PARA ENVIO E RECEBIMENTO DE CERTIFICADOS -->
<ref bean="sendCertificatesCronTrigger"/>
<ref bean="findCertificatesCronTrigger"/>
<ref bean="verifyCertificateResultCronTrigger"/>
<ref bean="verifyCertificateConfirmCronTrigger"/>

<!-- JOBS PARA ENVIO DE STORAGE SAVE -->
<ref bean="prepareStoragesCronTrigger"/>
<ref bean="sendStoragesCronTrigger"/>
<ref bean="verifyStorageResultsCronTrigger"/>

<!-- JOBS PARA RECEBIMENTO DE STORAGE SAVE -->
<ref bean="findStoragesCronTrigger"/>
<ref bean="processStoragesCronTrigger"/>
<ref bean="processPendingStoragesCronTrigger"/>
<ref bean="storeStudiesCronTrigger"/>
<ref bean="storePendingStudiesCronTrigger"/>

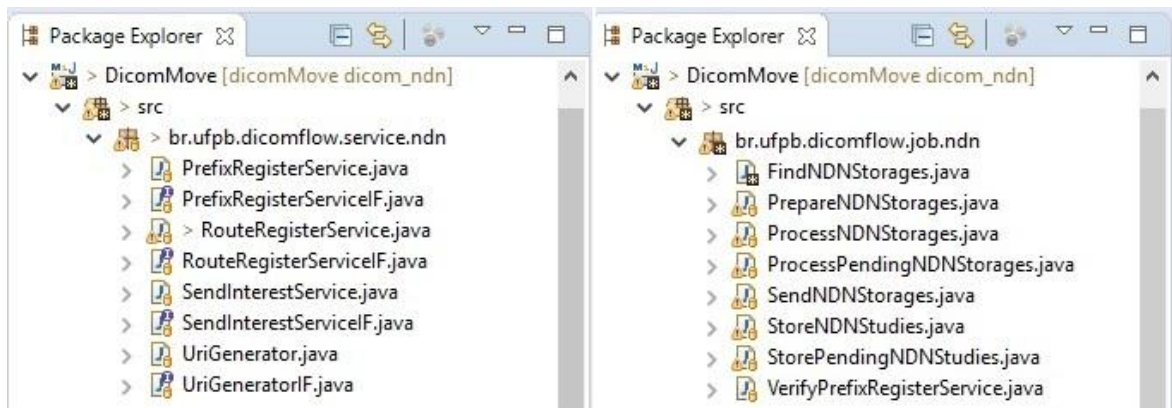
<!-- JOBS PARA ENVIO DE REQUEST PUT -->
<ref bean="prepareRequestsCronTrigger"/>
<ref bean="sendRequestsCronTrigger"/>
<ref bean="verifyRequestResultsCronTrigger"/>

```

Fonte: Elaborada pelo Autor

Na Figura 17 é possível verificar os *Jobs* DICOMAddress juntamente com os *Jobs* do DICOMFlow funcionando em conjunto. A Figura 18 mostra os pacotes produzidos no novo projeto.

Figura 18 – Pacotes adicionados ao novo projeto



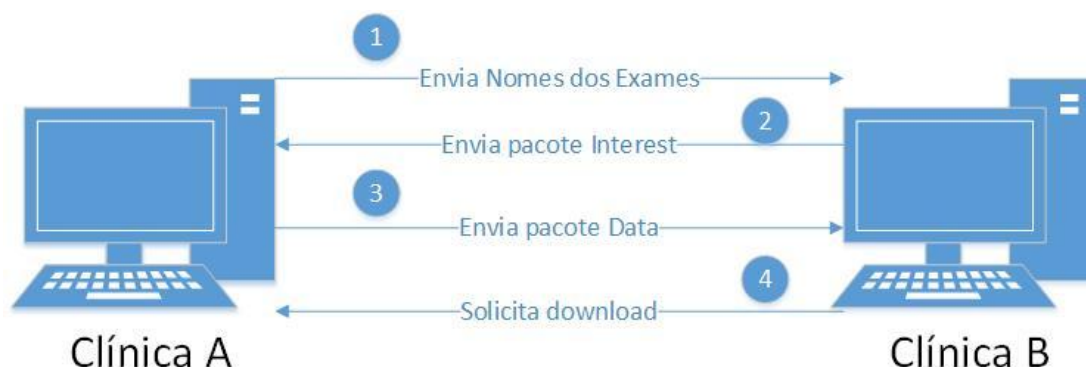
Fonte: Elaborada pelo autor

5.1. Experimento preliminar

Para validar o funcionamento da camada DICOMAddress, foi realizado um teste de viabilidade técnica em ambiente simulado, a fim de verificar de forma preliminar a adoção do modelo proposto. Foram simuladas duas clínicas associadas (Clínica A e Clínica B), ambas com seus servidores PACS com seus módulos DICOMFlow e sua camada DICOMAddress previamente configurados.

Este experimento teve como objetivo testar o modelo proposto e verificar o funcionamento dos módulos criados. Com ele, foi possível também afirmar sua validação. O experimento ocorreu da seguinte forma: A Clínica A gerou 10 exames de tomografia computadorizada e enviou e-mail via DICOMMessage para a Clínica B, que recebeu os nomes dos exames (e.g. clinicaa/1283.8888). Depois de receber estes nomes, a Clínica B utilizou o nome recebido e através do prefixo, foi possível buscar a localização dos exames (que estavam armazenados na Clínica A). Isso foi possível ao criar um pacote *Interest*, até chegar ao NFD da Clínica A. Após localizar os exames, o módulo URIGenerator da Clínica A gerou a URL para acesso aos exames e enviou como um pacote *Data*, que a Clínica B recebeu e depois foi possível fazer a aquisição dos exames normalmente, utilizando DICOMFlow. A Figura 19 ilustra o fluxo básico do teste realizado.

Figura 19 – Fluxo do teste preliminar de funcionamento do DICOMAddress

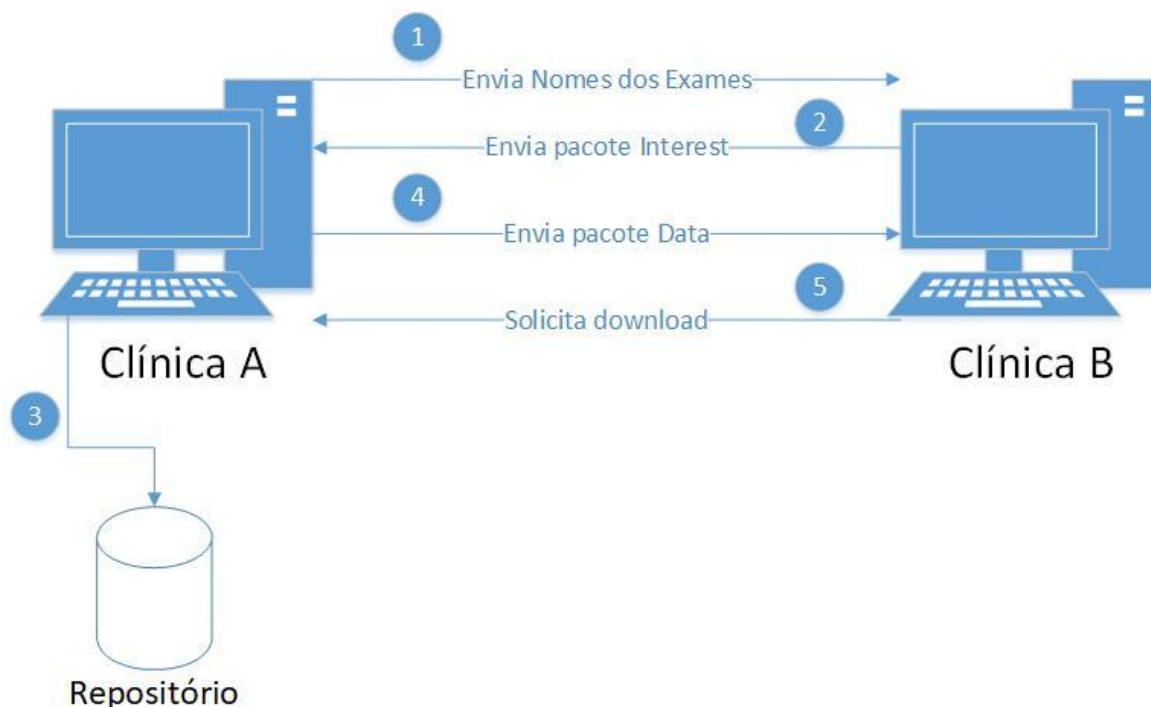


Fonte: Elaborada pelo Autor

Primeiramente, a Clíника A gerou as imagens (exames) e enviou os nomes via DICOMMessage como URIs (passo 1), após receber os nomes, o DICOMMove da Clíника B utilizou estas URIs a fim de buscar estes exames. Criou um pacote *Interest* e enviou à clíника A (passo 2). A Clíника A recebe este pacote *Interest* e vai buscar a localização dos exames em seu PACS. Após encontrá-los, envia dados ao módulo URIGenerator a fim de criar uma URL que forneça acesso às imagens. Então envia à Clíника B um pacote *Data* (passo 3), que possui esta URL que aponta onde estão os exames. Por fim, de posse desta URL, a Clíника B pode adquirir os exames utilizando o DICOMFlow (passo 4), para armazenar em seu PACS local.

Como os exames estavam armazenados no originador das imagens, foi possível verificar o funcionamento dos módulos e também a geração de nomes e URLs, porém não havia sido testada a funcionalidade do NFD, que é de buscar os dados através do seu nome único. Para isto, foi efetuado um segundo teste, onde a Clíника A gerou as imagens e enviou os nomes, porém a localização dos exames foi mudada para um repositório externo. Então, através do *NDN Forward Daemon* foi possível encontrar os exames e gerar a URL para acesso a estes.

Figura 20 – Fluxo do segundo teste de funcionamento do DICOMAddress



Fonte: Elaborada pelo Autor

A Clínica A gerou as imagens (exames) e enviou os nomes via DICOMMessage como URIs (passo 1), após receber os nomes, o DICOMMove da Clínica B utilizou estas URIs a fim de buscar estes exames. Criou um pacote *Interest* e enviou à clínica A (passo 2). A Clínica A recebe este pacote *Interest* e vai buscar utilizando seus módulos RouteRegister e PrefixRegister a localização atual dos exames. Aqui foi possível notar que o NFD localizou os exames mesmo com estes movidos para um repositório. Desta forma atualizando os módulos RouteRegister e PrefixRegister com o novo domínio para acesso aos exames (passo 3). Após encontrá-los, envia dados ao URIGenerator a fim de criar uma URL que forneça acesso às imagens. Então envia à Clínica B um pacote *Data* (passo 4), que possui esta URL que aponta onde estão os exames. Por fim, de posse desta URL, a Clínica B pode adquirir os exames utilizando o DICOMFlow (passo 5), para armazenar em seu PACS local.

Este experimento preliminar foi efetuado em laboratório, com a utilização de duas máquinas. A Tabela 3 demonstra a configuração das máquinas.

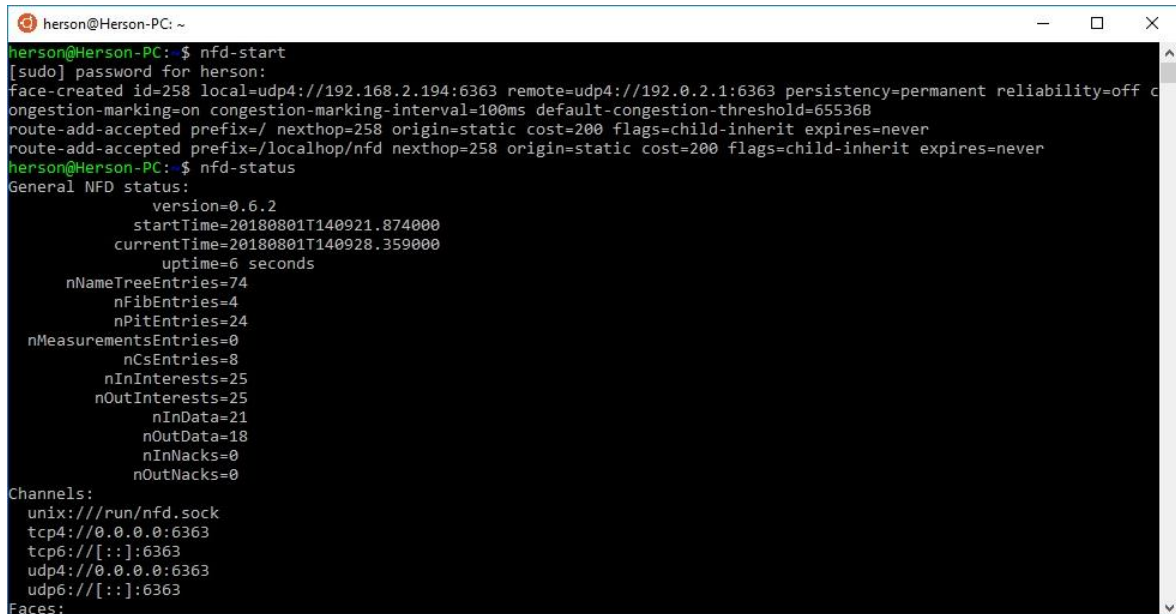
Tabela 3 – Configuração das máquinas do experimento

Recurso	Máquina 1	Máquina 2
CPU	Intel® Core™ i3-5005U 2.00 Ghz	Intel® Core™ i5-8500B 2.00 Ghz
Memória	4GB DDR3	8GB DDR3
Sistema Operacional	Windows 10 X64	Windows 10 X64
JVM	versão 1.7.x+JDK	versão 1.7.x+JDK
Banco de dados	MySQL 5.1	MySQL 5.1
DCM4CHEE	versão 2.18.x	versão 2.18.x
DCM4CHEE Toolkit	versão 2.0.23	versão 2.0.23
DICOMFlow	versão 1.2	versão 1.2
JBOSS	versão 4.2.3.GA	versão 4.2.3.GA
TOMCAT	versão 7.0.29	versão 7.0.29

Fonte: Elaborada pelo Autor

Como a solução NDN atualmente funciona apenas em sistemas baseados em Unix, como macOS e em sistemas Linux, não tendo funcionamento nativo através do Windows, foi necessário adicionar o interpretador de comandos do Linux Ubuntu no Windows 10 (*Bash*). Com ele, foi possível utilizar os serviços NDN no Windows 10. Após a adição deste interpretador, foi possível usar normalmente os serviços NDN. A Figura 21 mostra o aplicativo Ubuntu funcionando no Windows 10.

Figura 21 – Funcionamento do Bash Ubuntu no Windows 10



```
herson@Herson-PC: ~
herson@Herson-PC: $ nfd-start
[sudo] password for herson:
face-created id=258 local=udp4://192.168.2.194:6363 remote=udp4://192.0.2.1:6363 persistency=permanent reliability=off c
ongestion-marking=on congestion-marking-interval=100ms default-congestion-threshold=65536B
route-add-accepted prefix=/ nexthop=258 origin=static cost=200 flags=child-inherit expires=never
herson@Herson-PC: $ nfd-status
General NFD status:
    version=0.6.2
    startTime=20180801T140921.874000
    currentTime=20180801T140928.359000
    uptime=6 seconds
    nNameTreeEntries=74
    nFibEntries=4
    nPitEntries=24
    nMeasurementsEntries=0
    nCsEntries=8
    nInInterests=25
    nOutInterests=25
    nInData=21
    nOutData=18
    nInNacks=0
    nOutNacks=0
Channels:
    unix:///run/nfd.sock
    tcp4://0.0.0.0:6363
    tcp6://[:]:6363
    udp4://0.0.0.0:6363
    udp6://[:]:6363
Faces:
```

Fonte: Elaborada pelo Autor

Este capítulo apresentou o funcionamento detalhado da camada DICOMAddress, a criação dos seus componentes e seu fluxo de trabalho. Também explicou brevemente como foi efetuado seu experimento preliminar a fim de validar o funcionamento do modelo proposto. Esse experimento possibilitou verificar a funcionalidade do modelo proposto e o funcionamento do NDN *Forwarding Daemon*.

6. Discussão

Este trabalho apresentou um modelo em que utiliza o NFD e o DICOMFlow a fim de gerar uma solução para nomeação de exames de imagens médicas. Este modelo foi planejado em forma de camada, de modo que a sua adição não cause ruptura no fluxo de funcionamento do DICOMFlow.

A solução se apresenta de acordo com as características do DICOMFlow e do NDN, i.e., são distribuídos, o que significa que não há uma estrutura central que controla o fluxo de mensagens, de nomeação de objetos e de busca destes objetos. Por o DICOMFlow ter caráter assíncrono, i.e., o usuário pode apenas receber o aviso de que há um novo exame disponível, então ele pode não ter a necessidade de fazer o acesso imediatamente. Com isso, faz-se necessário a persistência de acesso a estes exames, fazendo com que os exames sejam encontrados, mesmo que sua localização física seja alterada.

A utilização da camada DICOMAddress em uma arquitetura que utiliza o DICOMFlow se mostra de fácil adoção, pois além de não pretender mudar o fluxo de trabalho do ambiente que o adota, esta solução se apresenta transparente ao usuário, i.e., os usuários não precisam saber da localização exata do recurso, precisam apenas acessá-los. Após recuperar a URL de acesso ao exame, o usuário pode acessar estas informações normalmente, protegidas pelas políticas de segurança utilizadas no DICOMFlow.

Comparando este trabalho às soluções que utilizam federação DICOM (Sharghi et al. 2015; Sharma et al. 2009), as soluções existentes tratam dos objetos DICOM e do seu compartilhamento, enquanto este trabalho se limita a compartilhar apenas a localização atualizada, fornecendo um *link* para acesso aos exames, não realizando o transporte dos mesmos. Em relação à solução de *blockchain* (Zyskind et al. 2015), este trabalho se alinha mais com o DICOMFlow, por ser concebido com o objetivo de solucionar o problema da persistência de URLs trocadas no compartilhamento de exames médicos. Diferentemente de blockchains, o DICOMAddress não necessita de uma rede robusta e com um grande número de usuários para que seu funcionamento se torne ideal.

6.1. Limitações

Este trabalho atingiu algumas limitações em seu desenvolvimento. Como ponto negativo, a solução NDN não possui muitos exemplos práticos empregados, desta forma, a utilização desta solução se mostrou de difícil entendimento e desenvolvimento.

Também como limitação, o modelo não sofreu um experimento robusto, com a utilização de vários servidores, onde seria possível adicionar cenários de utilização do modelo, para que fossem realizados testes de desempenho, de carga, e de stress, a fim de validar sua funcionalidade nas atribuições do seu funcionamento cotidiano.

6.2. Considerações finais

Este projeto sugeriu uma nova forma de lidar com a falha de referências que uma URL pode conter (i.e., uma URL estar inválida), pois, o recurso que ela apontava, já não está mais disponível. A forma de tratar este problema foi utilizando a nomeação de objetos, ideia presente na solução *Named Data Networking*, onde os objetos não são mais buscados pela sua localização, e sim pelo seu nome único.

A aplicação deste trabalho, utilizando o DICOMFlow, uma forma de compartilhar exames de internet via mensagens (e-mail) foi de bastante valia, pois, o DICOMFlow se vale de URLs para poder transportar e adquirir exames via internet.

Foi criada uma camada DICOMAddress, que funciona como um *gateway*, um elo entre o DICOMFlow e a implementação do encaminhador do NDN, o NFD. Esta camada funcionou de forma transparente ao usuário, ou seja, ela não interrompeu o fluxo de funcionamento do DICOMFlow, apenas adicionou uma nova forma de recuperar estes exames.

Esta camada não pretende realizar o transporte dos exames, e sim, uma nova forma de poder reaver estes por meio de seu nome único (URI), onde através dos módulos criados, foi possível gerar uma URL válida e atualizada de onde o recurso está disponível.

No seu teste preliminar, a camada obteve sucesso ao encontrar os exames através do seu nome único, mesmo que estes tenham sido movidos da sua localização original e poder gerar a URL atualizada de onde este exame se encontra, realizando o transporte do exame requerido se valendo das políticas de segurança empregadas no DICOMFlow. A utilização do protocolo NDN neste projeto resolve o problema supracitado, se mantendo uma solução distribuída e assíncrona.

Como trabalhos futuros, pretende-se realizar testes mais robustos em ambiente virtual, com a adição de cenários de uso, onde os resultados serão expostos em um artigo científico.

7. Referências Bibliográficas

AANESTAD, M. et al. Information Infrastructures and the Challenge of the Installed Base. In: **Information Infrastructures within European Health Care**. Springer, Cham, 2017. p. 35-51.

AFANASYEV, A. et al. NFD Developer's Guide. **NDN, Technical Report NDN-0021**, v. 4, p. 1–56, 2015.

ARAÚJO, Danilo AB et al. DicomFlow: Um modelo de arquitetura para formação de uma infraestrutura de informação telerradiológica. In: **Proceedings of the XII Brazilian Symposium on Information Systems**. 2016. p. 458-465.

BARNEVELD BINKHUYSEN, F. H.; RANSCHAERT, E. R. Teleradiology: Evolution and concepts. **European Journal of Radiology**, v. 78, n. 2, p. 205–209, 2011.

BOWKER, G. C. et al. Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment. In: **International Handbook of Internet Research**. Dordrecht: Springer Netherlands, 2010. p. 119–133.

BRAY, T. et al. Extensible markup language (XML). **World Wide Web Journal**, v. 2, n. 4, p. 27–66, 1997.

EDWARDS, P. N. et al. Understanding infrastructure: Dynamics, tensions, and design. **Report of a Workshop on “History & Theory of Infrastructure: Lessons for New Scientific Cyberinfrastructures”**, n. January, p. 54, 2007.

FIGUEIREDO, J. F. M.; MOTTA, G. H. M. B. SocialRAD: an infrastructure for a secure, cooperative, asynchronous teleradiology system. **Studies in health technology and informatics**, v. 192, n. MEDINFO 2013, p. 778–82, jan. 2013.

GRASSI, Giulio et al. Navigo: Interest forwarding by geolocations in vehicular named data networking. In: **World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a. IEEE**, p. 1-10. 2015.

HANSETH, O.; LYYTINEN, K. Design theory for dynamic complexity in information infrastructures: The case of building internet. **Journal of Information Technology**, v. 25, n. 1, p. 1–19, 2010.

HUANG, H. K. **PACS and imaging informatics: basic principles and applications**. John Wiley & Sons, 2011.

JACOBSON, V. et al. Networking named content. **Communications of the ACM**, v. 55, n. 1, p. 117, 1 jan. 2012.

LAWRENCE, S. et al. Persistence of web resources in scientific research. **Computer**, v. 34, n. 2, p. 26–31, 2001.

LUCENA-NETO, J. R. et al. DicomFlow: infraestrutura assíncrona, assimétrica, descentralizada para distribuição de imagens médicas. **Journal of Health Informatics**, p. 327-338, 2016.

MONTEIRO, E. et al. From Artefacts to Infrastructures. **Computer Supported Cooperative Work (CSCW)**, v. 22, n. 4–6, p. 575–607, 2013.

MOTTA, G. H. M. B. Towards social radiology as an information infrastructure: Reconciling the local with the global. **Journal of Medical Internet Research**, v. 16, n. 10, p. e27, 2014.

NAIK, B. B. et al. Security attacks on information centric networking for healthcare system. **Advanced Communication Technology (ICACT), 2017 19th International Conference on**, p. 436–441, 2017.

OEHLMANN, F. Content-Centric Networking. **Network**, v. 43, n. February, p. 11–18, 2013.

PASKIN, N. E-citations: actionable identifiers and scholarly referencing. **Learned Publishing**, v. 13, n. 3, p. 159–166, 2000.

PIANYKH, O. S. **Digital Imaging and Communications in Medicine (DICOM)**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

SAINI, A.; YADAV, P. K. **Distributed System and its Role in HealthCare System**. v. 4, n. 4, p. 302–308, 2015.

SAXENA, D.; ROORKEE, I. I. T. Named *Data* Networking: A Survey. **Computer Science Review, Elsevier**, v. 19, p. 15–55, 2016.

SHANG, W. et al. Securing building management systems using named *Data* networking. **IEEE Network**, v. 28, n. 3, p. 50–56, 2014.

SHARGHI, Hassan; MA, Weina; SARTIPI, Kamran. Federated service-based authentication provisioning for distributed diagnostic imaging systems. In: **Computer-Based Medical Systems (CBMS), 2015 IEEE 28th International Symposium on**. IEEE, 2015. p. 344-347.

SHARMA, Ashish et al. VirtualPACS—a federating gateway to access remote image data resources over the grid. **Journal of digital imaging**, v. 22, n. 1, p. 1-10, 2009.

TROSSEN, D.; SARELA, M.; SOLLINS, K. Arguments for an information-centric internetworking architecture. **ACM SIGCOMM Computer Communication Review**, v. 40, n. 2, p. 26, 2010.

VREELAND, A. et al. Considerations for Exchanging and Sharing Medical Images

for Improved Collaboration and Patient Care: HIMSS-SIIM Collaborative White Paper. **Journal of Digital Imaging**, v. 29, n. 5, p. 547–558, 2016.

WANG, L. et al. OSPFN: An OSPF Based Routing Protocol for Named *Data* Networking. **University of Memphis and University of Arizona, Tech. Rep**, p. 1–15, 2012.

ZHANG, H. et al. Sharing mHealth *Data* via Named *Data* Networking. **Proceedings of the 2016 conference on 3rd ACM Conference on Information-Centric Networking - ACM-ICN '16**, n. January, p. 142–147, 2016.

ZHANG, L. et al. Named *Data* networking. **ACM SIGCOMM Computer Communication Review**, v. 44, n. 3, p. 66–73, 2014.

ZYSKIND, Guy et al. Decentralizing privacy: Using blockchain to protect personal data. In: **Security and Privacy Workshops (SPW), 2015 IEEE**. IEEE, 2015. p. 180-184.