
Lisiane Maria Bannwart Ambiel

# Redes Orientadas a Conteúdo: Uma abordagem no Nível de Enlace

Campinas
2013

**Universidade Estadual de Campinas**

**Faculdade de Engenharia Elétrica e de Computação**

Lisiane Maria Bannwart Ambiel

# Redes Orientadas a Conteúdo: Uma abordagem no Nível de Enlace

Dissertação de Mestrado apresentada ao programa de Pós-Graduação em Engenharia Elétrica da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestra em Engenharia Elétrica.
Área de concentração: Engenharia de Computação.

Orientador: Prof. Dr. Maurício Ferreira Magalhães
Coorientador: Prof. Dr. Christian Rodolfo Esteve Rothenberg

Este exemplar corresponde à versão final da dissertação defendida pela aluna Lisiane Maria Bannwart Ambiel e orientada pelo Prof. Dr. Maurício Ferreira Magalhães.

Campinas
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

Am16r

Ambiel, Lisiane Maria Bannwart, 1962-
Redes orientadas a conteúdo : uma abordagem no nível de enlace / Lisiane Maria Bannwart Ambiel. – Campinas, SP : [s.n.], 2013.

Orientador: Maurício Ferreira Magalhães.
Coorientador: Christian Rodolfo Esteve Rothenberg.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Arquitetura de redes de computadores. 2. Redes de computadores - Protocolos. 3. Roteamento (Administração de redes de computadores). 4. Internet. I. Magalhães, Maurício Ferreira,1951-. II. Rothenberg, Christian Rodolfo Esteve. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Informações para Biblioteca Digital

**Título em outro idioma:** Content oriented networking : a link layer approach
**Palavras-chave em inglês:**
Computer networks architectures
Computer networks - Protocols
Routing (computer network management)
Internet
**Área de concentração:** Engenharia de Computação
**Titulação:** Mestra em Engenharia Elétrica
**Banca examinadora:**
Maurício Ferreira Magalhães [Orientador]
Fábio Luciano Verdi
Marco Aurélio Amaral Henriques
**Data de defesa:** 19-06-2013
**Programa de Pós-Graduação:** Engenharia Elétrica

## COMISSÃO JULGADORA - TESE DE MESTRADO

**Candidata:** Lisiane Maria Bannwart Ambiel

**Data da Defesa:** 19 de junho de 2013

**Título da Tese:** "Redes Orientadas a Conteúdo: Uma Abordagem no Nível de Enlace"

Prof. Dr. Maurício Ferreira Magalhães (Presidente): _____

Prof. Dr. Fábio Luciano Verdi: _____

Prof. Dr. Marco Aurélio Amaral Henriques: _____

A meu marido, Antonio César
e a meus filhos, Maria Letícia
e Leandro.

# Agradecimentos

Eu não tenho dúvidas de que a capacidade do ser humano de aprender e de se relacionar vem de Deus, caso contrário não seria tão prazeroso. Muito obrigada, meu Deus, por tudo o que eu aprendi e por todas as pessoas que conviveram comigo nestes anos do mestrado!

Ao meu orientador Prof. Dr. Maurício Ferreira Magalhães e coorientador Prof. Dr. Christian R. Esteve Rothenberg, agradeço pela orientação e empenho que tornaram possível este trabalho.

Aos colegas do grupo de estudos, Walter e Marcos que estiveram comigo no início dos trabalhos e Cabral que acompanhou a finalização, agradeço pelo apoio.

Aos colegas da Unicamp, em especial aos amigos do Laboratório de Computação e Automação (LCA lado A e lado B), agradeço pela convivência, almoços e cafés. O ambiente de amizade, respeito e colaboração foi muito importante e vai deixar saudades.

Ao meu marido, Antonio César, agradeço pelo amor e cumplicidade que sempre nos uniu e aos meus filhos, Maria Letícia e Leandro, agradeço pelo carinho, compreensão e apoio à mamãe-estudante. A conquista é nossa!

Aos meus familiares e amigos que sempre me motivaram, incentivaram e rezaram para que eu conseguisse completar o mestrado, mesmo sem entender o que eu estava fazendo. Amo todos vocês!

À Unicamp e professores, pelo ensino de qualidade e à CAPES pelo apoio financeiro.

*O desígnio de Deus permanece para sempre, os projetos do seu coração, de geração em geração.*

*Salmo 33,11*

# Resumo

As Redes Orientadas a Conteúdo (ROC) se apresentam como uma nova forma de pensar a Internet: mudam o paradigma de comunicação apresentando uma nova abordagem com base no conteúdo independente de sua localização. Esta dissertação propõe uma arquitetura de rede orientada a conteúdos no nível de enlace sem uso de qualquer esquema de endereçamento. Os *Content Routers* (CR) são a base desta arquitetura, responsáveis pelo armazenamento de dados e roteamento de pacotes diretamente no conteúdo. Diferente do ambiente IP, onde existe o conhecimento do endereço do provedor de conteúdo, a arquitetura proposta no nível de enlace requisita conteúdos através da inundação de mensagens de forma controlada. Um protótipo é desenvolvido para validação da arquitetura e é utilizado em alguns cenários comparando duas abordagens: IP/*overlay* e nível de enlace. Alguns cenários de uso da arquitetura de CRs em redes domiciliares também são avaliados. Os resultados sugerem que arquiteturas orientadas a conteúdo e sem uso do IP podem ser viáveis e interessantes para redes de menor escala, que se beneficiariam de uma arquitetura simples, sem necessidade de configuração e gerenciamento como ocorre na arquitetura TCP/IP.

Palavras-chave: Redes Orientadas a Conteúdo, Armazenamento, Roteamento.

# Abstract

Content Oriented Networking is a new way to think about networking by changing the communication paradigm to an approach where content becomes the basis in replace of network location identifiers. This thesis proposes a content oriented network architecture at the link layer without the use of network addressing schemes. Content Routers (CR) are the basis for this architecture and are in charge of packet caching and routing directly on content names. Different from IP environments, where the destination address of the content source is known, the proposed link-level architecture requests contents by controlled message flooding. The work includes a prototype implementation which is used in some scenarios comparing two approaches: IP/overlay and link layer. Scenarios using CR architecture in home networks are also evaluated. Results suggest that content oriented IP-less architecture may be interesting for small networks such as home networks that would benefit from the simplicity of such architecture, without configuration and management as required when using TCP/IP.

Key-words: Content-Oriented Networking, Caching, Routing.

# Lista de Figuras

# Lista de Tabelas

# Lista de Acrônimos

| | |
|---|---|
| API | Application Programming Interface |
| AVP | Attribute-Value Pair |
| CCN | Content-Centric Networking |
| CDN | Content Delivery Network or Content Distribution Network |
| CONET | Content Inter Network |
| CR | Content Router |
| CS | Content Store (CCN) |
| CSS | CONET Sub System (CONET) |
| DHT | Distributed Hash Tables |
| DNS | Domain Name System |
| DONA | Data-Oriented Network Architecture |
| FIB | Forward Information Base |
| ICN | Information Centric Networking |
| IO | Information Object (NetInf) |
| IP | Internet Protocol |
| ISP | Internet Service Provider |
| ITU-T | International Telecommunication Union - Standardization Sector |
| NetInf | Network of Information |
| NDN | Named Data Networking |
| NRS | Name Resolution Service (NetInf) |
| OSI | Open Systems Interconnect |
| OSPF | Open Shortest Path First |
| OSPFN | Open Shortest Path First for Named-data |
| PIT | Pending Interest Table (CCN) |
| PSIRP | Publish-Subscribe Internet Routing Paradigm |
| PURSUIT | Publish-Subscribe Internet Technology |
| P2P | Peer-to-Peer |
| RH | Resolution Handle (DONA) |
| ROC | Redes Orientadas a Conteúdo |
| RFC | Request for Comments |
| SAIL | Scalable and Adaptative Internet Solutions |
| TCP | Transmission Control Protocol |
| URL | Universal Resource Locator |
| WWW | World Wide Web |

# Sumário

Capítulo 1

# Introdução

Este capítulo apresenta as motivações para a pesquisa e projetos na área das Redes Orientadas a Conteúdo (ROC) como forma de atender aos novos requisitos da Internet. São apresentados os objetivos do trabalho e a organização do texto.

## 1.1 Motivações e Objetivos

A Internet tem sido utilizada amplamente e com sucesso nas últimas três décadas tendo a simplicidade do seu modelo como uma das razões do crescimento acelerado, visto que não é necessário realizar modificações no núcleo da rede para a criação de novas aplicações. No entanto, este crescimento e popularidade impedem alterações que se fazem necessárias e restringem a adoção de novas tecnologias, tornando difícil resolver problemas estruturais como: escalabilidade, gerenciamento, mobilidade, segurança, qualidade de serviço, entre outros. Um grande número de projetos de pesquisa estuda os problemas da Internet e como a arquitetura deve ser melhorada. Alguns deles seguem uma abordagem evolutiva, onde a rede seria melhorada de forma incremental. Outros projetos seguem uma abordagem *clean slate* (Roberts 2009), sem compromisso de compatibilidade com a Internet atual, estabelecendo princípios e tomando decisões de projeto a partir do zero, ignorando restrições de projeto existentes hoje.

As Redes Orientadas a Conteúdo (ROC) se apresentam como uma nova forma de pensar a Internet: mudam o paradigma de comunicação apresentando uma nova abordagem com base no conteúdo independente de sua localização.

A Internet foi projetada para interconectar computadores para aplicações como troca de correspondências e transferência de arquivos. Hoje é usada principalmente para disseminação e recuperação de partes de conteúdos nomeados, principalmente vídeos (ao vivo ou completamente transferidos), mas também música e imagens, além dos arquivos gerados pelos próprios usuários e encaminhados para serviços de armazenamento em nuvem (SugarSync, Dropbox, UbuntuOne, AmazonCloudDrive) ou para publicação (Twitter, Facebook) na Internet. Os consumidores de conteúdos, em geral, não estão interessados em saber de onde vem a informação e somente precisam de uma garantia sobre a autenticidade e integridade da informação (Rothenberg, Verdi & Magalhães 2008).

A mudança no uso e os esforços para acomodar os *déficits* da arquitetura da rede atual

justificam as Redes Orientadas a Conteúdo (ROC) como uma linha de pesquisa. Esta abordagem torna a arquitetura da rede mais adequada para a distribuição de conteúdo e se utiliza de novos conceitos como conteúdo nomeado, roteamento baseado em nomes, segurança aplicada diretamente a conteúdos e armazenamento de dados nos nós intermediários da rede (*in-network caching*).

Entre as arquiteturas propostas e mais referenciadas estão: DONA (Koponen, Chawla, Chun, Ermolinskiy, Kim, Shenker & Stoica 2007), CCN (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009), NetInf (Ahlgren, D'Ambrosio, Marchisio, Marsh, Dannewitz, Ohlman, Pentikousis, Strandberg, Rembarz & Vercellone 2008) e PSIRP (Tarkoma, Ain & Visala 2009). Em comum, estas arquiteturas utilizam nomes de conteúdo únicos e persistentes, armazenamento de conteúdo na rede, modelo de segurança que permite verificação da integridade e origem do dado independente da fonte, simplicidade para suportar situações de mobilidade e múltiplos provedores (*multihoming*) e tolerância a falhas. Porém, ainda existem desafios para que estas redes sejam utilizadas em grande escala devido ao fato do número de conteúdos ser muito maior do que o número de endereços na rede atual, o que causa impacto no sistema de roteamento e resolução de nomes (Ahlgren, Dannewitz, Imbrenda, Kutscher & Ohlman 2012). Estudos indicam uma mudança no espaço de endereçamento de um bilhão de endereços IP para um trilhão de nomes de conteúdos(Perino & Varvello 2011). Algumas questões são colocadas para pesquisa com relação à manutenção de estado por pacote, custos de processamento e de armazenamento no roteador, redução do tamanho de tabelas armazenadas de forma eficiente nos roteadores e como fazer buscas e operações em alta velocidade. A abordagem mais utilizada na proposta e validação de novas arquiteturas de rede baseia-se na introdução de uma camada acima da infraestrutura IP (*overlay over IP*).

Diferentemente da abordagem *Overlay* normalmente utilizada, este trabalho propõe uma rede orientada a conteúdos no nível da camada de enlace utilizando a arquitetura de *Content Routers* (CRs) sem uso de endereçamento de rede com o objetivo de tornar o conteúdo o elemento mais importante na rede. A arquitetura dos CRs foi inicialmente definida como uma rede *overlay* sobre IP com suporte ao *caching* e utilização de nomes planos entre outras características (Wong, Giraldi, Magalhaes & Kangasharju 2011). Diferente do ambiente IP onde existe o conhecimento do endereço do provedor de conteúdo (servidor), a proposta de arquitetura no nível de enlace apresentada nesse trabalho parte do princípio de que a localização do conteúdo não é conhecida e a requisição é feita através de mecanismo de inundação de mensagens pelas interfaces disponíveis nos CRs (Ambiel, Rothenberg & Magalhaes 2013b). Ao propor tal condição surgem questões relacionadas à quantidade de mensagens geradas na rede. Assim, para reduzir o número de mensagens, além do roteamento oportunístico e do mecanismo de busca na vizinhança, este trabalho introduz: (a) lista de requisições pendentes por interface; (b) conceito de Super CR como elemento especializado em *caching*; (c) primitivas para divulgação de conteúdos e do Super CR; (d) indicação da menor distância em número de *hops*.

O trabalho faz a verificação experimental do comportamento da arquitetura de CRs operando no nível IP/*overlay* e no nível de enlace utilizando um protótipo. As duas abordagens são avaliadas para diferentes condições de *caching* e mecanismo de busca em dois diferentes cenários: topologia em malha, tipo Internet e rede domiciliar (*home network*).

A proposta no nível de enlace apresenta-se como uma alternativa interessante para redes menores tais como as redes domiciliares onde os custos relativos às dependências do IP (ex: gerência de rede, configuração de protocolos de roteamento e interfaces, segurança) seriam reduzidos com uma migração da arquitetura nativa da rede para um serviço de busca e entrega de conteúdo (Ambiel, Rothenberg & Magalhaes 2013a).

## 1.2 Organização da Dissertação

Além desta introdução, o texto encontra-se estruturado da seguinte forma: o Capítulo 2 apresenta as tecnologias e trabalhos relacionados com a dissertação; o Capítulo 3 detalha as informações relacionadas à arquitetura dos CRs e a proposta no nível de enlace; o Capítulo 4 descreve os principais pontos relacionados à implementação do protótipo, além de apresentar a validação e a avaliação em alguns cenários. Finalmente, o Capítulo 5 apresenta as conclusões e discute trabalhos futuros.

Capítulo 2

# Tecnologias e Trabalhos Relacionados

Os sistemas de endereçamento e roteamento da Internet apresentam problemas de escalabilidade reconhecidos pela comunidade (Meyer, Zhang & Fall 2007). Os desafios maiores estão em atender requisitos de mobilidade, *multi-homing* e engenharia de tráfego entre domínios. Um grande número de projetos de pesquisa estuda os problemas da Internet e como a arquitetura pode ser melhorada (Roberts 2009).

Este capítulo apresenta uma visão de como a Internet evoluiu em direção às Redes Orientadas a Conteúdo (ROC), as definições, conceitos e características fundamentais das ROCs. Além disso, faz uma revisão bibliográfica das arquiteturas e trabalhos relacionados às ROCs.

## 2.1 A Evolução da Internet em direção às ROCs

No seu início a Internet foi utilizada para compartilhar recursos nos computadores e os usuários precisavam saber onde procurar para encontrar o conteúdo que eles necessitavam. O sistema *World Wide Web* (WWW) simplificou esta tarefa. Esta ferramenta da Internet interliga informações acessíveis através de uma rede de computadores. Essas informações são representadas na forma de páginas Web, ou objetos Web, contendo textos, gráficos, áudios e vídeos. Os objetos Web são disponibilizados em máquinas conhecidas como servidores Web e recebem requisições de aplicações conhecidas como clientes Web.

Entre as aplicações Web se destacam quatro tipos que evoluíram no tempo causando impacto significativo na evolução da tecnologia: (a) recuperação de conteúdo estático, ou seja, conteúdos com pouca alteração no tempo; (b) recuperação de conteúdo dinâmico, ou conteúdos cuja forma final não está armazenada, mas são gerados no momento em que são requisitados através da execução de um programa específico no servidor Web; (c) recuperação de conteúdo contínuo (*streaming*), como áudios e vídeos que têm restrições de tempo para execução; e (d) interatividade, que são as aplicações onde dois ou mais usuários interagem em tempo real (ex. vídeo conferência, jogos em rede, mensagens instantâneas).

Estas diferentes aplicações demonstram a nova forma de uso pelos usuários da Internet. Tais aplicações exigem qualidade de serviço diferenciada demandando eficiência e confiabilidade na entrega dos conteúdos. Desenvolvimentos envolvendo tipos de conteúdo mais complexos como multimídia, aplicações interativas e conteúdo dinâmico expõem fragilidades do modelo

tradicional e levam a indústria a voltar-se para tecnologias referenciadas como redes de conteúdo (Hofmann & Beaumont 2005).

### 2.1.1 Redes de Conteúdo

As redes de conteúdo podem ser vistas como uma camada de rede virtual no topo de uma infraestrutura de rede já existente, uma forma eficiente para adicionar funcionalidade, segurança e flexibilidade às redes IP. Na evolução das redes de conteúdo, além do aumento da capacidade de processamento, pode-se observar mudanças: (i) no foco do servidor para o usuário e (ii) na capacidade de adicionar melhorias de forma incremental.

As quatro fases desta evolução podem ser resumidas em:

1. Melhoria no servidor com a adição de processadores, memória e espaço em disco. Como cresceu a busca por conteúdos, também cresceu a demanda no servidor de conteúdo que se tornou um gargalo na arquitetura. Assim, o primeiro momento desta evolução tem como foco melhorar o lado do servidor. Esta abordagem, além de manter a arquitetura centralizada, não se mostra flexível. Algumas melhorias podem ser feitas de maneira incremental conforme cresce a demanda, mas em algum momento será necessário trocar o servidor completamente.

2. Uso de *Caching Proxies* com objetivo de distribuição do conteúdo, movendo-o para mais próximo do usuário e reduzindo a carga no servidor de origem e o congestionamento na rede. As requisições de usuários são direcionadas e atendidas por estes dispositivos a partir de configuração específica feita nos *browsers* dos usuários. Um *proxy cache*, preferencialmente próximo ao cliente, armazena cópias de conteúdos conforme passam por ele. Dessa forma as requisições subsequentes podem ser resolvidas pelo mesmo *proxy cache* com menor tempo de resposta e economizando recursos significativos: banda e processamento.

3. Redes do tipo *Server Farm*, onde um grupo de servidores oferece o mesmo serviço. Este tipo de rede utiliza roteadores inteligentes que examinam as requisições e encaminham para um elemento do grupo de servidores, criando a impressão de que o grupo de servidores é um único servidor. Esta abordagem é mais flexível e atende melhor ao requisito de escalabilidade além de agregar o benefício inerente de tolerância a falhas.

4. Redes de Distribuição de Conteúdo ou *Content Distribution Network* ou ainda *Content Delivery Network* (CDN), um sistema de computadores interconectados através da Internet cooperando de forma transparente para entrega de conteúdo aos usuários (Hofmann & Beaumont 2005), (Day, Cain, Tomlinson & Rzewski 2003) e (Pathan, Buyya & Vakali 2008).

Embora *Caching Proxies* e *Server Farms* sejam técnicas úteis, elas têm limitações. Os *Server Farms* podem melhorar a escalabilidade com um grupo de servidores, porém como estes múltiplos servidores e demais elementos são tipicamente disponibilizados próximos uns aos outros, eles pouco colaboram na melhoria de problemas de desempenho relativos ao congestionamento

na rede. Os *Caching Proxies* podem melhorar problemas de desempenho devido a congestionamento na rede pois estão situados próximo ao usuário, porém eles armazenam conteúdos com base na demanda do usuário, que precisa ser configurado adequadamente (o *browser* do usuário deve ser configurado com o endereço do *proxy cache*). As CDN procuram contornar estas limitações.

### 2.1.2 Redes de Distribuição de Conteúdos

Uma Rede de Distribuição de Conteúdos, ou *Content Delivery Networks*(CDN), consiste em uma rede de servidores espalhados por vários domínios na Internet que oferece serviços de entrega de conteúdos definidos por um contrato de serviço (Vakali & Pallis 2003). Os servidores da rede operam de forma orquestrada com gerenciamento centralizado para otimizar a distribuição de réplicas e a utilização de recursos. As requisições dos conteúdos são redirecionadas pelo servidor de nomes, o DNS da CDN, que irá verificar a origem da requisição e selecionar o servidor mais próximo e disponível que possa atender a requisição. Assim, uma requisição de um cliente para um determinado conteúdo é direcionada para uma réplica que vai responder muito mais rapidamente do que se a requisição fosse encaminhada para o servidor de origem, com a devida integridade e consistência (Day et al. 2003). Conforme mais clientes solicitam pelo mesmo conteúdo, o serviço de DNS vai redirecionando as requisições para outros servidores, com a finalidade de garantir o balanceamento de carga. Neste cenário, o sistema da CDN passa a distribuir os conteúdos para os servidores no mesmo domínio, para que estes realizem o *caching* do conteúdo. Desta forma, futuras requisições passam a ser atendidas por esses servidores sem a necessidade da transferência dos dados do servidor original. Uma CDN tipicamente incorpora informação dinâmica sobre condições da rede e a carga dos servidores, direcionando as requisições de forma a balancear a carga, não utilizando para decisão somente informações estáticas sobre localização geográfica e conectividade da rede. O valor de uma CDN para um provedor de conteúdo é uma combinação de infraestrutura de distribuição e melhoria na entrega de conteúdo.

### 2.1.3 Redes Orientadas a Conteúdos

Mais recentemente, Jacobson et al. (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009) introduziram um modelo de arquitetura com base no nome do conteúdo: *Content Centric Networks*, CCN. Outros autores reforçam a ideia de uma nova geração de redes cujo foco principal é a informação ou conteúdo (Ahlgren et al. 2008).

A Internet tem sido usada prioritariamente para distribuição de conteúdo, porém foi projetada para conversação entre usuários finais. Esta natureza está incorporada no formato do pacote que nomeia estes dois pontos: endereços IP origem e destino. A proposta das Redes Orientadas a Conteúdo (ROCs) é justamente remover esta característica e tornar o conteúdo mais importante do que a sua localização. Assim, o conteúdo passa a ser cidadão de primeira classe nesta nova arquitetura. A arquitetura também prevê que cada pedaço de conteúdo (conhecido como *chunk*) possa ser nomeado de forma única e não impõe um único caminho para roteamento, mas cada nó é livre para usar toda sua conectividade para solicitar ou distribuir con-

teúdos (Zhang, Estrin, Burke, Jacobson, Thornton, Smetters, Zhang, Tsudik, Claffy, Krioukov, Massey, Papadopoulos, Abdelzaher, Wang, Crowley & Yeh 2010).

Juntamente com as ROCs, a computação em nuvem e a interconexão de redes heterogêneas aparecem como abordagens a serem consideradas ao se pensar em uma nova arquitetura para a Internet. Tais abordagens devem dar suporte adequado para aplicações e tratar os desafios da Internet de forma integrada: armazenamento, processamento e transporte. Nas ROCs, o conteúdo é independente do dispositivo de armazenamento e da aplicação associada o que permite o seu armazenamento de forma eficiente na rede (*in-network caching*). Ao mesmo tempo, as aplicações em rede exigem provisionamento e gerenciamento de servidores e repositórios, justificando o investimento na área de computação em nuvem com uso de virtualização. Também o transporte da informação através de redes heterogêneas, utilizando diferentes tecnologias, traz dificuldades diversas aos modelos de negócios e acordos de serviço, exigindo maior abertura no campo da conectividade (Ahlgren, Aranda, Chemouil, Oueslati, Correia, Karl, Söllner & Welin 2011).

## 2.2 Conceitos e Características das ROCs

Os seguintes conceitos são propostos no contexto de Redes Orientadas a Conteúdo (ROC) (Ahlgren et al. 2012, de Brito, Velloso & Moraes 2012):

- **Objetos de dados:** representam os diferentes conteúdos como, por exemplo, páginas Web, documentos, filmes, fotos, músicas, enfim, todos os tipos de objetos que podem ser armazenados e acessados via computadores. Um conteúdo mantém o seu nome independente da sua localização, método de armazenamento ou aplicação.

- **Nomeação de conteúdos (*naming*):** esquemas de nomeação que permitem identificar o conteúdo e servem de base para a sua distribuição na rede, requisição e processo de autenticação. Esquemas básicos: nomeação plana, nomeação hierárquica e nomeação por atributo.

- **Roteamento baseado em nomes (*name-based routing*):** a rede entrega os conteúdos requisitados por nome sem qualquer informação referente à localização, tanto de usuário quanto de armazenamento de conteúdos.

- **Armazenamento de conteúdos (*caching*):** qualquer nó da rede pode atuar como um *cache* independente da aplicação ou protocolo. Isso inclui os nós da rede (infraestrutura) e os nós dos usuários como computadores, dispositivos domésticos e terminais móveis. O *caching* é aplicável a qualquer conteúdo. Assim, uma requisição por determinado conteúdo pode ser respondida por qualquer nó que tenha a cópia em seu *cache*.

### 2.2.1 Nomeação de Conteúdos

Os mecanismos para nomeação podem ser classificados em três classes básicas: nomes planos, hierárquicos e baseados em atributos (Choi, Han, Cho, Kwon & Choi 2011). Cada uma das

classes de nomes atende parcialmente os requisitos indicados para os mecanismos de nomeação: persistência, escalabilidade e inteligibilidade ao usuário final (Ghodsi, Koponen, Rajahalme, Sarolahti & Shenker 2011). A nomeação hierárquica é legível e escalável, mas possui restrições ao uso persistente dos nomes. Esta restrição não existe no caso de nomes planos que apresentam vantagem com relação à segurança, mas seu uso depende de um processo de resolução entre um nome inteligível e o nome plano. O uso de atributos relacionados ao conteúdo pode facilitar a busca, mas depende de definições adequadas para evitar ambiguidades. Existe, portanto, um ponto de compromisso a ser considerado entre o esquema de nomeação e os esquemas de roteamento e segurança. Os esquemas de nomeação podem afetar o roteamento e o modelo de segurança como indicado na classificação a seguir:

- **Esquema de nomeação hierárquica**: utiliza estrutura similar às URLs (*Universal Resource Locator*) atuais. Em geral, trata-se de uma sequência de caracteres legíveis que facilita a escalabilidade do sistema de roteamento por meio da agregação de prefixos, feita de forma similar à agregação de rotas nos protocolos IP. Em contrapartida, mudanças na hierarquia, como transferência do provedor ou proprietário do conteúdo, refletem diretamente no nome do mesmo porque os nomes hierárquicos refletem a informação de propriedade dos conteúdos de forma explícita. A relação com o sistema de segurança requer mecanismo adicional para mapeamento do nome a uma chave de segurança de modo que o receptor (elemento de rede ou o requisitante do conteúdo) possa proceder à verificação do conteúdo recebido.

- **Esquema de nomeação plana**: utiliza nomes planos e auto-certificáveis pela definição de um identificador de conteúdo como sendo, por exemplo, um *hash* criptográfico de uma chave pública associada ao conteúdo. O nome plano (ou identificador) pode garantir persistência e unicidade com o custo de não ser legível, o que requer um mecanismo adicional que faz o mapeamento entre o identificador e o nome legível do conteúdo (nome amigável, ou seja, como aquele conteúdo é conhecido no mundo real).

  A auto-certificação de conteúdos e identificadores utiliza mecanismo de *hash* criptográfico. A arquitetura DONA, por exemplo, utiliza o formato P:L (*Principal:Label*), onde P representa o *hash* criptográfico da chave pública do originador do conteúdo e L representa um rótulo escolhido pelo originador. Os usuários ou consumidores de conteúdos possuem meios para verificar a validade da chave utilizada na codificação do conteúdo pela aplicação de função de *hash* criptográfico (Ghodsi, Koponen, Rajahalme, Sarolahti & Shenker 2011). Uma abordagem diferente daquela adotada pela arquitetura DONA é a da arquitetura NetInf, onde o nome não possui vínculo com o proprietário, ou responsável pelo conteúdo. NetInf propõe duas abordagens onde a informação de propriedade está desacoplada do nome, ou identificador de conteúdo.

  Uma das críticas aos sistemas de nomeação utilizando nomes planos está na ausência de hierarquia, o que não permite agregação direta e que pode causar problemas de escalabilidade pois será necessário uma entrada na tabela de roteamento para cada conteúdo. Entretanto, alguns autores como (Ghodsi, Koponen, Rajahalme, Sarolahti & Shenker 2011) alegam que os nomes planos auto certificáveis, além de permitir verificação direta do conteúdo,

podem sim permitir formas flexíveis de agregação, contribuindo para uma melhor escalabilidade destes sistemas. A arquitetura NetInf sugere o uso deste método para agregação dos identificadores (Dannewitz, Kutscher, Ohlman, Farrell, Ahlgren & Karl 2013).

- **Esquema de nomeação por atributos**: utiliza pares atributo-valor ou *attribute-value pair* (AVP) para identificar o conteúdo. Um usuário especifica seu interesse por um conteúdo com um conjunto de AVPs e um elemento da rede pode localizar conteúdos elegíveis por comparação. Este tipo de nomeação pode facilitar a busca na rede, porém, apresenta algumas desvantagens pois depende de um sistema semântico bem definido para evitar ambiguidade e, além disso, o número de possíveis AVPs pode ser bem grande.

### 2.2.2 Roteamento Baseado em Nomes

Os mecanismos de roteamento com base em conteúdos (ou nomes de objetos) podem ser classificados em dois grupos em função da existência de uma estrutura sistemática para manutenção das tabelas de roteamento (Choi et al. 2011):

- **Roteamento não-estruturado**: assume a inexistência de estruturas para manutenção das tabelas de roteamento, sendo que a publicação dos conteúdos ocorre, em geral, por difusão (*flooding*).
- **Roteamento estruturado**: pode utilizar árvores hierárquicas ou tabelas *hash* distribuídas (*Distributed Hash Tables* - DHT).

Uma outra forma de classificar os mecanismos de roteamento toma por base as propriedades do espaço de nomes de objetos, em particular se os nomes são agregáveis ou não (Ahlgren et al. 2012) e apresenta duas abordagens:

- **Roteamento com uso de serviço de resolução de nomes ou *Name Resolution Service* (NRS)**: nesta abordagem o NRS faz o mapeamento do nome de um objeto para um localizador que aponta para o local de armazenamento na rede (servidor ou fonte do conteúdo). O processo consta de três etapas:

  1. Encaminhamento da mensagem de requisição para o nó responsável pelo NRS onde o nome do objeto é traduzido para endereço(s) de um ou mais servidores de conteúdo.
  2. Encaminhamento da mensagem de requisição para o endereço do servidor.
  3. Encaminhamento do dado (ou mensagem de resposta) do servidor para o requisitante

  Todas as fases podem potencialmente utilizar diferentes algoritmos de roteamento. Um método de roteamento com base em nomes pode ser utilizado principalmente na $1^a$ fase (ex. DHT). A $2^a$ e $3^a$ fases podem usar roteamento com base na topologia como o *Internet Protocol* (IP).

- **Roteamento direto no nome**: nesta abordagem a mensagem de requisição é encaminhada diretamente do requisitante para uma ou mais fontes de dados na rede com base no nome do objeto. O algoritmo de roteamento usado nesta abordagem depende fortemente das propriedades do espaço de nomes. Após a fonte ter recebido a mensagem de requisição, o dado é roteado de volta ao requisitante.

Além dessas classificações apresentadas, pode-se diferenciar entre as fases de roteamento:

1. Roteamento do registro do conteúdo, ou da mensagem que faz publicação do conteúdo.
2. Roteamento da requisição dos objetos de dados.
3. Roteamento dos objetos de dados para o requisitante.

O roteamento com base no conteúdo é uma das vantagens da ROC com relação às CDNs (ver seção 2.1.2). Embora as CDNs ofereçam um mecanismo de roteamento de conteúdos, elas não usam os recursos da rede porque operam em nível de aplicação. As ROCs permitirão que os provedores executem um roteamento por conteúdo nativo, ou seja, será uma facilidade incorporada à rede, diferente das CDNs(Detti, Pomposini, Blefari-Melazzi & Salsano 2012).

Alguns autores apontam como desafio a definição de um protocolo de roteamento escalável, que seja capaz de endereçar todas as cópias disponíveis. Comparam o encaminhamento determinístico e o encaminhamento exploratório em ambiente intra-domínio. O primeiro é direcionado a uma cópia conhecida pela FIB (*Forward Information Base*) e o segundo é direcionado a uma cópia cuja localização não é conhecida e faz uso de mecanismo de inundação (*flooding*). Os autores mostram que é possível o uso de cada um destes métodos além de uma estratégia híbrida (Chiocchetti, Rossi, Rossini, Carofiglio & Perino 2012).

### 2.2.3 Armazenamento de Conteúdos (*Caching*)

A capacidade de armazenamento ou *caching* é parte integral das ROCs, comum a todas as propostas de projetos. É objetivo das ROCs o acesso eficiente aos conteúdos mais populares aliviando efeitos como *flash crowd* causados pelo acesso simultâneo de milhares de usuários a conteúdos populares e que pode acontecer de forma inesperada. Em geral, todos os nós da rede têm potencial para armazenamento, incluindo os nós que operam na infraestrutura da rede e os nós de usuários que operam em rede particulares ou domésticas, assim como os terminais móveis. Nas ROCs, a requisição por objetos de dados pode ser atendida por qualquer um destes nós contendo uma cópia do conteúdo em seu *cache*. Assim, as ROCs combinam armazenamento na borda da rede com armazenamento na rede (*in-network caching*). O armazenamento ocorre de forma genérica, ou seja, é independente da aplicação e se aplica ao conteúdo de qualquer provedor, incluindo os gerados pelos usuários (Ahlgren et al. 2012).

O *caching* pode ajudar a reduzir o custo de transporte para os provedores da rede e melhorar o tempo de entrega para os usuários finais. A disponibilidade de diversas cópias depende de fatores como popularidade do conteúdo e estratégia de substituição, entre outros, e está condicionada ao modelo de encaminhamento das requisições.

A eficiência de *caching* cooperativo é questionada por alguns autores (Ghodsi, Shenker, Koponen, Singla, Raghavan & Wilcox 2011) e apontada como área de pesquisa a ser avaliada. Neste sentido, outros autores estudam algoritmos de *caching* em ambiente descentralizado e não coordenado, diferente das operações tradicionais, com objetivo de reduzir redundância e tornar mais eficiente o uso dos recursos de armazenamento em roteadores (Psaras, Chai & Pavlou 2012). O *caching* cooperativo é tratado como forma de reduzir a redundância deixando espaço para mais dados diferentes em (Wong, Wang & Kangasharju 2012), o qual (i) apresenta uma estratégia para decidir se um conteúdo deve ser ou não armazenado no roteador utilizando informação no cabeçalho da mensagem e (ii) implementa um protocolo para troca de informação sobre conteúdos armazenados. A avaliação do mecanismo de cooperação mostrou melhoria na taxa de *cache-hit* e redução na média de número de *hops* para atingir o conteúdo, o que indica menor tráfego.

### 2.2.4 Primitivas

Os projetos voltados às redes de conteúdo adotam um estilo comum para as primitivas (Ghodsi, Shenker, Koponen, Singla, Raghavan & Wilcox 2011). Em geral, as primitivas utilizam o nome do conteúdo e o paradigma *publish/subscribe* onde o provedor do conteúdo não conhece a localização do solicitante e vice-versa (desacoplamento no espaço). Da mesma forma, provedor e solicitante de conteúdo não precisam estar *online* simultaneamente (desacoplamento no tempo).

Quase todos os projetos de ROC utilizam primitivas básicas que lembram a noção de `PUBLISH` e `SUBSCRIBE`: a arquitetura DONA usa `REGISTER` e `FIND`; a arquitetura CCN usa `REGISTER` e `INTEREST`; e a arquitetura PSIRP usa explicitamente as primitivas `PUBLISH` e `SUBSCRIBE`.

Nas ROCs, as primitivas atuam sobre os nomes de conteúdos oferecendo operações que buscam e recuperam o conteúdo previamente publicado (*fetch*) e operações que se subscrevem a conteúdos que serão publicados e recuperados no futuro.

### 2.2.5 Segurança

O modelo de segurança orientado a conteúdo baseia-se em uma característica comum a essas redes. Essa característica deve-se ao fato de que nas ROCs os conteúdos podem ser obtidos a partir de outros elementos da rede e não somente do provedor original de conteúdo. Assim, nestes modelos o conteúdo é assinado pelo provedor original (aquele que publica o conteúdo na rede), de forma que os elementos da rede e os consumidores possam verificar a sua validade pela verificação da assinatura relacionada ao conteúdo (Ghodsi, Shenker, Koponen, Singla, Raghavan & Wilcox 2011).

Em geral, a segurança visa atender quatro aspectos (Ghodsi, Koponen, Rajahalme, Sarolahti & Shenker 2011):

- **disponibilidade**: o dado é encaminhado de forma confiável
- **proveniência**: o conteúdo foi gerado pelo provedor original
- **integridade**: o dado não foi corrompido

- **confidencialidade**: o dado não foi lido por outros

O mecanismo de nomeação tem papel importante na questão da segurança, em especial nos três últimos aspectos citados acima. A disponibilidade é uma responsabilidade da rede, ou seja, do mecanismo de encaminhamento e roteamento que também envolve o mecanismo de nomeação.

### 2.2.6 Hardware e Software dos Roteadores

As ROCs representam um desafio para a tecnologia atual dos roteadores para suportar maior velocidade no encaminhamento com base no nome do conteúdo e armazenamento de pacotes de dados (*caching*) (Perino & Varvello 2011). A mudança no espaço de endereçamento de um bilhão de endereços IP para cerca de um trilhão de nomes de conteúdos requer um evidente aumento de recursos para manter o estado do roteamento. O ponto positivo está no fato do *caching* local permitir que os roteadores respondam às solicitações aliviando potencialmente a frequência das operações de encaminhamento. Além disso, o uso de estruturas de dados probabilísticas como os filtros de Bloom (Tarkoma, Rothenberg & Lagerspetz 2012) aparecem como abordagem promissora para implementar métodos necessários para encaminhamento de pacotes a um custo efetivo.

Com base em uma avaliação sistemática da adequação do software e hardware existentes em roteadores comerciais, o trabalho de (Perino & Varvello 2011) conclui que os roteadores atuais podem suportar apenas uma parte das informações (ou estado) necessárias para operar uma ROC na escala da Internet. No entanto, a redução do escopo, isto é, da escala Internet para escalas de Redes de Distribuição de Conteúdos (*Content Delivery Network* - CDN) ou no escopo de um provedor de serviços (*Internet Service Provider* - ISP), os roteadores atuais podem ser adaptados para se tornarem roteadores de conteúdo. As avaliações consideraram um modelo genérico com componentes FIB, PIT e CS como no projeto NDN, e nomes hierárquicos. Os resultados das avaliações indicam que uma regulação no tamanho do nome do conteúdo também seria benéfica para as ROC (Perino & Varvello 2011).

### 2.2.7 Desafios das ROCs e Direção da Pesquisa

As ROCs apresentam pontos positivos dentre os quais podemos destacar (Detti et al. 2012), (Ahlgren et al. 2012): (a) controle da comunicação feito pelo receptor; (b) persistência e unicidade do nome do conteúdo; (c) roteamento com base em nomes mais eficientes que os mecanismos utilizados hoje pelas CDNs, pois faz uso dos recursos da rede e do conhecimento da topologia ; (d) armazenamento na rede; (e) qualidade de serviço diferenciada por conteúdo, que pode ser oferecida com base em transmissão ou armazenamento do mesmo; (f) simplificação no tratamento de mobilidade, pois a mudança de um ponto de acesso para outro (*handover*) não gera necessidade de manutenção do estado da comunicação; o próximo conteúdo ou *chunk* será solicitado normalmente, porém pode ser fornecido por outro nó da rede, diferente do que forneceu o conteúdo antes do *handover*, pois não existe vínculo com localização; (g) simplificação no tratamento de comunicação *multicast*, pois a solicitação de um conteúdo por múltiplos usuários será resolvida na rede, tirando proveito do armazenamento; (h) simplificação em situações de

múltiplos provedores (*multihoming*), porque não gerencia conexão fim a fim, podendo escolher entre enviar para um, vários ou todos provedores; e (i) tolerância a falhas.

Além da inevitável mudança na operação da rede, entre os principais desafios para que as redes de conteúdo possam ser utilizadas em grande escala podem ser destacados (Ahlgren et al. 2012): (a) escalabilidade devido ao fato do número de conteúdos ser muito maior do que o número de endereços na rede, o que impacta no sistema de roteamento e resolução de nomes; (b) privacidade, uma vez que as requisições por conteúdos são visíveis à rede resultando em situações piores que as atuais; além disso, pode não ser possível relacionar uma requisição a um cliente em particular; (c) questões legais relacionadas à preocupação dos proprietários de conteúdos com a disseminação do mesmo; (d) padronização (ITU-T Study Group 13 2011); (e) crescimento incremental.

Recentemente, o seminário de Dagstuhl (Ghodsi, Ohlmann, Ott, Solis & Wählisch 2013) discutiu algumas questões relacionadas às ROCs e apresentou pontos interessantes para os grupos de pesquisa. Entre outros:

- Casos de uso de ROC e implantação: deve ser considerado que os primeiros produtos a serem lançados em futuro próximo podem não ser dispositivos da rede, como o caso de roteadores, mas dispositivos do consumidor como telefones, computadores ou TV. Como possíveis campos de aplicação o documento cita, entre outros, Internet das Coisas e Comunicação Máquina-a-Máquina.

- O Projeto NDN tem focado seu esforço de pesquisa em projeto, desenvolvimento e uso real de uma variedade de aplicações executando sobre redes NDN. O ambiente de teste NDN (*testbed*) é usado não somente para facilitar os experimentos mas auxilia no direcionamento da pesquisa. Tal experiência sugere que a pesquisa deve incluir componentes experimentais para verificar se e quão bem o projeto proposto pode resolver problemas reais; esta situação força o detalhamento da arquitetura, valida as vantagens do NDN, e direciona o desenvolvimento da arquitetura com base em visão mais ampla das futuras aplicações.

## 2.3 Propostas de Arquiteturas para ROCs

Entre as propostas de arquiteturas para ROC se destacam: DONA, CCN, NetInf e PSIRP. Outras abordagens têm sido apresentadas na literatura sempre com o objetivo de melhorar o desempenho da rede atual.

### 2.3.1 Data-Oriented Network Architecture (DONA)

A arquitetura *Data-Oriented Network Architecture* (DONA) (Koponen et al. 2007) propõe a substituição da resolução de nomes via *Domain Name System* (DNS) por um serviço de nomes planos, auto certificáveis. É um trabalho na linha *clean slate* para nomeação e resolução de nomes. Um dos objetivos desta arquitetura é resolver problemas relacionados à persistência dos nomes, disponibilidade do conteúdo e autenticidade dos dados. Atualmente os nomes no DNS

são construídos de forma hierárquica representando os domínios administrativos. Sempre que um dado é transferido de um servidor para outro seu nome também é alterado, ou seja, não é persistente. Com relação à disponibilidade, o objetivo é recuperar o conteúdo, ou pedaços dele, da fonte disponível mais próxima. Como o conteúdo pode ser recuperado de qualquer fonte (servidores, parceiros ou *cache* na rede), é importante ter um mecanismo de segurança para autenticar esses dados.

As primitivas básicas do modelo de comunicação são `REGISTER` e `FIND`. O roteamento é feito de forma estruturada, seguindo um esquema hierárquico em árvore numa rede *overlay*. Esta rede é formada pelos *Resolution Handles* (RH), que são elementos que podem prover armazenamento (*caching*).

Na arquitetura DONA os nomes são organizados em torno de *Principals*, que correspondem aos proprietários dos dados. Os nomes estão associados a um par de chaves pública-privada. Cada dado, serviço ou entidade está associado a um *Principal*. Os nomes se apresentam na forma P:L, onde P é o *hash* criptográfico da chave pública do *Principal* e L é um rótulo (*Label*) escolhido pelo *Principal* garantindo a unicidade do nome. Os *Principals* são donos de seus dados e autorizam as estações a fornecer acesso aos mesmos. Cada dado é recebido como uma tripla <dado, chave pública, assinatura>. A granularidade dos dados é definida pelo *Principal* e pode ser um Web Site, uma página Web ou simplesmente uma foto.

Para resolução dos nomes DONA utiliza o paradigma *route-by-name* através dos RHs e das primitivas básicas: `FIND`(P:L) e `REGISTER`(P:L). Um cliente requisita um conteúdo através da primitiva `FIND`(P:L). O RH encaminha esta requisição na direção da cópia mais próxima com base na tabela de registros que mapeia um nome para o próximo salto (*hop*) e a distância da cópia em termos de número de *hops*. O encaminhamento da primitiva `FIND` é simples: se existe um registro na tabela, o `FIND` é encaminhado para o próximo *hop*, caso contrário, é encaminhado para o nó acima (pai). Cada cliente conhece a localização do seu RH local por meio de configuração. Qualquer estação com autorização de fornecer conteúdos ou serviços com nomes P:L envia uma primitiva `REGISTER`(P:L) ao seu RH local. Cada `REGISTER` deve ser autenticado. Ao receber um `FIND` ele troca o endereço IP de origem pelo seu próprio IP forçando a resposta passar por este RH, viabilizando, eventualmente, o armazenamento no *cache*. Quando o RH recebe um `FIND` e acontece um *cache hit* ele responde conforme o protocolo de transporte em uso.

A Figura 2.1 oferece uma visão geral de como os pacotes `FIND` são encaminhados aos RHs apropriados (passos de 1 a 4) e como o dado é enviado, seja pelo caminho reverso que possibilita o *caching* nos RHs (passos de 5 a 8), seja por uma rota direta (passo 9). Neste caso, o caminho a ser feito por uma requisição pode ser diferente do caminho a ser percorrido para a entrega dos dados utilizando o IP.

Como parte dos trabalhos, os autores da arquitetura DONA implementaram um protótipo de um RH com módulos de roteamento, armazenamento e aplicações como HTTP Proxy, *Session Initiation Protocol* (SIP) Proxy e P2P *client* para distribuição de um grande arquivo. A implementação ajudou a entender melhor certos detalhes da arquitetura e não trouxe nenhuma questão mais grave relacionada à viabilidade da proposta. Com relação à escalabilidade, apesar de não ter sido viável um experimento para avaliação deste item, apresentam estimativas para

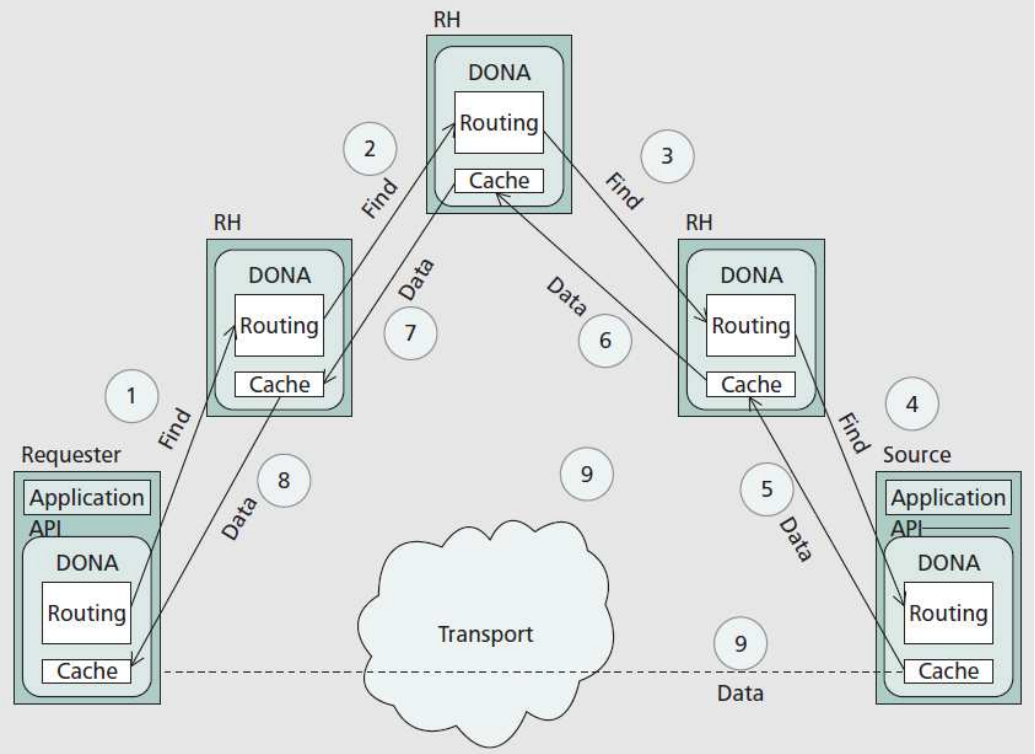



Figura 2.1: Arquitetura DONA (Ahlgren et al. 2012)

uso de memória ou disco nos RHs e discutem uma infraestrutura com hierarquia de RHs como forma de atender ao requisito de escalabilidade das redes orientadas a conteúdo.

### 2.3.2 Content Centric Networking (CCN) / Named Data Networking (NDN)

A arquitetura *Content Centric Networking* (CCN) proposta inicialmente por Jacobson et al. em 2007 (Jacobson, Mosko, Smetters & Garcia-Luna-Aceves 2007) e apresentada com detalhes em 2009 (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009) é base do projeto *Named Data Networking* (NDN) (Zhang et al. 2010).

A arquitetura tem todo seu foco no nome dos conteúdos e não na localização dos *hosts* onde estes se encontram. Enquanto os pacotes na rede IP contêm localização, os pacotes na rede CCN contêm nomes de dados assim como nos demais projetos voltados para as redes de conteúdo. Os nomes na CCN são estruturados de forma hierárquica e formados por um determinado número de componentes, similar aos nomes qualificados por domínios. A estrutura de nomes hierárquicos é utilizada para atender melhor à escalabilidade do roteamento pela agregação de prefixos. Os usuários podem fazer requisições por um prefixo de publicador. Assim, os nomes na arquitetura CCN são utilizados também com finalidade de roteamento.

O modelo utiliza três estruturas de dados principais conforme apresentado na Figura 2.2: uma tabela de encaminhamento, *Forward Information Base* (FIB), que mantém a lista de interfaces por onde os pacotes INTEREST podem ser encaminhados; uma tabela de interesse, *Pending Interest Table* (PIT), que armazena a interface de recebimento e o nome descrito no

pacote de `INTEREST`; e um repositório de dados, *Content Store* (CS).

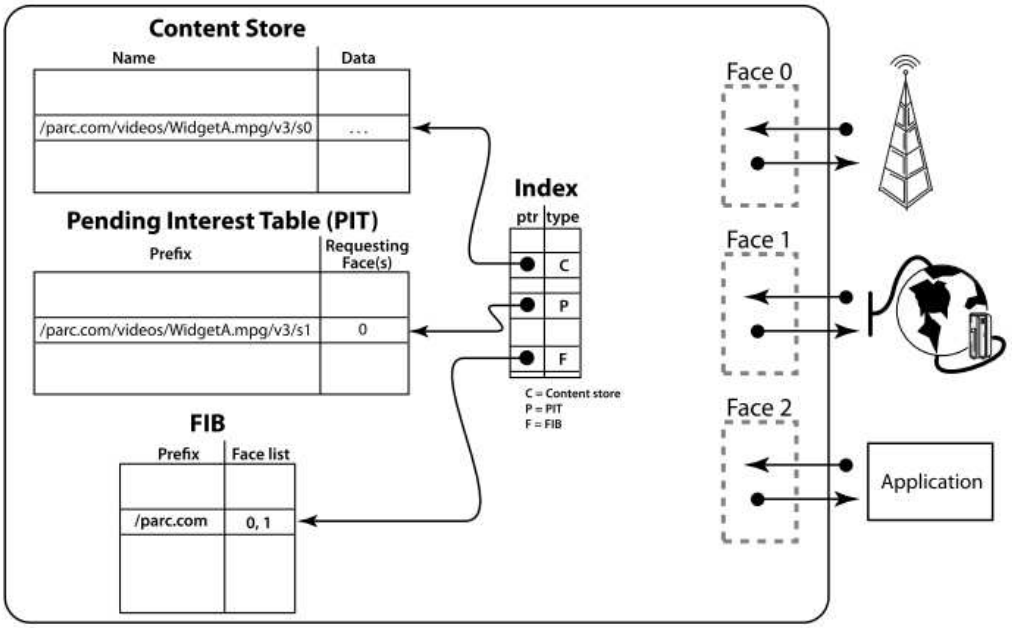



Figura 2.2: Arquitetura CCN (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009)

Todos os nós podem prover armazenamento (*caching*), sujeitos somente à disponibilidade de recursos e políticas. Qualquer nó com acesso a múltiplas redes pode servir como um roteador de conteúdo entre elas. Todo conteúdo é autenticado com assinatura digital e conteúdo particular é protegido com criptografia. A autenticação é uma ligação entre nome e conteúdo.

A arquitetura CCN utiliza esquema de roteamento com base no protocolo *Open Shortest Path First* (OSPF) aplicado aos dados nomeados (OSPF *for Named-data*, ou OSPFN), que faz o anúncio de prefixos de nomes. Cada roteador mantém sua FIB com base nos anúncios recebidos (Wang, Hoque, Yi, Alyyan & Zhang 2012).

Na arquitetura CCN existem dois tipos de pacotes: `INTEREST` e `DATA`. Um consumidor requisita um conteúdo disseminando seu pacote `INTEREST` (*flooding*). Qualquer roteador da rede CCN que satisfaça as condições responde com o pacote `DATA`, que é transmitido somente em resposta a um `INTEREST`. Caso contrário, o roteador insere uma referência para a interface de onde recebeu o `INTEREST` na PIT e encaminha pelas interfaces indicadas na FIB conforme decisão da camada de estratégia, camada que possibilita múltipla conectividade (cada entrada da FIB está associada a um programa especializado). O modelo CCN segue o padrão das ROCs onde a comunicação é orientada pelo consumidor de dados e a camada de estratégia do receptor/consumidor é responsável por requisitar conteúdos não entregues ou corrompidos.

O roteamento é feito de forma não estruturada: um consumidor pede por um conteúdo disseminando seu pacote `INTEREST` pelas conexões disponíveis. Estes pacotes propagam seu caminho em direção à fonte de dados e deixam um rastro para que o pacote `DATA` possa fazer o caminho de volta em direção a quem originou a requisição. Qualquer nó que escute e tenha os dados que satisfaçam a requisição pode responder a um `INTEREST` e consumir aquele pacote. Os pacotes de `INTEREST` e `DATA` têm uma relação um a um e mantêm um balanceamento de

fluxo similar aos pacotes TCP `DATA` e `ACK`. No modelo CCN somente os pacotes `INTEREST` são roteados. Estes pacotes utilizam um valor aleatório (ou *nonce*, número utilizado uma única vez) para evitar *loops*.

A Figura 2.3 ilustra como os pacotes `INTEREST` são encaminhados em direção a um provedor de conteúdos (passos 1 a 3) e como o pacote `DATA` é roteado pelo caminho reverso usando a informação mantida pela PIT (passos 4 a 6). Um roteador CCN pode fazer *caching* de um conteúdo recebido em resposta a uma requisição e responder a uma requisição subsequente para o mesmo objeto (passos 7 e 8).

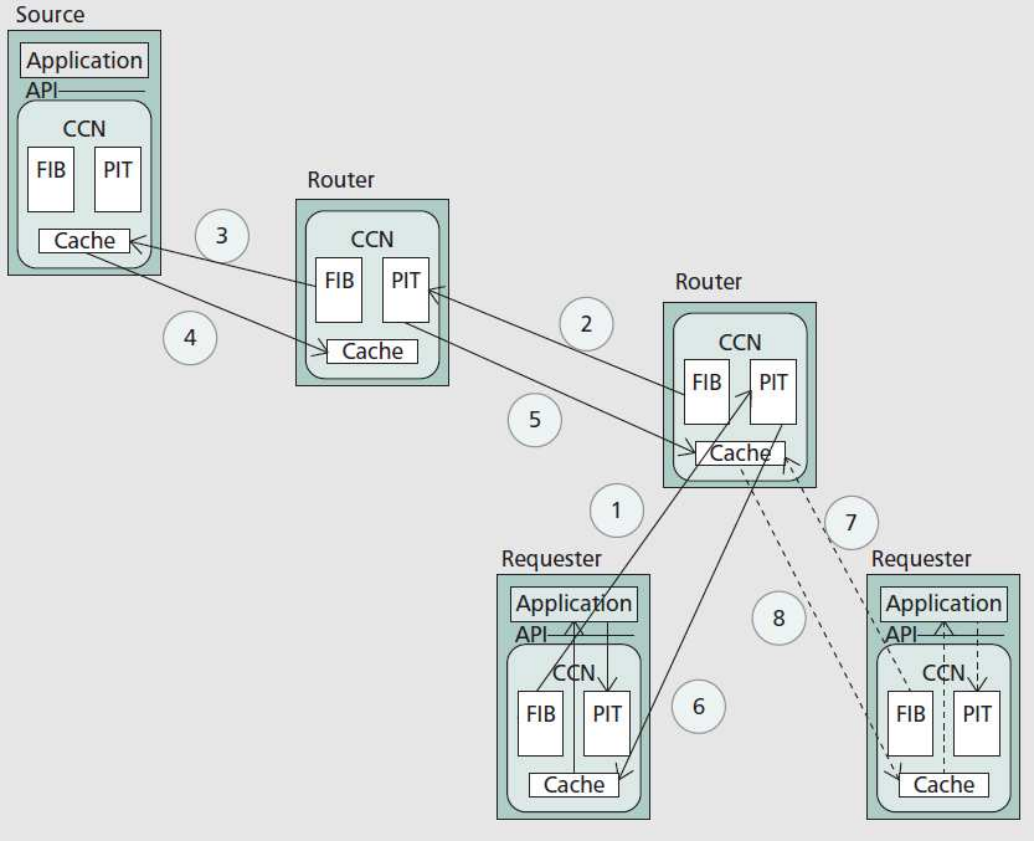



Figura 2.3: Arquitetura CCN (Ahlgren et al. 2012)

Entre as características do modelo de encaminhamento na arquitetura NDN destaca-se o fato de ser adaptativo (Yi, Afanasyev, Wang, Zhang & Zhang 2012). Na arquitetura IP, a entrega de um pacote acontece em duas fases: no plano de roteamento, os roteadores trocam informações e atualizam as rotas disponíveis escolhendo a melhor delas para manter na FIB; e, no plano de dados, os roteadores encaminham os pacotes usando a FIB. No caso do IP, somente o sistema de roteamento é adaptativo e mantém informação de estado, como também é o único responsável pelo encaminhamento dos pacotes. Na arquitetura NDN, o plano de dados tem dois processos: o envio dos pacotes `INTEREST` e o retorno dos pacotes `DATA` pelo mesmo caminho na ordem inversa. Os roteadores mantêm o estado das requisições pendentes para encaminhamento da resposta. Entre os benefícios do plano de dados NDN estão a facilidade de armazenamento (*caching*), o suporte a *multicast*, e o encaminhamento adaptativo, que é possível pela manutenção do estado na PIT e observação dos pacotes de dados. Neste processo, os roteadores são capazes de fazer

medições relacionadas ao desempenho (*round trip time* e *throughput*), detectar falhas, utilizar rotas alternativas evitando áreas com problemas. Uma opção ocorre pela introdução de um novo tipo de pacote, o `INTEREST NACK`, para indicar que um roteador não pôde encaminhar um pacote por razões diversas (congestionamento, sem interface disponível). O roteador que recebe este pacote pode tomar ações diversas a partir das informações contidas na FIB (preferências de roteamento, informação de desempenho, ordem de encaminhamento). O módulo da arquitetura que trata a estratégia de encaminhamento determina a interface que será utilizada para encaminhar um pacote `INTEREST` e decide de forma *adaptativa* em função das condições da rede. A recepção de pacote `NACK`, ou a ocorrência de *timeout*, força o roteador a fazer nova tentativa por outro caminho. Estudos de casos são feitos e apresentados para os seguintes cenários: falha no enlace, congestionamento, ataque do tipo *sequestro de prefixo* demonstrando vantagens no uso do plano de dados utilizando estado. O sequestro de prefixo corresponde ao anúncio de prefixo por quem não é o proprietário, ou responsável, pelo conteúdo(s) correspondente(s), gerando, consequentemente, a criação de rotas que levam a buracos negros. Nesse caso, como existe o monitoramento das mensagens de dados nas interfaces, não haverá encaminhamento para essas rotas pois não são recebidos dados pelas interfaces correspondentes.

Algumas questões ainda são colocadas para pesquisa: com relação à manutenção de estado por pacote, existem custos de processamento e de armazenamento no roteador. Assim, ficam abertas áreas de pesquisa relacionadas com a redução do tamanho da PIT para ser armazenada de forma eficiente nos roteadores e como fazer buscas e operações em alta velocidade. Com relação às estratégias de encaminhamento, é preciso ainda investigar: como descobrir um novo caminho para um pacote de `INTEREST` novo ou sendo retransmitido? Como utilizar vários caminhos? Além disso, os roteadores periodicamente enviam `INTEREST` para interfaces que falharam ou que não foram utilizadas para procurar melhores caminhos. Disto surgem outras questões: quando fazer esta verificação ou investigação? em quais interfaces? (Yi et al. 2012)

Como parte do trabalho, os autores da proposta CCN descrevem e avaliam o desempenho de um protótipo através de comparações entre TCP e CCN executando aplicações como transferência de arquivo, recuperação de múltiplas cópias de um arquivo de voz (VoCCN, VoIP on top of CCN). Detalhes dos experimentos são relatados em outros trabalhos dos mesmos autores (Jacobson, Smetters, Briggs, Plass, Stewart, Thornton & Braynard 2009).

O grupo apresenta a experiência de desenvolvimento de ferramenta para áudio conferência utilizando CCN. A ferramenta faz propagação do interesse pela informação sobre conferências em toda a rede a fim de coletar a lista das conferências existentes. Após receber a lista de conferências, seleciona uma delas e faz o envio do interesse por esta conferência específica e aguarda o recebimento dos dados. Entre as vantagens do uso do CCN para esta aplicação está a agregação das solicitações pelo mesmo dado enviado por diferentes receptores: somente uma solicitação será encaminhada para o transmissor; além disso, os roteadores com dados armazenados em *cache* respondem a novos pedidos (Zhu, Wang, Yang, Jacobson & Zhang 2011).

### 2.3.3 Network of Information (NetInf)

A arquitetura NetInf (*Network of Information*) (Ahlgren et al. 2008) e (Dannewitz et al. 2013) faz parte dos projetos da União Européia 4WARD [1] e SAIL (Scalable and Adaptive Internet Solutions) [2].

NetInf manipula os conteúdos independente das respectivas localizações e utiliza identificadores (ou nomes planos) para nomear os conteúdos. As informações sobre um determinado conteúdo estão contidas em um *Information Object* (IO) - uma unidade de informação. Cada IO tem um identificador exclusivo que é usado para fazer referência ao conteúdo de forma que este pode ser armazenado em qualquer parte da rede. Este identificador (ID) contém o *hash* criptográfico de uma chave pública que é parte de um par de chaves pública/secreta (PK/SK, ou *Public Key/Secret Key*). Essa chave pública está associada ao conteúdo em si e não ao proprietário. Assim, o conceito de propriedade é retirado do nome (que existe na arquitetura DONA, por exemplo). Um metadado também está associado ao IO e contém informação de segurança.

Para manter o nome persistente, duas abordagens são utilizadas: na abordagem básica, sempre que houver mudança na posse do conteúdo a chave de segurança é repassada ao novo proprietário; na abordagem mais complexa, um novo par de chaves pública/secreta é gerado pelo novo proprietário que passa a autorizar o uso do ID que permanece inalterado. A informação necessária para o processo de autorização é inserida em metadado (Dannewitz, Golic, Ohlman & Ahlgren 2010).

Todos os nós da rede NetInf utilizam o mesmo protocolo com base nas mensagens: `PUBLISH`, `GET` e `SEARCH`. Diferentes implementações serão possíveis com o uso da *Convergence Layer* (CL), uma camada responsável pelo mapeamento das mensagens específicas do protocolo NetInf em pacotes de um protocolo existente (HTTP, IP ou Ethernet) (Dannewitz et al. 2013).

A arquitetura oferece dois modelos de recuperação de conteúdos: via resolução de nomes e via roteamento com base no nome do conteúdo. Conforme o modelo utilizado, os provedores publicam os conteúdos pelo registro do par nome/localização em um sistema de resolução de nomes *Name Resolution Service* (NRS) ou anunciam a informação sobre o conteúdo utilizando protocolo de roteamento. Os receptores podem fazer uso do NRS quando disponível e obter uma ou mais localizações relativas ao conteúdo ou requisitar o conteúdo na rede através da primitiva `GET` que será encaminhada em direção a uma cópia do conteúdo. O uso do NRS é uma forma de melhorar a escalabilidade da rede NetInf.

A Figura 2.4 mostra as duas opções: o receptor utilizando do NRS para obter a melhor rota para recuperar um dado (passos de 1 a 4) e o roteamento pelo nome (passos 5 a 8).

### 2.3.4 Publish/Subscribe Internet Routing Paradigm (PSIRP)

A arquitetura *Publish/Subscribe Internet Routing Paradigm* (PSIRP) é apresentada como uma nova abordagem para a Internet (Tarkoma et al. 2009) (Jokela, Zahemszky, Esteve Rothenberg, Arianfar & Nikander 2009) (Lagutin, Visala & Tarkoma 2010). A proposta é base do

[1] http://www.4ward-project.eu
[2] http://www.sail-project.eu

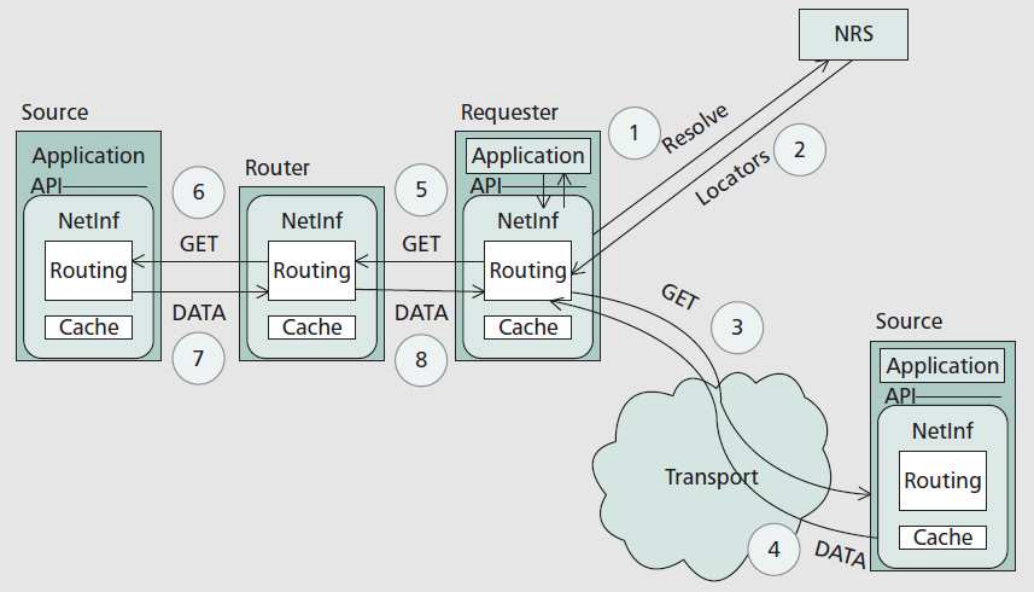



Figura 2.4: Arquitetura NetInf (Ahlgren et al. 2012)

projeto PURSUIT [3] com uma abordagem de comunicação seguindo o modelo *publish/subscriber* na linha *clean-slate*, sem compromisso com protocolos existentes como o IP. Também tem por base a informação e não os nós da rede passando o controle da comunicação para os receptores. Neste modelo de comunicação, o transmissor e o receptor são desacoplados no tempo e espaço tendo a publicação como elo entre ambos. Desacoplados com relação à localização, pois provedor e consumidor não estão conectados diretamente; desacoplados com relação ao tempo, pois a geração e o consumo dos dados não necessitam ocorrer simultaneamente.

Os *subscribers* são usuários que se subscrevem nos identificadores dos conteúdos de interesse. Os *publishers* são os provedores de conteúdo e responsáveis pela publicação dos mesmos na rede. A publicação é uma associação persistente e imutável entre um identificador e um dado criado pelo *publisher*, o que permite armazenamento em qualquer lugar da rede e transfere a responsabilidade pela sincronização para níveis superiores. A rede PSIRP é responsável pelo casamento dos interesses entre usuários e provedores de conteúdo através dos seus identificadores e faz a entrega eficiente do conteúdo às partes interessadas.

A arquitetura PSIRP utiliza vários tipos de identificadores nos níveis de: aplicação *Application level Identifier* (AI), rede *Rendezvous level Identifier* (RI), controle *Scope level Identifier* (SI) e *Forwarding level Identifier* (FI). Para cada conteúdo há dois identificadores: o identificador de escopo (SI) e o identificador de *rendezvous* (RI). RIs e SIs são compostos por um par <P,L> onde P é uma chave pública do proprietário do espaço de nomes e L é um rótulo que codifica a informação sobre a publicação; são autocertificados colaborando no mecanismo de segurança.

Visando eliminar a retransmissão de conteúdo redundante, a rede PSIRP realiza o armazenamento dos conteúdos identificados pelo par de identificadores acima. Novas requisições são prontamente atendidas pela rede sem a necessidade de se obter o conteúdo do servidor. Logo,

[3] http://fp7pursuit.ipower.com

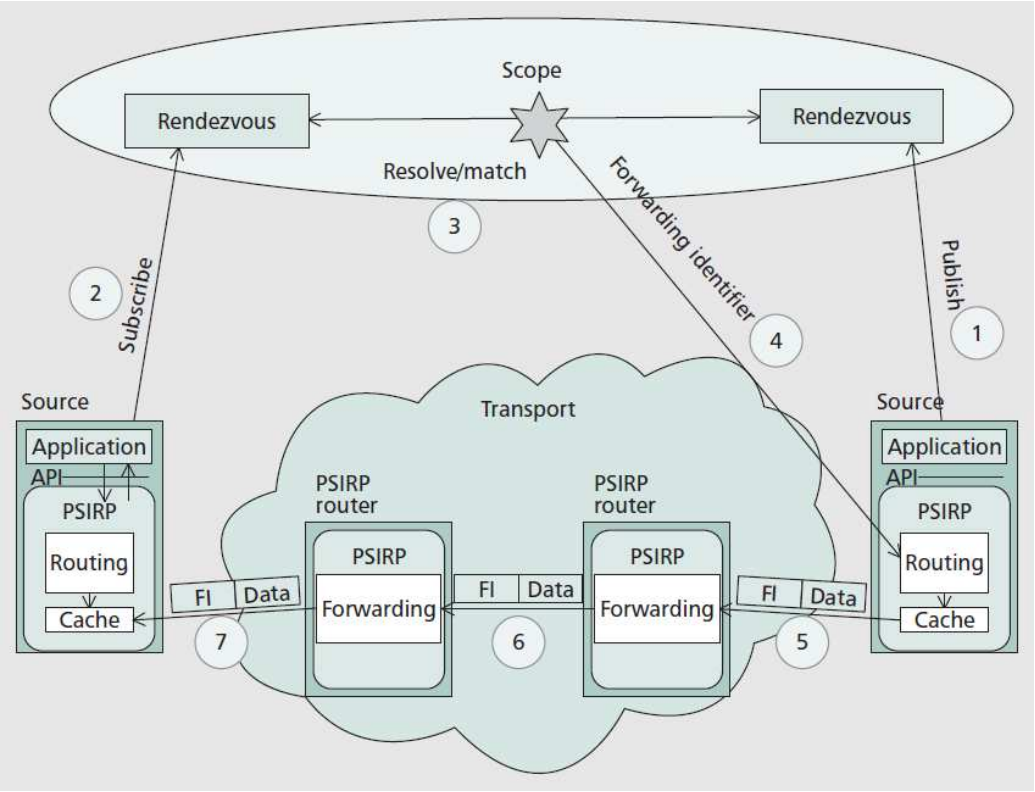



Figura 2.5: Arquitetura PSIRP (Ahlgren et al. 2012)

como é comum nos projetos das ROCs, o armazenamento é uma funcionalidade nativa da arquitetura e está disponível para qualquer usuário.

A arquitetura consiste de quatro módulos: sistema de *rendezvous*, que atua entre os *publishers* e os *subscribers*, localiza as publicações na rede, forma uma rede independente e conectada com uso de uma *Distributed Hash Table* (DHT); função de topologia, que gerencia a topologia física da rede e seleciona rotas que serão utilizadas para entrega das publicações; função de roteamento, responsável por construir e manter a árvore de entregas e armazenar os conteúdos mais populares; e a função de encaminhamento, que faz a entrega aos *subscribers* utilizando filtros de Bloom que mantêm os identificadores para encaminhamento. A separação das funcionalidades visa escalabilidade e independência. A DHT do sistema de *rendezvous* permite distribuir a carga de roteamento entre os domínios participantes.

A Figura 2.5 ilustra a relação entre os módulos e como os proprietários publicam seus conteúdos (passo 1), os usuários se subscrevem aos conteúdos (passo 2) e o casamento entre os dois (passo 3) que resulta no FI (*Forwarding Identifier*) que é encaminhado ao provedor (passo 4) que procede ao encaminhamento do dado (passos de 5 a 7) aos usuários.

### 2.3.5 Content Routers (CR)

A arquitetura de Content Routers (CRs) foi proposta como rede *overlay* sobre IP com foco no *caching* feito pelos CRs. A arquitetura é composta principalmente por roteadores de conteúdo (*Content Routers*) e tem como objetivos: (i) o aumento da eficiência da rede

reduzindo a transmissão de dados redundantes entre provedores de conteúdo e seus clientes através do armazenamento do conteúdo na borda da rede, e (ii) a redução do tempo de resposta às requisições feitas pelos usuários através do acesso a conteúdos mais próximos ao usuário (Wong et al. 2011).

Os roteadores de conteúdo executam no nível acima do IP e são elementos de rede que realizam o encaminhamento de mensagens e o armazenamento temporário de conteúdos. O objetivo desse armazenamento é que solicitações futuras pelo mesmo conteúdo sejam atendidas a partir dos CRs ao invés de percorrer toda a rede até o servidor original. Assim ocorre a recuperação do conteúdo de múltiplas fontes, inclusive mais próximas do usuário, o que gera a necessidade de validação do conteúdo pelo receptor.

A Figura 2.6 mostra a arquitetura dos *Content Routers* com os elementos *Clients*, que fazem a requisição, e os *Servers*, que têm o conteúdo original e respondem às requisições. A rede pode ter outros roteadores IP que não sejam CRs.

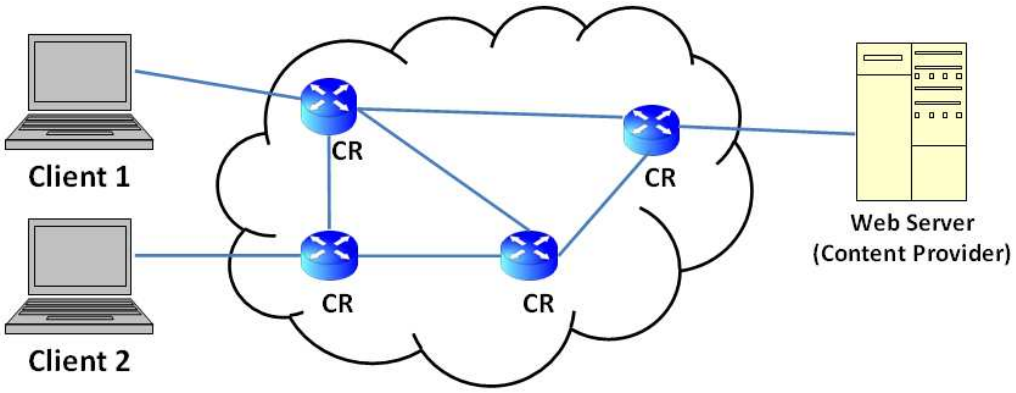



Figura 2.6: Arquitetura de *Content Routers* (CR)

O *chunk* é a unidade básica de comunicação, ou seja, a parte de um conteúdo que será identificada e requisitada. A identificação de um conteúdo resulta de um *hash* criptográfico aplicado ao *chunk* gerando o *chunkId*, uma identificação persistente. Dois tipos de mensagens são utilizadas: `REQUEST` e `RESPONSE`. As mensagens `REQUEST` são encaminhadas na direção do conteúdo pelos CRs e são respondidas por algum CR caso o conteúdo requisitado esteja armazenado no seu repositório local (*cache*).

As mensagens `RESPONSE` são encaminhadas aos usuários (ou *Clients*) que originaram a requisição do conteúdo. No tratamento destas mensagens pelos CRs, pode ocorrer o armazenamento do conteúdo contido na mensagem (*caching*) de acordo com uma função de probabilidade pois o mesmo tem memória limitada (armazenamento ou *caching* probabilístico). Além do armazenamento de conteúdos, o CR armazena informação sobre qual CR encaminhou aquele conteúdo na sua tabela de roteamento. Essa abordagem oportunística simplifica a estratégia de roteamento, já que a construção da tabela de encaminhamento para busca de conteúdo não necessita da implementação de um protocolo complexo.

O roteamento das requisições na direção do conteúdo é feito pelos CRs com base na identificação dos conteúdos (*chunkId*) e dos CRs vizinhos mantidas na tabela de roteamento. Além

dessa tabela, a arquitetura introduz a idéia da busca na vizinhança (*neighborhood search*) com o uso do *neighbor zones*. Essa variável permite o CR desviar a requisição por 'n' saltos antes de direcionar a requisição ao servidor na tentativa de reduzir o tráfego na rede, diminuir a carga no servidor e melhorar o tempo de resposta.

No processo de roteamento, se o CR tem o conteúdo requisitado, ele responde à requisição do cliente que fez o pedido; se ele não tem o conteúdo, ele encaminha a requisição diretamente para o endereço IP do servidor ou para um CR vizinho, onde ele acredita que o conteúdo possa ser encontrado. A arquitetura utiliza mecanismo de árvore binária, *Merkle Tree* (Merkle 1989), para garantir a autenticação e integridade do conteúdo através de assinatura digital.

Trabalho mais recente (Wong et al. 2012) introduz o *caching* cooperativo entre os CRs com objetivo de armazenar mais conteúdo na rede e reduzir conteúdo redundante. O protocolo inclui: (i) a indicação que um determinado conteúdo sendo transmitido por uma mensagem `RESPONSE` já foi armazenado na rede e (ii) o envio da lista de conteúdos armazenados por um CR para os seus vizinhos.

### 2.3.6 Content Inter-Network (CONET)

A arquitetura *Content Inter-Network* (CONET) (Detti, Blefari Melazzi, Salsano & Pomposini 2011) é um sistema que interconecta subsistemas CONET (CSSs - *CONET Sub Systems*). Um CSS contém nós CONET e utiliza uma tecnologia sub-CONET (rede *underlay* IP ou nível de enlace) para transferir dados entre estes nós conforme Figura 2.7.

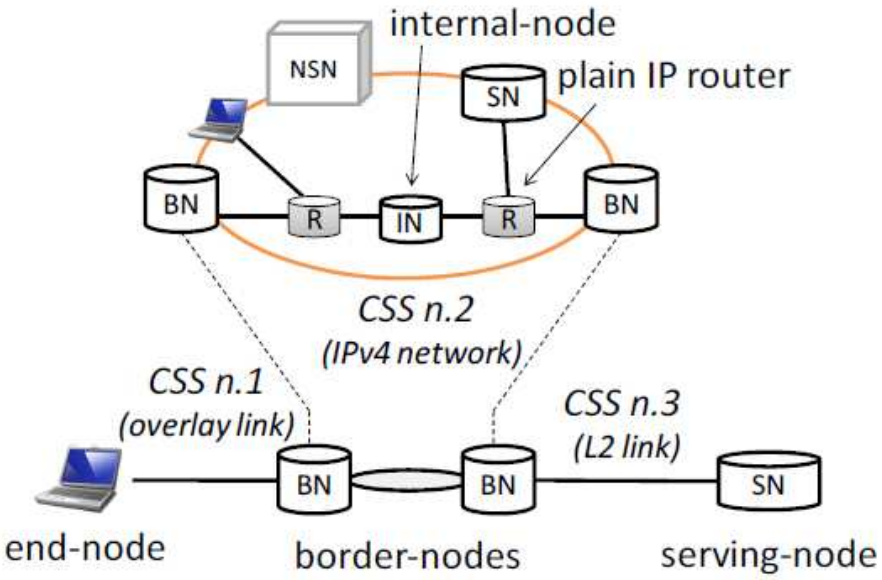



Figura 2.7: Arquitetura CONET (Detti et al. 2011)

Os tipos de elementos (ou nós) da rede CONET são: End Node (EN), que faz envio das requisições; Serving Node (SN), que armazena, anuncia e fornece dados; Border Node (BN), que encaminha e armazena utilizando o mecanismo CONET; Name System Node (NSN), que tem a função de auxiliar no mecanismo de roteamento através de um sistema de nomes centralizado; Internal Node (IN), que faz o *caching* na rede, encaminha utilizando o mecanismo sub-CONET.

Todos os nós da rede apresentam dois níveis: (i) o nível CONET, sem conexão, que manipula pacotes do tipo CONET Information Unit (CIU), com funcionalidades de *caching* e (ii) o nível sub-CONET. Os nós dentro de um subsistema CONET usam um espaço de endereçamento homogêneo e autônomo e, quando necessário, também um protocolo de roteamento.

A arquitetura é inspirada nas redes orientadas a conteúdo com as seguintes características: não mantém informação sobre a comunicação nos nós da rede (estado); roteamento na origem; tabela de rotas com base em nomes, de tamanho limitado; substituição de informações mais velhas pelas mais novas; as rotas desconhecidas são consultadas no sistema de nomes centralizado (NSN) e atualizam a tabela de rotas local; solução pode ser integrada com a rede IP atual.

CONET utiliza um identificador de rede no formato <*namespaceID, name*> e utiliza nomes planos no formato <*Principal,Label*>.

A interconexão dos subsistemas CONET pode ocorrer no nível 2, nível 3 ou em nós conectados ponto a ponto. A arquitetura suporta três tipos de implementação: (a) *clean-slate*: acima de tecnologias de nível 2 e neste caso substituindo o nível IP; (b) *overlay*: acima do nível IP; (c) integrada: como uma extensão do nível IP através do uso de opção do cabeçalho de forma que o IP tenha conhecimento do conteúdo. A opção de integração (c) é a forma proposta de convivência da rede atual com a rede CONET sendo que os elementos BN e IN seriam roteadores IP que implementam a funcionalidade CONET. Nestes roteadores um *fast forwarding path* observa o cabeçalho da mensagem e faz o encaminhamento de pacotes CONET e pacotes IP. As tabelas de roteamento no hardware (RIB ou FIB) incluem prefixos IP e nomes que podem endereçar dados remotos ou locais, neste caso apontam para o *cache* local.

Os nós CONET trocam unidades de informação próprias, as *CONET Information Units* (CIUs), sendo que os *interest* CIUs transportam as requisições e os *named-data* CIUs transportam partes de um arquivo ou *chunks*. Os CIUs trafegam em pacotes CONET (ou *carrier packets*). A Figura 2.8 mostra as unidades de informação e pacotes CONET.

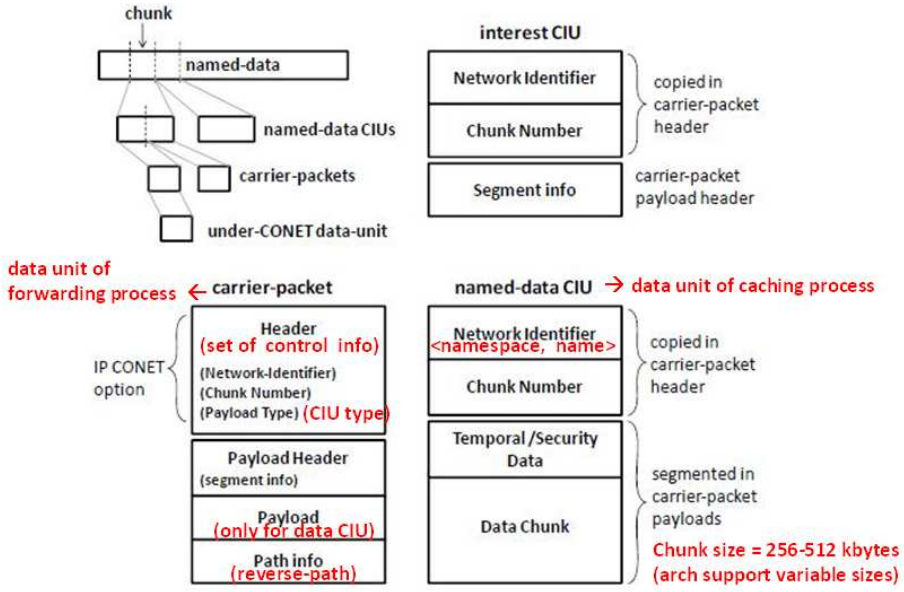



Figura 2.8: Unidades de Informação CONET (Detti et al. 2011)

O modo de operação da rede CONET comporta um processo de anúncio iniciado pelo SN

com encaminhamento da informação na rede CONET, o que permite a um EN encontrar um conteúdo pelo nome. A recuperação de conteúdos nomeados consiste de duas fases: requisição e entrega de dados.

Na fase de requisição: um EN produz e propaga um *interest CIU*; o *interest CIU* é encapsulado em um pacote I; EN e BNs intermediários encaminham pelo nome o pacote I; este processo seleciona o endereço do CSS do próximo BN em direção ao SN; o mecanismo de roteamento encapsula o pacote I em uma unidade de dados sub-CONET e usa o endereço CSS como destino.

Na fase de entrega de dados: o primeiro nó capaz de prover o dado irá enviar o CIU de volta sem continuar a propagação do pacote I; o *named-data CIU* é encapsulado em um pacote C; o pacote percorre os mesmos CSSs na direção inversa sendo que SN e o BN utilizam informação do campo de controle (roteamento na origem); dentro de um CSS, a tecnologia abaixo do nível CONET (ex. IP) executa o roteamento do pacote C; todos os BN e IN podem armazenar o dado contido no pacote C em seu cache local.

### 2.3.7 Resumo das Arquiteturas

Em sua essência, uma Rede Orientada a Conteúdo provê conteúdo e não um canal de comunicação com outro *host*; conhece conteúdos através de seus nomes e não de sua localização. Esta abordagem traz alguns problemas relacionados ao tamanho das tabelas de roteamento e à eficiência do processo de busca uma vez que a quantidade de conteúdos disponíveis é muito maior do que a quantidade de máquinas. Algumas das características de uma Rede Orientada a Conteúdo:

- Esquema de endereçamento com base em nomes, sem referência a localização;
- Faz o roteamento das requisições de usuários em direção à cópia do conteúdo que estiver mais próxima;
- Funcionalidade de *caching* na rede para tornar a entrega de conteúdo mais eficiente;
- Utiliza informação de segurança com foco no conteúdo evitando difusão de conteúdo falso e protegendo o mesmo.

A Figura 2.9 resume as arquiteturas estudadas com foco nas principais características das redes orientadas a conteúdo. A arquitetura dos CRs será considerada no próximo capítulo.

## 2.4 Conclusão do Capítulo

Este capítulo apresentou uma análise das principais arquiteturas de ROC propostas na literatura. Algumas delas serviram de base para esta dissertação como é o caso da arquitetura dos Content Routers (CR) e da arquitetura CCN. A arquitetura CONET é apresentada como um exemplo de uso do nível de enlace para a interconexão dos subsistemas. Na arquitetura NetInf existe a possibilidade da camada de convergência utilizar o nível de enlace, assim como a camada de estratégia no CCN.

Vários conceitos e características das ROCs abordados neste capítulo como, por exemplo, roteamento e armazenamento de conteúdos são itens importantes da arquitetura proposta no próximo capítulo.

| Características | ARQUITETURAS | | | | |
|---|---|---|---|---|---|
| | DONA | CCN | PSIRP | NetInf | CONET |
| Descrição | Infraestrutura para resolução de nomes (substitui DNS) | rede orientada a conteúdo | paradigma publish/subscriber | rede de informações | sistema que interconecta subsistemas CONET |
| Nomeação | Nomes planos; formato Principal:Label (P:L); agregação pelo publicador (explícita) | Nomes hierárquicos (legíveis); permite agregação de prefixos | 2 identificadores planos: Scope Identifier (SI) e Rendezvous Identifier (RI); agregação possível pelo escopo | Nomes planos | Nomes planos |
| Registro dos conteúdos em tabelas | Publicados na rede pelos provedores | Publicados nos nós e informação sobre localização é distribuída por protocolo de roteamento | Provedores publicam os identificadores na rede - relaciona com um escopo | Conteúdos são publicados através do registro de nome e localização no Name Resolution Service (NRS); ou conteúdos são anunciados via protocolo de roteamento | Processo de anúncio iniciado pelo SN: tabela de rotas e NSN (sistema centralizado) |
| Encaminhamento da Requisição | Esquema hierárquico estruturado, em árvore - FIND encaminhado em direção ao RH mais próximo até ao provedor | INTEREST encaminhado para roteadores em direção ao dado (FIB). PIT mantém estado da pendências (rastro=lista de interfaces requisitantes) | Receptores se subscrevem a um nome de interesse via NRS (sistema rendezvous) | 2 modelos: Name Resolution Service (NRS) e Name-based resolution. Receptores podem encaminhar para NRS ou aos roteadores. | EN gera Interest CIU que é encaminhado em direção a um SN |
| Entrega de dados | Pelo caminho reverso (RH) ou diretamente a quem requisitou (via IP) | Pelo caminho reverso usando dados da PIT (estado mantido no roteador) | Sistema rendezvous faz o mapeamento entre publicação e subscrição; Forwarding Identifier (FI) é enviado ao provedor que inicia processo de entrega de dados. FI identifica um caminho usando filtro de Bloom. | O conteúdo é encaminhado para o receptor via caminho reverso ou diretamente (via IP) | Direção inversa utilizando dados do campo de controle (roteamento na origem) |
| Armazenamento | habilitado no Resolution Handler (RH) | habilitado em todos os elementos; ocorre na recepção de DATA | funcionalidade nativa da arquitetura | Storage - long term memory Caching - dynamic short term memory | SN, BN e IN |
| Segurança | Nomes autocertificáveis | Chave pública; processo de certificação externo | Nomes autocertificáveis | Nomes autocertificáveis | informação não disponível |
| Mensagens ou Primitivas | REGISTER FIND / DATA (síncrono) | INTEREST / DATA (síncrono) | PUBLISH / SUBSCRIBE | PUBLISH / GET / SEARCH (request/response) | Pacotes (carrier packets) trafegam Interest e Data CIU |
| Elementos da Rede | Resolution Handler (RH) | Roteadores | Sistema rendezvous | Roteadores, Name Resolution Service (NRS) | EN, SN, BN, NSN, IN |
| Código fonte | | www.ccnx.org | www.fp7-pursuit.eu/Pursuit Web/?page_id=12 | www.netinf.org | |

Figura 2.9: Resumo das Arquiteturas

Capítulo 3

# Arquitetura de CRs no Nível do Enlace

Este capítulo apresenta a proposta de uma rede orientada a conteúdos no nível da camada de enlace (Ethernet[1]) utilizando conceitos da arquitetura de *Content Routers* (CRs) sem uso de endereçamento de rede, priorizando o conteúdo.

Os CRs são os elementos de rede que realizam o encaminhamento de mensagens e o armazenamento (*caching*) de conteúdos. Também são considerados elementos da rede: Cliente, que faz as requisições e Servidor, que tem o conteúdo original, que faz o anúncio do mesmo e responde às requisições.

A arquitetura dos CRs (Wong et al. 2011) foi definida como uma rede *overlay* sobre IP com suporte ao *caching* e utilização de nomes planos, entre outras características conforme apresentado em 2.3.5.

Diferente do ambiente IP onde existe o conhecimento do endereço do provedor, na proposta de arquitetura no nível de enlace (Ethernet) parte-se do princípio de que não se tem qualquer informação da localização do conteúdo e, nesse caso, a requisição de conteúdos é feita através de mecanismo de inundação de mensagens pelas interfaces disponíveis nos CRs (Ambiel et al. 2013b). Esse tipo de procedimento traz questões relacionadas à quantidade de mensagens geradas na rede. Assim, para reduzir o número de mensagens, além do roteamento oportunístico baseado nas mensagens de `RESPONSE` e do mecanismo de *Neighbor Zones* já existentes na arquitetura original, introduz-se:

- lista de requisições pendentes por interface;
- conceito de Super CR como elemento especializado em *caching* ou outras funcionalidades;
- primitivas para anúncio de conteúdos e anúncio do Super CR, `ANNOUNCE_CONTENT` e `ANNOUNCE_SCR` respectivamente;
- indicação de menor distância, em número de *hops*, na tabela de roteamento e na lista de Super CRs.

Todos esses itens são explicados neste capítulo.

[1]Ethernet é uma família de tecnologias para redes locais cabeadas padronizada na especificação 802.3 do IEEE. Pode atingir velocidade de até 10Gbit/s.

A proposta no nível de enlace apresenta-se como uma alternativa interessante para redes menores tais como as redes domiciliares onde os custos relativos às dependências do IP (ex: gerência de rede, configuração de protocolos de roteamento e interfaces, segurança) seriam reduzidos com uma migração da arquitetura nativa da rede para um serviço de busca e entrega de conteúdo (Ambiel et al. 2013a).

Considerando as características de Redes Orientadas a Conteúdo (ROC) apresentadas no capítulo anterior, a Figura 3.1 resume as características adotadas pela arquitetura dos CRs. Também identifica as diferenças entre a arquitetura original na forma de uma rede *overlay* sobre IP e a arquitetura definida nesse capítulo, diretamente no nível de enlace.

| Características | ARQUITETURA de CRs (Overlay/IP) | ARQUITETURA de CRs (Nível de Enlace) |
|---|---|---|
| Descrição | rede orientada a conteúdo no nível overlay/IP | rede orientada a conteúdo no nível de enlace |
| Nomeação | - nomes planos ou identificadores: *chunkId* ;<br>- mapeamento entre nome do conteúdo e identificador por um Sistema de Resolução de Nomes | idem Arquitetura de CRs |
| Registro dos conteúdos em tabelas | - ocorre oportunisticamente na recepção de RESPONSE | - ocorre oportunisticamente na recepção de RESPONSE<br>- ocorre opcionalmente pelo anúncio dos identificadores dos conteúdos pelo provedor (ANNOUNCE_CONTENT) |
| Encaminhamento da Requisição | REQUEST encaminhado a um endereço IP com base:<br>- na tabela de roteamento, indexada pelos *ChunkIds*, e com uso da informação de *NeighborZones*<br>- no endereço IP de destino (servidor) | REQUEST encaminhado a uma interface com base:<br>- na tabela de roteamento, indexada pelos *ChunkIds* , e com uso da informação de NeighborZones<br>- *flooding* pelas interfaces disponíveis exceto a de recepção<br>O *ChunkId* requisitado é incluído na lista de requisições pendentes, por interface (estado mantido no CR) |
| Entrega de dados | RESPONSE encaminhado para o endereço IP requisitante (origem no REQUEST será destino no RESPONSE) | RESPONSE encaminhado pelo caminho reverso usando informação da lista de requisições pendentes. *ChunkId* é removido da lista. |
| Armazenamento | - ocorre probabilisticamente na recepção de RESPONSE<br>- habilitado em todos os CRs | idem Arquitetura de CRs |
| Segurança | Nomes autocertificáveis (*Chunk Ids* ) | idem Arquitetura de CRs |
| Mensagens ou Primitivas | REQUEST / RESPONSE | REQUEST / RESPONSE / ANNOUNCE_CONTENT / ANNOUNCE_SCR |
| Elementos de Rede | Content Routers (CR), Cliente, Servidor | Content Routers (CR), Cliente, Servidor, Super CR (S_CR) |

Figura 3.1: Características da Arquitetura de CRs

## 3.1 Definições

As seguintes definições, já presentes na proposta original, são apresentadas para facilitar o entendimento da arquitetura de CRs no nível de enlace:

- ***Cryptographic Identifier* (CryptoId):** Identificação do conteúdo que resulta de um *hash* criptográfico aplicado a um bloco de dados. Quando aplicado ao *Data Chunk*, resulta no *chunkId*. Quando aplicado à identificação do CR, resulta no *CRId*, utilizado para controle de *loops* e identificação de Super CR.

- ***Data Chunk*:** Unidade básica de comunicação; parte de um conteúdo (ex. arquivo) que

será identificado e requisitado. Cada *Data Chunk*, ou simplesmente *chunk*, é identificado pelo seu *chunkId*. A Figura 3.2 apresenta este processo.

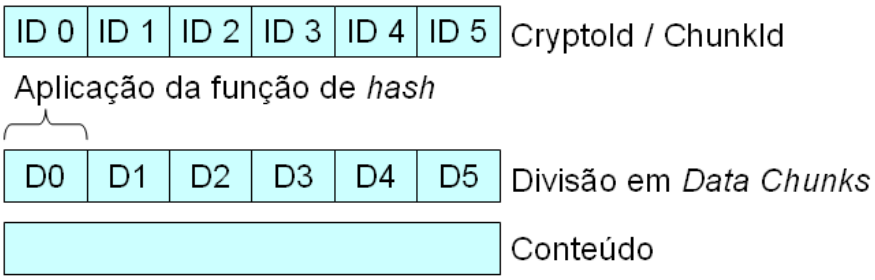



Figura 3.2: Geração dos *chunkIds*

- ***Metadata*:** Estrutura de metadados que agrega a lista de *chunkIds* relacionados a um conteúdo e outras informações adicionais como versão e validade. Informação gerada pelo provedor do conteúdo (Servidor). O Cliente precisa obter o metadado, por exemplo através de um sistema de nomes, para ser capaz de proceder às requisições dos *Data Chunks* através da lista de *chunkIds*.

- ***Caching Threshold*:** Determina a probabilidade de um CR armazenar um conteúdo encaminhado por ele.

- ***Neighbor Zones* (NZones):** permite ao CR desviar uma requisição até o limite de $n$ saltos na direção de um servidor. Como esse número pode indicar uma área em volta do CR, usa-se o termo *zones*. No nível de enlace, o NZones ajuda a controlar a inundação de mensagens sendo que a informação da tabela de roteamento é sempre utilizada antes de recorrer ao processo de inundação. Além disso, quando nZones atinge o valor limite, o processo de busca pode ser encerrado ou continuar através de inundação de mensagens. No ambiente IP, o NZones indica quantos CRs uma requisição irá visitar antes desta mesma requisição ser encaminhada diretamente para o servidor utilizando o endereçamento IP.

- ***Visited Neighbors* (BFvisited):** estrutura de dados compactada do tipo filtro de Bloom, presente no cabeçalho das mensagens. Essa estrutura contém a identificação dos CRs (*CRId*) visitados pela mensagem com o objetivo de evitar *loops*. Aplica-se a todos os tipos de mensagens.

## 3.2 Características Principais da Arquitetura

As principais características adotadas para a arquitetura de CRs no nível de enlace são descritas a seguir. As três primeiras características listadas: Nomeação, Armazenamento e Segurança são utilizadas como na proposta original (Wong et al. 2011).

- **Nomeação:** um identificador plano (*chunkId*) é utilizado para nomear os conteúdos. A opção por este tipo de identificação deve-se ao fato de atender ao requisito de persistência

pois está desacoplado da informação de localização. Tais identificadores também dão suporte ao modelo de segurança adotado na arquitetura, o qual associa o *chunkId* de um bloco de dados do conteúdo ao seu respectivo provedor atendendo, dessa forma, ao requisito de autenticação (Wong 2011).

A arquitetura pressupõe um processo de mapeamento entre o nome do conteúdo e a lista de *chunkIds* que compõem o mesmo conteúdo. Diferente do NRS que está sempre associado ao mapeamento de um nome a um endereço (conforme apresentado 2.2.2), a resolução de nomes na arquitetura de CRs está relacionada ao mapeamento de um nome a um identificador ou ao *Metadata*. Este trabalho sugere uma outra solução associada ao Super CR em 3.3.1.

- **Armazenamento:** o CR inspeciona o cabeçalho de cada mensagem em trânsito e armazena uma cópia do conteúdo da mensagem (os dados) com certa probabilidade (*Caching Threshold*). O objetivo desse armazenamento é que solicitações futuras pelo mesmo conteúdo sejam atendidas a partir dos CRs ao invés de percorrerem toda a rede até o servidor original, aumentando a disponibilidade do conteúdo na rede e reduzindo o tempo de recuperação do conteúdo pelo cliente requisitante.

- **Segurança:** Os *chunkIds* são identificadores auto certificáveis. A arquitetura de CRs utiliza *Merkle Tree* para autenticar o conteúdo particionado na forma de *Data Chunks*, recuperado a partir de fontes diversas. Trata-se de uma árvore binária construída sobre os blocos de dados com um elemento raiz no topo da árvore ao qual é aplicado uma função de *hash*. Esta raiz é uma assinatura digital que será recuperada pelos clientes de forma segura no metadado e utilizada para autenticar o conteúdo recebido (*Data Chunks*).

### 3.2.1 Mensagens ou Primitivas

Além das mensagens definidas na proposta original, `REQUEST` e `RESPONSE`, este trabalho introduz as mensagens `ANNOUNCE_CONTENT` e `ANNOUNCE_SCR`. Ambas são válidas para as duas abordagens: IP/Overlay e nível de enlace.

#### Cabeçalho das Mensagens

Todas as mensagens utilizam um cabeçalho comum que as identificam, além de conter informações necessárias ao processamento. A Tabela 3.1 ilustra o cabeçalho das mensagens:

Tabela 3.1: Cabeçalho das mensagens

| messageType | CryptoId | nZones | BFvisited | numHops | lastCR | source | destination | data |
|---|---|---|---|---|---|---|---|---|

Os campos do cabeçalho indicam:

- messageType: tipo da mensagem (`REQUEST`, `RESPONSE`, `ANNOUNCE_CONTENT` ou `ANNOUNCE_SCR`).

- CryptoId ou *Cryptographic Identifier*: *chunkId* nas mensagens `REQUEST`, `RESPONSE`, `ANNOUNCE_CONTENT` e CRId na mensagem `ANNOUNCE_SCR`.

- nZones: valor do *Neighbor Zones*.
- BFvisited ou *Visited Neighbors*: conjunto dos CRIds dos CRs visitados pela mensagem na forma de filtro de Bloom.
- numHops: contador do número de *hops*.
- lastCR: identificação do último CR que encaminhou o conteúdo; é atualizado a cada CR no tratamento da mensagem `RESPONSE`. Utilizado somente na abordagem IP/*overlay* para dar suporte à busca na vizinhança (*Neighbor Search*).
- source: endereço IP do cliente que requisitou o conteúdo (somente para a abordagem IP/*overlay*).
- destination: endereço IP do provedor do conteúdo (somente para a abordagem IP/*overlay*).
- data: dado ou *Data Chunk* retornado pela mensagem `RESPONSE`.

**Descrição das Mensagens**

- `REQUEST`: mensagem de requisição, originada pelo Cliente que requisita um *chunkId*; encaminhada por inundação e controlada pelo valor de *NeighborZones* (NZones) e pela lista de requisições pendentes; *numHops* é incrementado para indicar a distância para a resolução de um conteúdo; NZones é decrementado a cada CR.
- `RESPONSE`: originada por um Servidor ou por um CR que tem o dado correspondente ao *chunkId* armazenado no seu cache em resposta a um `REQUEST`; *numHops* é incrementado para indicar distância do conteúdo.
- `ANNOUNCE_CONTENT`: originada pelo Servidor, ou eventualmente por um CR, para anunciar os *chunkIds* disponíveis; encaminhada por inundação e controlada pelo valor de *NeighborZones* (NZones); *numHops* é incrementado para indicar a distância do conteúdo e também auxiliar no controle de inundação, conforme explicado em 3.2.4. A mensagem pode ser originada por um Super CR na situação indicada na Figura 3.9. A introdução desta mensagem na arquitetura de CRs no nível de enlace visa diminuir o número de mensagens: a inclusão dessa informação (*chunkIds*) na tabela de roteamento reduz a necessidade do mecanismo de inundação por parte dos CRs.
- `ANNOUNCE_SCR`: originada pelo CR que está configurado como Super CR para anunciar seu CRId; encaminhada por inundação e controlada pelo valor de *NeighborZones* (NZones); *numHops* é incrementado para indicar a distância do Super CR e também auxilia no controle de inundação como no caso da mensagem `ANNOUNCE_CONTENT`; NZones é decrementado a cada CR.

Todas as mensagens são tratadas pelos CRs, que também geram as mensagens `ANNOUNCE_SCR` e `RESPONSE` em situações que serão explicadas a seguir. O elemento Cliente gera mensagens `REQUEST`, aguarda `RESPONSE` e ignora qualquer outra mensagem. O elemento Servidor gera as

mensagens ANNOUNCE_CONTENT e RESPONSE, aguarda eventuais REQUEST caso essas mensagens não tenham sido atendidas pelo *caching* dos CRs e ignora outras mensagens.

### 3.2.2 Roteamento da Requisição

As mensagens REQUEST são roteadas com base nos identificadores de conteúdo, os *chunkIds*. A tabela de roteamento é preenchida com a informação anunciada pelos servidores (ou CRs) via ANNOUNCE_CONTENT ou de forma oportunística com a informação da mensagem RESPONSE que trafega na rede. Desta forma, o CR cria o conhecimento das interfaces, ou das direções, por meio das quais ele pode alcançar um conteúdo, independente desse conteúdo estar disponível no seu cache local. A tabela mantém somente a melhor interface que pode levar a um conteúdo, ou seja, a menor distância em *hops*. Quando uma mensagem REQUEST alcança um CR que tem uma cópia do conteúdo requisitado, o CR então gera a mensagem RESPONSE.

Na arquitetura proposta no nível de enlace, caso o identificador do conteúdo (*chunkId*) não seja encontrado na tabela de roteamento do CR, é feita a inundação da requisição por todas as interfaces disponíveis, exceto a interface de recepção, até que se alcance o limite definido pela variável *Neighbor Zones*. O CR mantém informação sobre quais conteúdos (*chunkIds*) estão sendo aguardados por interface (de recepção) equivalente à tabela PIT (Pending Interest Table) da arquitetura CCN (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009). Assim, o CR não encaminha requisições por conteúdos que já estão nas listas de pendências.

A Figura 3.3 apresenta o encaminhamento da mensagem REQUEST quando a rede não tem o conhecimento da interface relacionada a um determinado *chunkId*. Nesse caso, os CRs encaminham a mensagem REQUEST para as interfaces disponíveis, exceto a interface de recepção. O *loop* será contido pela informação do campo *Visited Neighbors* presente no cabeçalho da mensagem e atualizado a cada CR que acrescenta sua própria identificação (*CRId*): mensagens que já foram "marcadas" por um determinado CR serão descartadas se voltarem a ele (*loop*). O *flooding* é contido também pela variável *Neighbor Zones* e pela lista de requisições pendentes.

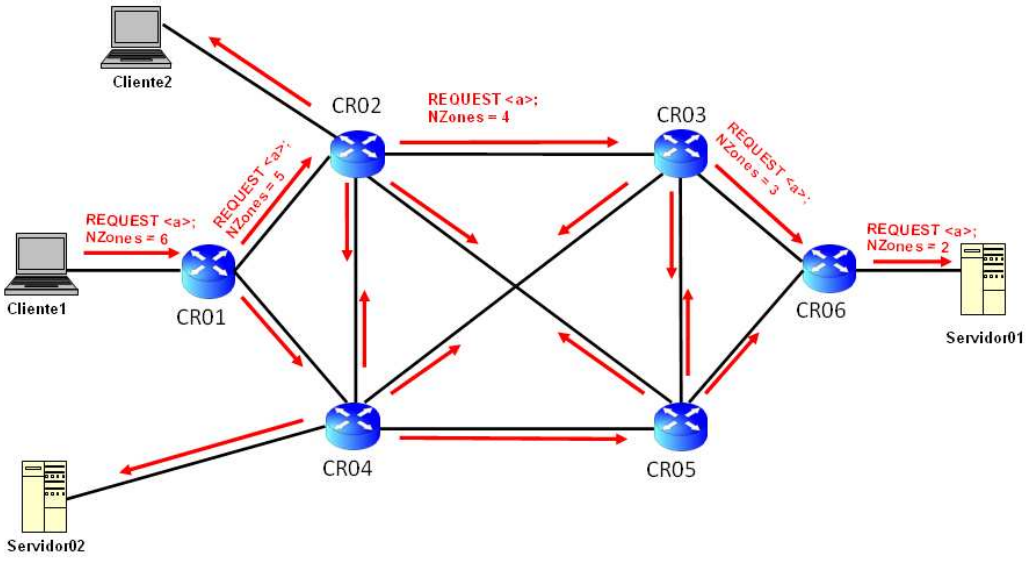



Figura 3.3: Encaminhamento do REQUEST - NZones = 6: Impacto do *flooding*

A inundação apresentada na Figura 3.3 corresponde à situação de pior caso pois, dependendo da ordem de recebimento das mensagens `REQUEST` nos CRs, algumas podem ser ou não inundadas nas outras interfaces. Supondo que a mensagem `RESPONSE` retornou por uma determinada interface e os CRs "aprenderam" essa direção, a Figura 3.4 apresenta o encaminhamento da mensagem `REQUEST` quando a rede já tem o conhecimento da interface relacionada a um determinado *chunkId*. Nesse caso, os CRs encaminham a mensagem `REQUEST` pela interface indicada na tabela de roteamento como sendo a melhor opção em número de *hops*. O encaminhamento será feito por `N` *hops* conforme a informação do campo *Neighbor Zones* (no exemplo com o valor 6) presente na mensagem e atualizado (decrementado) a cada CR. Neste exemplo não ocorre *flooding*.

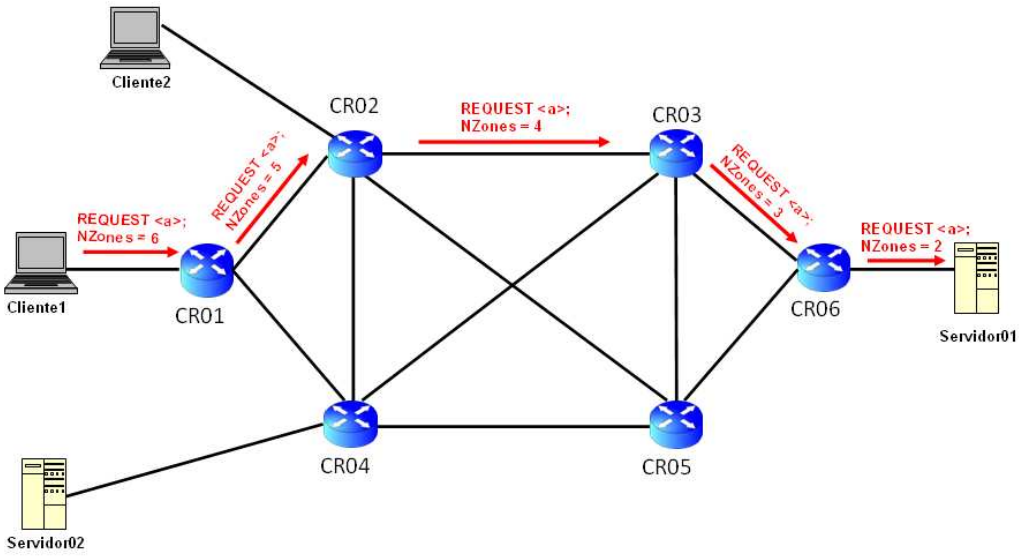
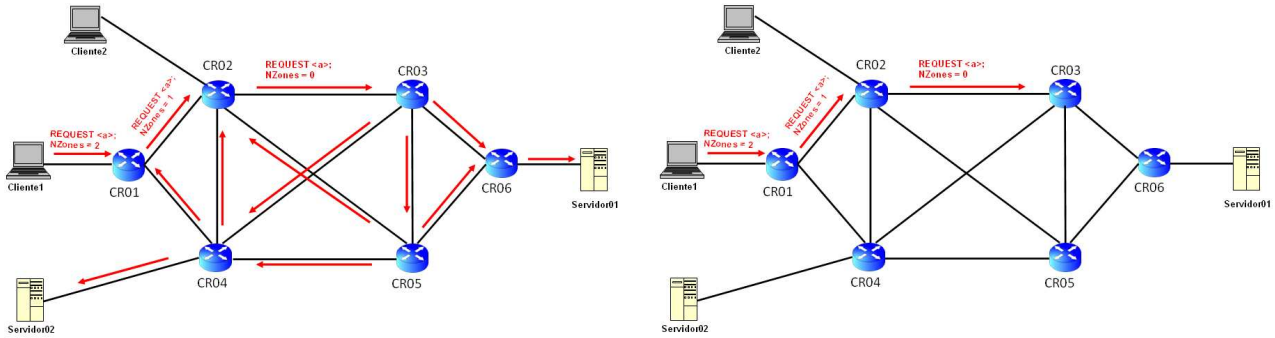



Figura 3.4: Encaminhamento do `REQUEST` - NZones = 6: Sem *flooding*

As Figuras 3.5(a) e 3.5(b) apresentam o encaminhamento da mensagem `REQUEST` quando a rede já tem o conhecimento da interface relacionada a um determinado *chunkId* e a variável *Neighbor Zones* tem um valor menor que o número de *hops* até o servidor. Assim, ao atingir o valor zero, que no exemplo acontece no CR03, dois comportamentos são possíveis: o *flooding* pelas interfaces disponíveis (3.5(a)) ou a finalização da busca (3.5(b)).

Um dos motivos de recorrer ao *flooding* é aprender a direção de um conteúdo, eventualmente melhor. Essa situação pode ocorrer quando não se tem o anúncio dos conteúdos por parte dos servidores e, também, não houve o aprendizado oportunístico através da passagem da mensagem `RESPONSE`. Como exemplo, no caso da Figura 3.5(a), se o Servidor02 passa a responder à requisição feita pelo Cliente1, haverá atualização na tabela de roteamento de alguns CRs pois este servidor está mais próximo com relação a número de *hops*.

Finalizar a busca pode ser uma opção para minimizar a quantidade de mensagens quando o valor de NZones é suficiente para atingir um servidor ou algum CR que tenha o conteúdo em cache. Caso contrário, esta finalização do processo de busca resulta em "lixo" na lista de requisições pendentes. Situação similar ocorre quando o conteúdo não é encontrado nos servidores. Neste caso, o uso de uma mensagem `NACK`, como sugerido em (Yi et al. 2012),




(a) Opção Com Flooding

(b) Opção Sem Flooding

Figura 3.5: Encaminhamento do `REQUEST` - NZones = 2

encaminhada pelo CR ou servidor, ajudaria na manutenção da lista de pendencias.

Um trabalho recente (Chiocchetti et al. 2012) compara o encaminhamento determinístico em direção a um conteúdo (ou cópia do conteúdo) conhecido e a exploração da rede em direção a uma cópia não conhecida (ex. via inundação da requisição). Além da possibilidade de uso dessas estratégias, os autores apontam o uso de uma estratégia híbrida para escopo intra-domínio. O conceito de *Neighbor Zones* utilizado na arquitetura de CRs é similar ao conceito do *Kpercentile* proposto na estratégia híbrida. No mesmo trabalho, os autores reportam que a estratégia de explorar a vizinhança pode melhorar o desempenho na entrega de dados pela descoberta de cópias do conteúdo mais próximas ao usuário que fez a requisição.

### Estado das Requisições Pendentes

Para evitar o reencaminhamento de `REQUEST` para um mesmo *chunkId* já encaminhado anteriormente, a arquitetura de CRs no nível de enlace adota a ideia de uma lista de requisições pendentes como a tabela PIT apresentada na arquitetura CCN (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009). Os CRs mantêm a informação de quais conteúdos (*chunkIds*) estão pendentes. As listas de pendências são mantidas por interface e são verificadas na recepção da mensagem `REQUEST` para evitar reencaminhamento e na recepção da mensagem `RESPONSE` para identificar a(s) interface(s) para a entrega do conteúdo. O conteúdo é removido da lista de pendências após o encaminhamento da mensagem `RESPONSE` a todas as interfaces que têm o conteúdo nas respectivas listas.

A Figura 3.6 apresenta a sequência temporal do tratamento das mensagens `REQUEST` e `RESPONSE` com foco no estado das requisições pendentes.

### O roteamento das requisições na abordagem IP/*overlay*

No ambiente IP o cabeçalho das mensagens contém os endereços IP do cliente (origem) e do servidor (destino). Assim, no tratamento da mensagem `REQUEST`, se o conteúdo não é encontrado na tabela de roteamento, a requisição é encaminhada ao servidor utilizando o endereço IP de destino e tendo por base o protocolo de roteamento ativo na rede (ex: *Open Shortest Path First - OSPF*), ou seja, neste ambiente existem dois níveis de roteamento: inicialmente no conteúdo e, posteriormente, pelo roteamento IP na direção do servidor do conteúdo.

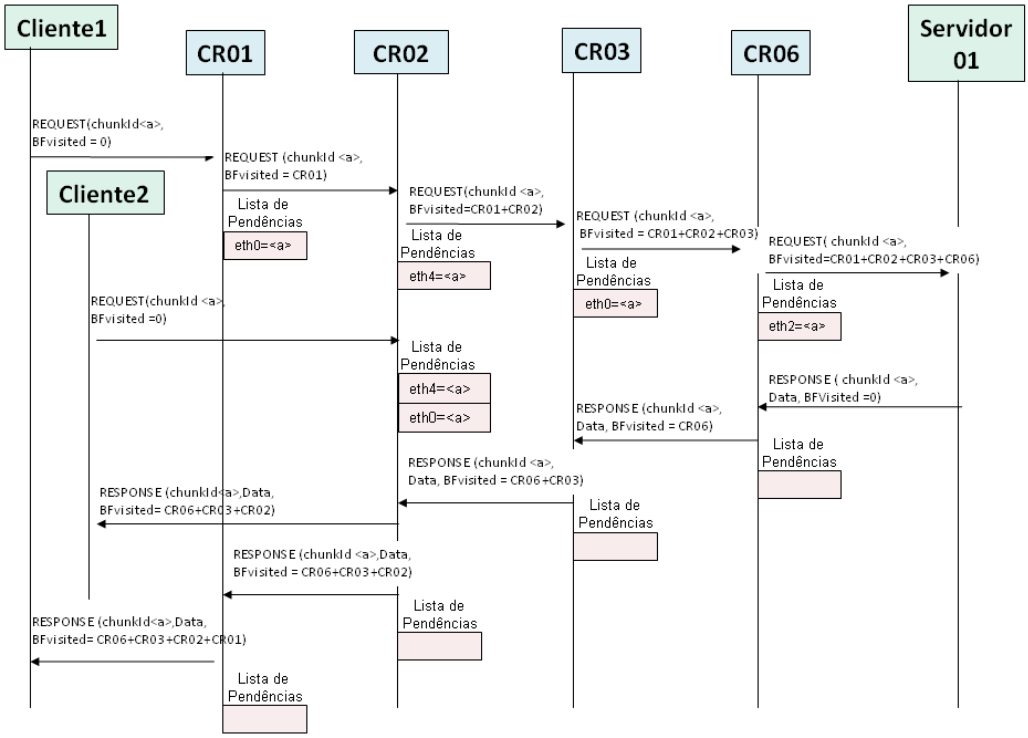



Figura 3.6: Estado das Requisições Pendentes no tratamento de `REQUEST` e `RESPONSE`

O roteamento IP ocorre em duas situações: (i) quando não existe conhecimento do conteúdo e (ii) quando o limite definido pela variável *Neighbor Zones* é alcançado.

### 3.2.3 Mecanismo de Entrega de Dados

As mensagens `RESPONSE` são geradas pelos servidores ou pelos CRs quando estes têm a cópia do conteúdo requisitado.

Na arquitetura proposta no nível de enlace, a mensagem `RESPONSE` é encaminhada pelo caminho inverso ao da requisição e sem *loops* como o encaminhamento dos dados da arquitetura CCN (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009). Ao receber uma mensagem `RESPONSE`, o CR verifica a lista de pendências de cada interface para o *chunkId* recebido e repassa a mensagem às interfaces conforme ocorre o *matching* e o *chunkId* é removido da lista de pendências. Eventuais mensagens duplicadas geradas por outras fontes do mesmo conteúdo devido ao *flooding* da mensagem `REQUEST` serão descartadas pelo CR.

No tratamento da mensagem `RESPONSE`, a tabela de *caching* é probabilisticamente preenchida em função do parâmetro *Caching Threshold* enquanto que a tabela de roteamento é oportunisticamente preenchida, ou atualizada, em função da informação de número de *hops* contida na mensagem. Somente a melhor direção é mantida na tabela: a interface do CR que encaminhou a mensagem informando número de *hops* menor. A informação de número de *hops* é atualizada a cada CR (incrementada).

O conceito da melhor direção é introduzido junto com a proposta da arquitetura dos CRs no nível de enlace e outros critérios podem ser utilizados a partir de medições relacionadas ao

desempenho (*round trip time* e *throughput*) como proposto em (Yi et al. 2012).

**A entrega dos dados na abordagem IP/*overlay***

No ambiente IP, a mensagem `RESPONSE` é encaminhada para o elemento que fez a requisição utilizando o endereço IP de origem da mensagem `REQUEST` que se torna endereço IP destino na mensagem `RESPONSE`.

### 3.2.4 Registro dos Conteúdos

Os CRs aprendem oportunisticamente sobre os conteúdos e qual a direção dos mesmos em termos de interfaces no tratamento de mensagens `RESPONSE`. Trata-se de um processo similar ao aprendizado do endereço MAC (*Media Access Control*) para Ethernet. Opcionalmente, os CRs podem aprender sobre os conteúdos pelo tratamento de mensagens `ANNOUNCE_CONTENT` enviadas pelos servidores, que são os provedores de conteúdo.

A Figura 3.7 apresenta o encaminhamento da mensagem `ANNOUNCE_CONTENT` na rede para um determinado *chunkId*. Os CRs encaminham para as interfaces disponíveis, exceto a interface de recepção. A tabela de roteamento de cada CR será atualizada no caso da mensagem indicar uma direção para o conteúdo contendo um menor número de *hops*. O encaminhamento da mensagem `ANNOUNCE_CONTENT` será feito pelos CRs por `N` *hops*, conforme a informação do campo *Neighbor Zones* presente na mensagem e atualizado (decrementado) a cada CR. A informação de número de *hops* além de indicar a distância, contribui para conter a inundação: a mensagem recebida com valor maior do que o valor atual da tabela será ignorada e não será encaminhada. O *loop* será contido pela informação do campo *Visited Neighbors* presente na mensagem e atualizado a cada CR que acrescenta sua própria identificação (*CRId*).

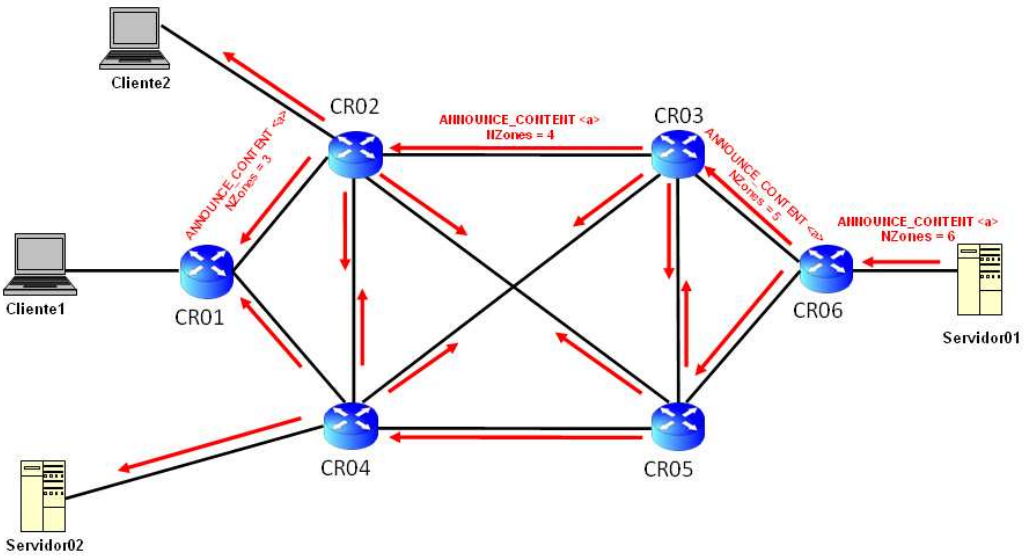



Figura 3.7: Encaminhamento do `ANNOUNCE_CONTENT` - Impacto do *flooding*

## 3.3 *Content Router* (CR)

O *Content Router* (CR) é um nó da rede responsável pelo mecanismo de roteamento que faz o tratamento das mensagens que trafegam na rede. Para a realização desta atividade, o CR mantém duas tabelas: a tabela de roteamento de conteúdo (ou tabela de vizinhança) e a tabela de *caching*, sendo as duas tabelas indexadas pelo *chunkId*.

As mensagens `REQUEST` são encaminhadas para os servidores ou provedores de conteúdo e respondidas diretamente pelo CR quando o *chunkId* é encontrado na sua tabela de *caching*, ou seja, quando ocorre *cache-hit*. Quando o CR não tem o conteúdo armazenado ocorre *cache-miss*. Neste caso o CR verifica se existe uma entrada na tabela de roteamento para o *chunkId* e faz o roteamento da mensagem conforme explicado anteriormente. Assim, o processo de encaminhamento considera a informação na tabela de roteamento e o parâmetro *Neighbor Zones*.

As mensagens `RESPONSE` são encaminhadas de volta aos clientes que originaram a requisição e é responsabilidade dos mesmos clientes requisitar conteúdos que não foram entregues ou que forem corrompidos. No tratamento da mensagem `RESPONSE` pode ocorrer o *caching* do conteúdo, em função do parâmetro *Caching Threshold*. Esse parâmetro é aplicado na função de probabilidade a fim de não sobrecarregar o cache pois o mesmo tem capacidade limitada.

Com relação ao *caching*, não existe nenhum tipo de sinalização entre os CRs, ou seja, eles não funcionam de forma cooperada. Assim, um mesmo conteúdo pode estar armazenado em todos os CRs no caminho feito por uma mensagem `RESPONSE`, exceto pelo uso do parâmetro *Caching Threshold*. Neste trabalho mantivemos o *caching* probabilístico da arquitetura de CRs original e também propusemos o Super CR, como uma especialização do CR na função de *caching*.

### 3.3.1 Super CR

Este trabalho introduz o Super CR como um CR especializado no armazenamento de informações sobre os conteúdo com a finalidade de reduzir a inundação de mensagens. Este tipo de CR é opcional e permite criar uma hierarquia na rede usando um plano de controle simples pela mensagem `ANNOUNCE_SCR`, que informa aos CRs vizinhos sua presença na rede conforme apresentado na Figura 3.8. O encaminhamento da mensagem `ANNOUNCE_SCR`, da mesma forma como no caso da mensagem `ANNOUNCE_CONTENT`, será feito pelos CRs por `N` *hops* conforme a informação do campo *Neighbor Zones* presente na mensagem e atualizado (decrementado) a cada CR. A mensagem não será encaminhada quando NZones atingir o valor zero. Os CRs mantêm a informação sobre o(s) Super CR(s) em lista apropriada.

Do ponto de vista de um CR, quando o conteúdo requisitado (*chunkId*) não está presente na sua tabela de roteamento de conteúdo, ou a variável NZones atingiu o valor zero, o processo de roteamento irá tentar o caminho pelo Super CR em lugar de iniciar o processo de *flooding*.

Qualquer CR pode funcionar como um Super CR e este pode ser um modo conveniente de obter vantagem da sua posição na rede para uma determinada topologia ou da sua disponibilidade de recursos. Nesse caso, o Super CR poderia, de forma proativa, após receber o anúncio de um conteúdo (1), proceder à requisição do conteúdo (2), receber e armazenar localmente (3) e em seguida anunciar o conteúdo a outros S_CRs (4). Este processo é apresentado na Figura 3.9.

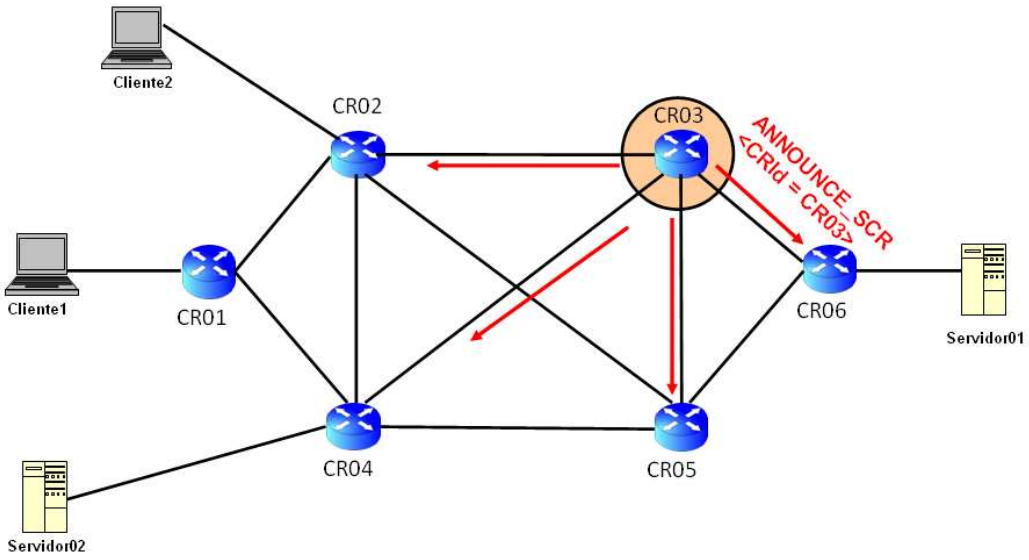



Figura 3.8: Encaminhamento do `ANNOUNCE_SCR`

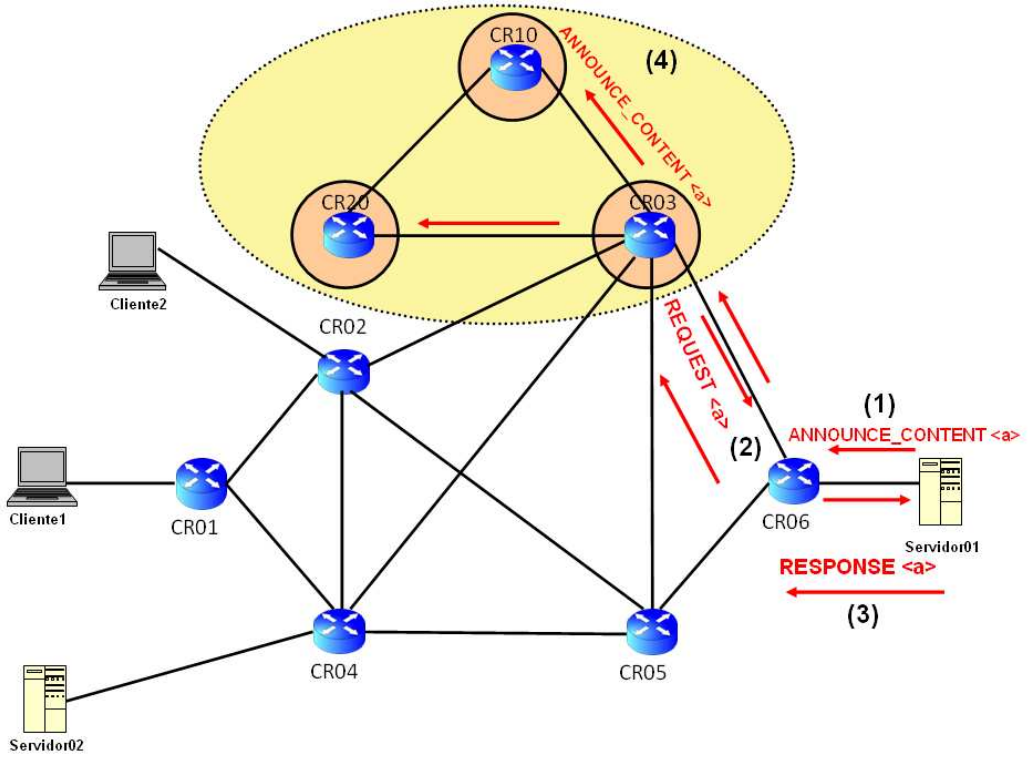



Figura 3.9: Encaminhamento do `ANNOUNCE_CONTENT` de forma proativa

Outras funcionalidades podem ser associadas ao Super CR:

- **Gateway:** o Super CR pode ser especializado e assumir uma função de *gateway* fazendo a conexão de uma rede de conteúdos no nível de enlace com uma rede IP. Também pode ser ponto de acesso inter-domínios.
- **Rendezvous:** o Super CR pode ser especializado e ser um ponto de *rendezvous* como

no modelo *publish/subscriber*. Ao receber anúncio de novo conteúdo via mensagem ANNOUNCE_CONTENT, verifica se existe requisições pendentes e procede ao encaminhamento das mesmas. Neste caso, a disponibilidade de recursos para manter requisições não atendidas precisa ser melhor avaliada.

- **Servidor de Resolução de Nomes:** o Super CR pode ser especializado e assumir a função de servidor para resolução de nomes, replicando o *Metadata* associado a um conteúdo. Nesse caso, os servidores, ou provedores de conteúdo, encaminham o registro de *Metadatas* e os clientes recuperam os mesmos para proceder à requisição dos *chunkIds*. A Figura 3.10 apresenta uma possibilidade da obtenção do metadata com o Super CR atuando como servidor de nomes.

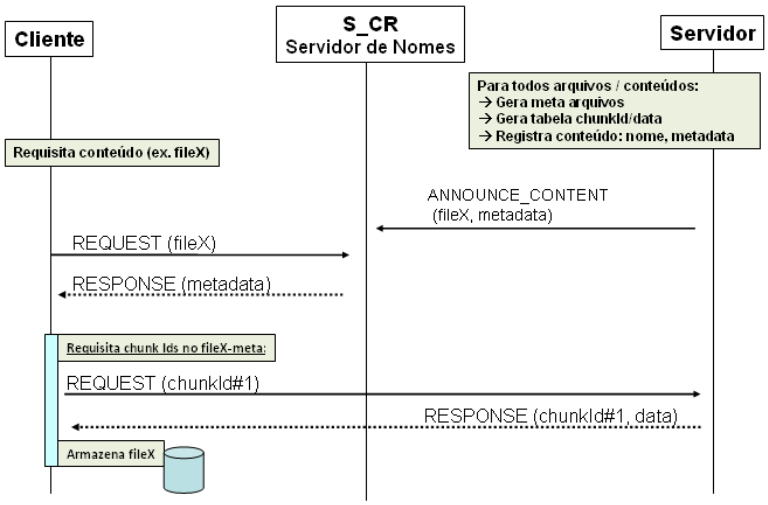



Figura 3.10: Super CR: Recuperação do Metadata via Servidor de Nomes

## 3.4 Conclusão do Capítulo

Este capítulo apresentou a proposta da arquitetura dos CRs com roteamento de conteúdo diretamente no nível de enlace diferentemente de outras propostas que utilizam uma camada intermediária: camada de estratégia no CCN e camada de convergência na NetInf.

Utilizando os conceitos da arquitetura de CRs, a nova proposta dá prioridade ao conteúdo tornando-o o elemento mais importante na rede e introduz novos mecanismos para roteamento no conteúdo como a lista de requisições pendentes e a indicação da menor distância em número de *hops* na tabela de roteamento.

Capítulo 4

# Implementação e Resultados

Este capítulo apresenta a implementação do protótipo utilizado como instrumento de validação da arquitetura proposta. Também descreve a metodologia de validação e, por fim, apresenta e discute os resultados obtidos nos experimentos efetuados com o protótipo.

A implementação apresentada neste trabalho parte de uma versão inicial do protótipo para ambiente IP/*overlay* com foco em *caching*. Este trabalho implementa algumas funcionalidades da proposta original (Wong et al. 2011): controle de *loop* pelo uso e verificação do campo BFvisited, inclusão do campo lastCR, e busca nos CRs vizinhos controlada pelo campo NZones. Além disso, acrescenta um módulo para permitir a operação no nível de enlace sem dependência da rede IP e implementa as funcionalidades propostas para a arquitetura no nível de enlace: a lista de requisições pendentes, o Super CR, as mensagens `ANNOUNCE_CONTENT` e `ANNOUNCE_SCR` e o conceito da menor distância em número de *hops*.

## 4.1 Implementação

O protótipo foi concebido com objetivo de validar a arquitetura de CRs procurando facilitar a adaptação e a inserção de novas funcionalidades. Desenvolvido em Linguagem C para ambiente Linux, o protótipo implementa as funções dos elementos da arquitetura apresentados anteriormente: *Content Router* (CR), Cliente e Servidor.

O protótipo será explicado a partir de uma abordagem *"top-down"* privilegiando a abordagem no nível de enlace. Algumas diferenças da implementação para o nível IP/*overlay* estão indicadas no final do capítulo.

O Diagrama de Implantação da Figura 4.1 apresenta os elementos da arquitetura no nível de enlace em tempo de execução, assim como os vínculos de comunicação entre eles, utilizando uma visão de alto nível.

O *Content Router* (CR) é o elemento principal da arquitetura e está constantemente ativo à espera de mensagens. O Servidor é o elemento que tem a responsabilidade de disponibilizar os conteúdos que são requisitados e o Cliente é o elemento que requisita um determinado conteúdo. Os conteúdos são arquivos de tipos diversos (pdf, mp3, jpg etc) que serão particionados em *data chunks* (1350 bytes) aos quais são atribuídos identificadores, os *chunkIds*, gerados pelo aplicativo genmd ou pelo servidor (server-eth). Tais conteúdos são disponibilizados nos servidores.

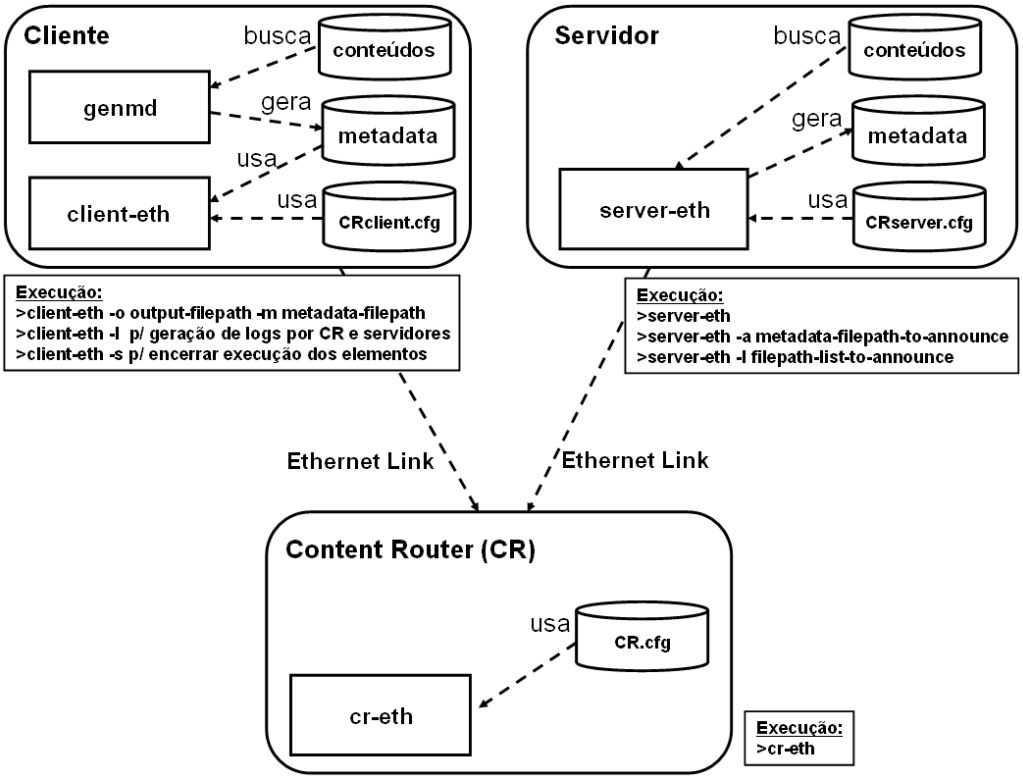



Figura 4.1: Diagrama de Implantação para nível de enlace

Basicamente, o funcionamento dos elementos do protótipo é conforme o seguinte processo: o Cliente requisita conteúdos a partir do seu identificador (*chunkId*) através das mensagens `REQUEST` e aguarda os dados através das mensagens `RESPONSE`; re-envia a requisição no caso de *timeout*. O Servidor espera por solicitação de conteúdos e responde com dados através da mensagem `RESPONSE` quando tem o conteúdo. Os CRs interceptam as mensagens em trânsito na rede e processam as mesmas conforme as informações presentes nas suas estruturas de dados (tabelas de roteamento e *caching*, lista de requisições pendentes, etc).

A implementação atual não inclui mecanismo de resolução de nomes e o Cliente é iniciado com o metadado dos conteúdos (lista dos *chunkIds*) para proceder às requisições conforme mostra a Figura 4.2.

A implementação atual não inclui um mecanismo para a verificação e autenticação dos conteúdos (*chunks*) recebidos pelo Cliente que somente verifica se o (*chunkId*) na mensagem `RESPONSE` é o mesmo que foi requisitado na mensagem `REQUEST`. Este mecanismo de verificação e autenticação é discutido em (Wong 2011).

### 4.1.1 *Content Router* (CR)

O *Content Router* (CR) é executado a partir do módulo **cr-eth**. Funcionalmente, o CR é composto por um módulo de controle responsável pela comunicação e um módulo que trata da manipulação das mensagens, além de estruturas de dados como: tabela de *caching*, tabela de roteamento de conteúdos, lista de interfaces, lista de requisições pendentes e lista de Super CRs.

Ao iniciar a execução, o CR verifica quais interfaces estão ativas (ou disponíveis) e compõe

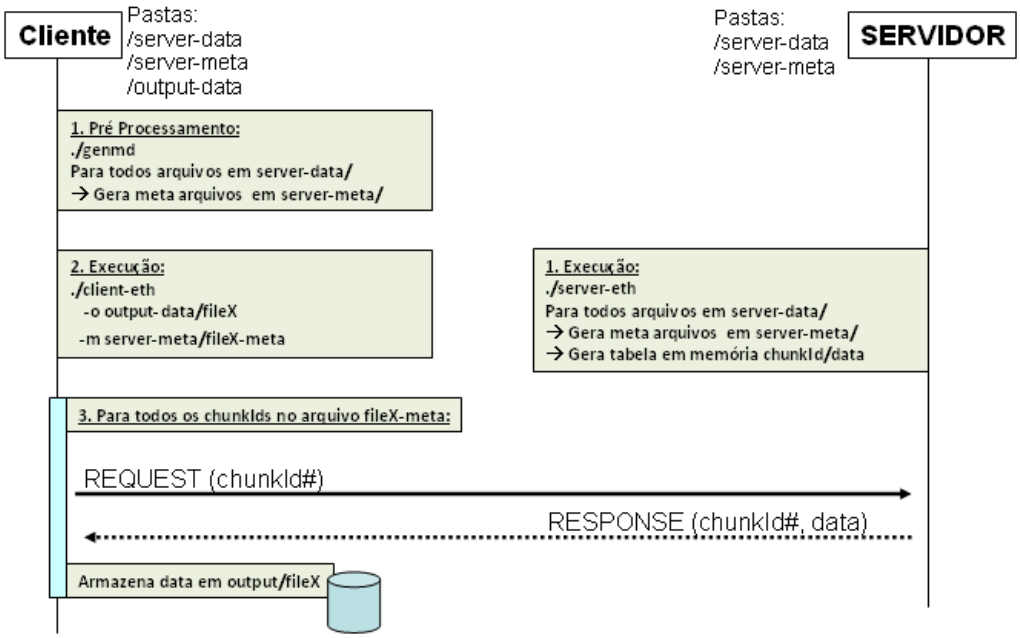



Figura 4.2: Geração do Metadata no Cliente e Servidor

a sua lista de interfaces. O CR é identificado pelo endereço MAC associado à primeira interface ativa. A esta identificação será aplicada uma função de *hash* gerando o CRId para manter compatibilidade com a versão IP que utiliza o endereço IP do CR para gerar o CRId.

A Figura 4.3 apresenta uma visão geral dos módulos do CR indicando os que são comuns (ou reutilizados) na implementação das abordagens no nível IP e nível de enlace.

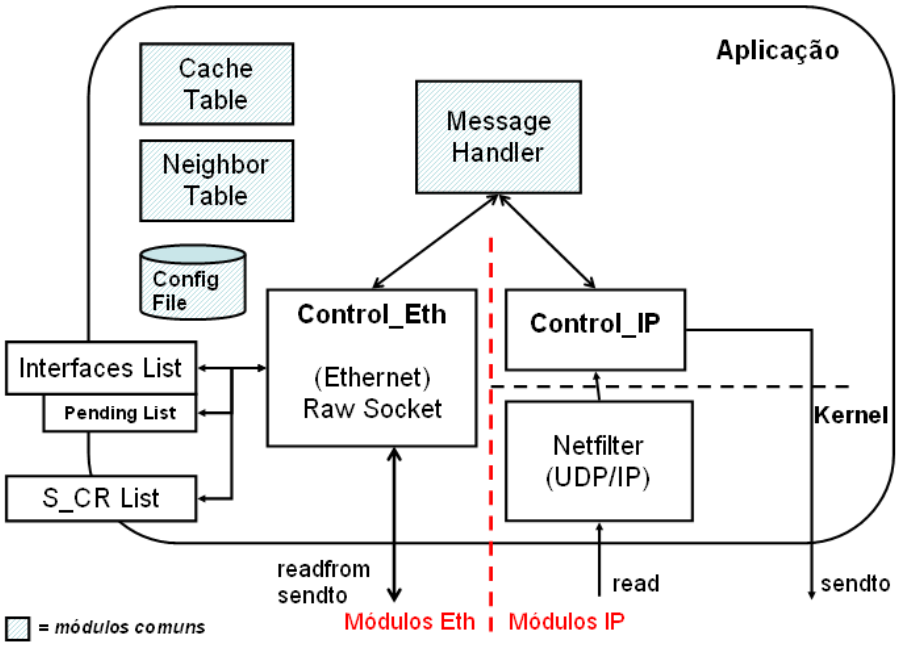



Figura 4.3: Módulos do CR

### Estruturas de Dados

O CR mantém as seguintes estruturas de dados:

- *Cache Table* ou tabela de *caching*: contém os dados ou *Data Chunks*; é indexada pelo identificador do conteúdo (*chunkId*); possui capacidade limitada; utiliza política *First In First Out* (FIFO) como estratégia de substituição. Cada entrada da tabela possui as seguintes informações: *chunkId*, tamanho do dado, o dado (conteúdo ou *Data Chunk*). O total de entradas na tabela é limitado e definido no arquivo common.h.

- *Neighbor Table* ou tabela de roteamento: mantém a melhor interface para atingir o *chunkId*; é indexada pelo identificador do conteúdo (*chunkId*); possui capacidade limitada; utiliza política *First In First Out* (FIFO) como estratégia de substituição; o valor da melhor interface pode ser alterado em função da informação do número de *hops* presente nas mensagens `RESPONSE` ou `ANNOUNCE_CONTENT`. Cada entrada da tabela possui as seguintes informações: *chunkId*, número de *hops*, tamanho do dado, o dado (interface). O total de entradas na tabela é limitado e definido no arquivo common.h.

No nível de enlace o CR mantém ainda:

- lista das interfaces disponíveis: obtida com a execução da função ioctl() tendo o argumento *request* com valor SIOCGIFCONF; funciona de forma similar ao comando `ifconfig` do Linux. As interfaces correspondem aos dispositivos Ethernet reconhecidos pelo sistema (ex. eth0, eth1). Cada entrada da lista possui as seguintes informações: nome da interface(ex. eth0), índice da interface (correspondente ao dispositivo), endereço MAC, estado (ATIVA/INATIVA).

- lista de requisições pendentes: está associada a uma determinada interface e mantém os *chunkIds* requisitados via mensagem `REQUEST` recebidos pela mesma interface.

- lista de Super CRs ou *S_CR List*: mantém a lista de CRIds anunciados pela mensagem `ANNOUNCE_SCR` com a melhor interface associada. Cada entrada da lista possui as seguintes informações: *CRId*, número de *hops*, interface.

A Figura 4.4 apresenta uma visão das estruturas de dados do CR indicando os arquivos de código fonte C onde as mesmas são implementadas.

### Configuração de Variáveis

O protótipo do CR utiliza um arquivo de configuração como forma de facilitar a variação de valores associados às variáveis do programa conforme mostra a Figura 4.4. As seguintes variáveis de configuração são definidas para o CR:

- CRId: endereço IP do CR (usado somente na abordagem IP/overlay)

- *Flooding Flag*: indica o comportamento do roteamento de requisições quando NZones atinge o valor zero. Assume valor 0 ou 1.

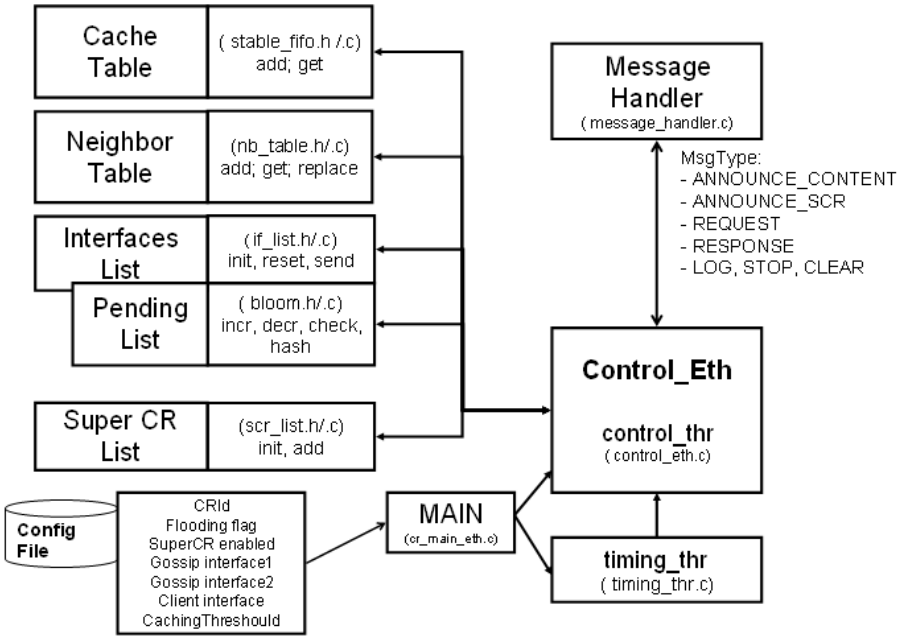



Figura 4.4: Estruturas de dados do CR no nível de enlace

- *SuperCR enabled*: indica se o CR vai se comportar como Super CR. Assume valor 0 ou 1.
- *Gossip interface*1 e 2: identificação de interface utilizada para envio de mensagens de LOG (detalhes em 4.2).
- *Client interface*: identificação de interface ligada ao Cliente quando CR e Cliente executam no mesmo elemento.
- *Caching Threshold*: valor para função de probabilidade. Pode assumir valores entre 0 a 1.

### Lista de Requisições Pendentes

Esta estrutura do CR no nível de enlace mantém o estado da comunicação com relação aos *chunkIds* requisitados via mensagem `REQUEST` recebido por uma determinada interface. Os mesmos serão removidos da lista quando da recepção da mensagem `RESPONSE` correspondente. Cada interface do CR tem a sua lista de pendências associada. O CR verifica as listas de todas as interfaces na recepção das mensagens `REQUEST` e `RESPONSE`.

A primeira implementação utilizou uma lista simples para armazenar os valores dos *chunkIds*. Este tipo de estrutura é suficiente para uma rede de pequena escala. Num segundo momento, como forma de atender requisitos de escalabilidade, a lista foi implementada como uma estrutura de dados comprimida utilizando filtros de Bloom (Tarkoma et al. 2012).

Os filtros de Bloom são estruturas de dados probabilísticas e eficientes em termos de espaço. São utilizados para verificar se um dado elemento é membro de um conjunto. Esta verificação é livre de falsos negativos, mas não de falsos positivos, que podem ocorrer com uma dada probabilidade. Um filtro de Bloom é basicamente uma sequência de bits inicializados com o valor zero. Para armazenar um elemento no filtro de Bloom, aplica-se a este funções do tipo

*hash*, sendo que cada função resultará em uma posição diferente de bit dentro da sequência. Cada bit apontado deverá ter seu valor alterado para 1.

No caso da lista de requisições pendentes optou-se por um filtro de Bloom contador (Fan, Cao, Almeida & Broder 2000). Este tipo de filtro utiliza um contador no lugar de um bit, o que torna possível implementar a operação de remoção no conjunto representado pela estrutura. Os bits apontados pelas funções de *hash* indicarão quais contadores serão incrementados no `REQUEST` ou decrementados no `RESPONSE`. Desta forma é possível manter uma estrutura compacta, de tamanho fixo para controlar as pendências de cada interface.

No protótipo são utilizadas três funções de *hash* aplicadas aos *chunkIds*: SAX (Ramakrishna & Zobel 1997), SDBM (Jain & Pandey 2012) e Bernstein (Bernstein 2009). Outros trabalhos utilizam as mesmas funções aplicadas a filtros de Bloom (Pal, Sardana & Yadav 2012) e (Qwasmi & Liscano 2013).

### Módulos de Controle e Comunicação

O módulo de controle é especializado para atender nível de enlace e nível IP fazendo a recepção e o envio de mensagens, além de inspecionar o cabeçalho das mesmas.

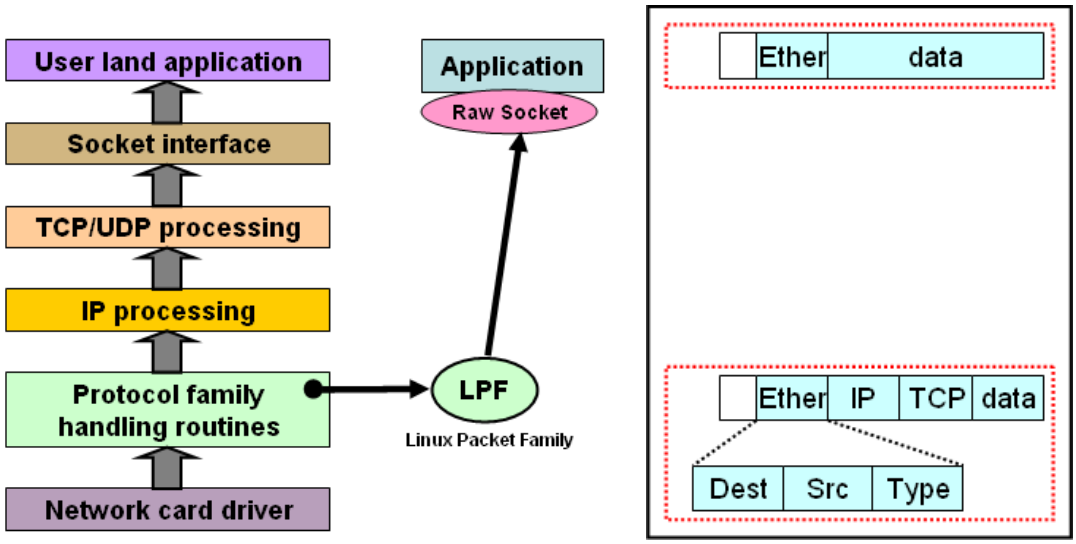
No nível de enlace, o CR utiliza *Raw Sockets*. Em geral, os pacotes trafegam nas camadas de rede de forma que cada camada retira o cabeçalho correspondente. Assim, somente os dados são entregues à camada de aplicação. Os *Raw Sockets* são uma forma de entregar um pacote diretamente na camada de aplicação com todos os cabeçalhos relativos às camadas inferiores que podem ser analisados e da mesma forma a aplicação pode gerar um pacote e enviar à rede.

No protótipo, os *Raw Sockets* são criados através da interface PF_PACKET que é uma interface de software do Linux para receber e enviar pacotes no nível 2 da camada OSI (*Open System Interconnection*). Todos os pacotes são recebidos completos (cabeçalho e dados). Todos os pacotes enviados são transmitidos sem modificações pelo *kernel*. Uma identificação para o protocolo do CR (0x9E9E) foi utilizada, assim todos os pacotes Ethernet deste tipo serão repassados para a aplicação. A identificação do protocolo ocorre no campo *Length/Type* do pacote Ethernet, considerando que valores entre 0 e 1500 são entendidos como *Length* e valores acima de 1500 são entendidos como *Type*, ou seja, identificam um protocolo da camada de rede[1]. O endereço de destino utilizado é sempre *broadcast*.

A Figura 4.5 apresenta o caminho do pacote feito com o uso dos *Raw Sockets* e a estrutura do pacote entregue para a aplicação.

No CR foi acrescentado um módulo de temporização para periodicamente enviar a mensagem `ALIVE`. Este módulo executa como uma *thread* separada e se comunica com o módulo principal através de *socket*. Ao receber a mensagem `ALIVE`, o CR pode então tomar ações como enviar a mensagem `ANNOUNCE_SCR` ou algum outro procedimento. A mensagem `ALIVE` não faz parte do protocolo da rede de conteúdos e tem somente uma função operacional.

[1] http://www.iana.org/assignments/ethernet-numbers




(a) Caminho do pacote (b) Estrutura da informação

Figura 4.5: *Raw Sockets*

### Módulo de Tratamento de Mensagens

O módulo de manipulação das mensagens (message_handler) faz o tratamento específico de cada tipo de mensagem conforme indicado no cabeçalho. Além disso, verifica e armazena conteúdos na tabela de *caching*, verifica e armazena informação sobre interfaces na tabela de roteamento e responde às requisições relativas aos conteúdos por ele armazenado.

O comportamento do CR quando da recepção das mensagens `REQUEST` é apresentado pelo Diagrama de Atividade na Figura 4.6. Além da variável Neighbor Zones (NZones), o *Flooding Flag* implementa o comportamento adicional quando NZones atinge o valor zero: se *Flooding Flag* é zero, o processo de busca pára; se *Flooding Flag* é um, o processo de busca continua com inundação. O valor do *Flooding Flag* é um parâmetro de configuração do CR.

Ao receber a mensagem `RESPONSE`, o CR encaminha o dado para todas as interfaces que têm o *chunkId* na lista de pendencias e remove o mesmo *chunkId* das listas. O comportamento do CR, quando da recepção das mensagens `RESPONSE`, é apresentado pelo Diagrama de Atividade na Figura 4.7.

No cabeçalho da mensagem foi introduzido o campo nhToCtt, número de *hops* para encontrar o conteúdo. O objetivo é guardar o valor do campo numHops quando a mensagem `RESPONSE` é gerada para fins de estudo do comportamento do protótipo.

A mensagem `ANNOUNCE_SCR` é gerada no início da execução do CR quando o mesmo está configurado como Super CR (variável de configuração).

Outras mensagens foram acrescentadas ao protótipo com finalidades diversas: `LOG`, `STOP`, e `ALIVE`.

## 4.1.2 Cliente

Este elemento é executado a partir dos módulos:

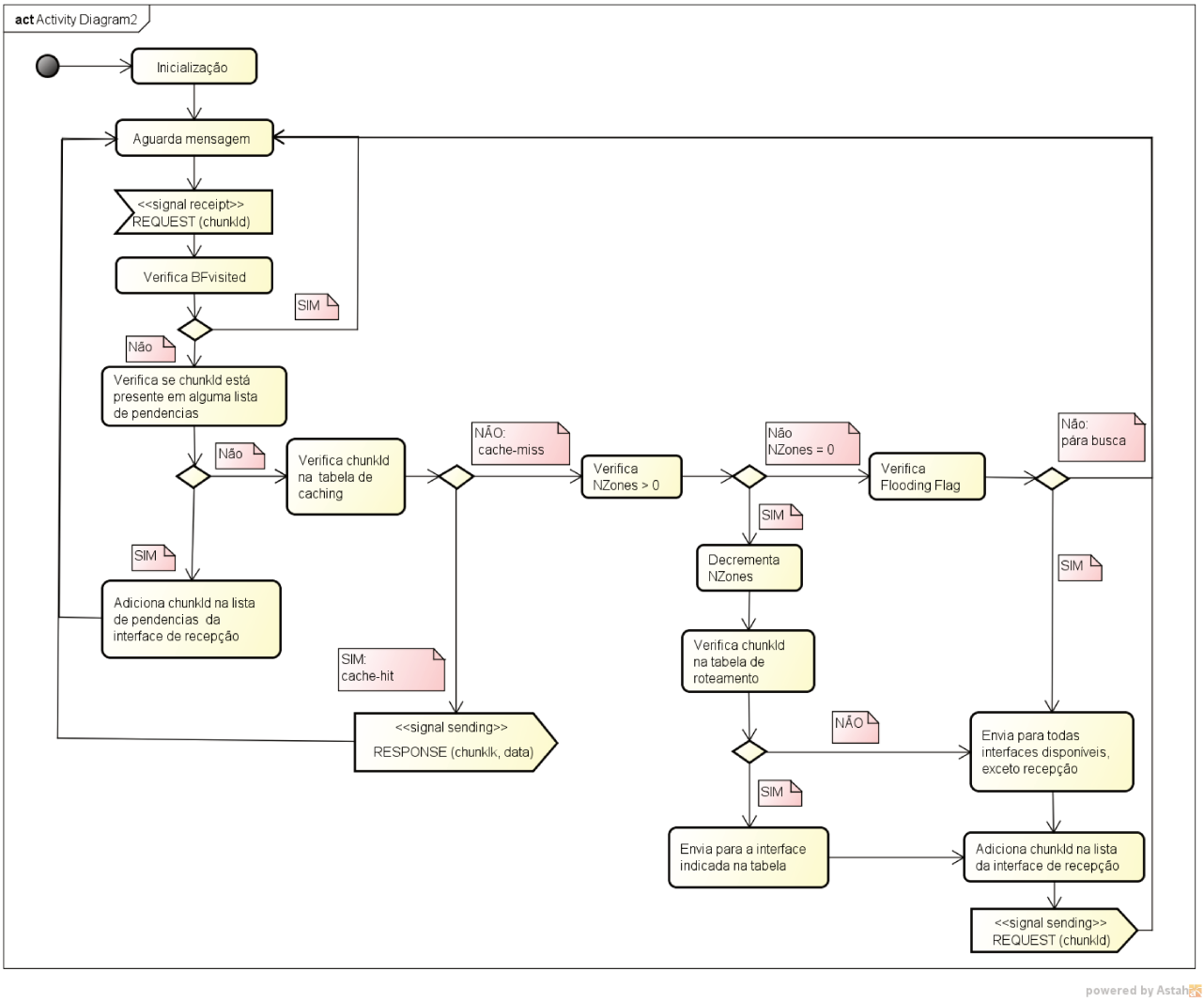



Figura 4.6: CR - Tratamento da mensagem `REQUEST`

- **genmd:** módulo responsável pela geração do metadata; processo manual pois não está disponível o elemento para a resolução de nomes.
- **client-eth:** módulo principal no nível de enlace.

O módulo principal identifica as interfaces disponíveis, valida os dados de entrada e faz a requisição de forma síncrona dos *chunkIds* relativos ao conteúdo solicitado, gerando mensagens `REQUEST` e aguardando pela mensagem `RESPONSE` correspondente. Existe uma temporização associada ao envio da mensagem `REQUEST`. A ocorrência de *timeout* significa que o conteúdo não foi recebido e a mensagem `REQUEST` será reenviada pelo Cliente por um número limitado de tentativas. Os dados recebidos são armazenados no arquivo de saída por uma *thread* conforme acontece a recepção das mensagens `RESPONSE`. A execução do elemento Cliente corresponde à requisição de um arquivo ou conteúdo através do envio de mensagens `REQUEST` para os *chunkIds* que compõem aquele conteúdo; a execução é finalizada após a recepção de todo o conteúdo, ou quando atingir o limite máximo de tentativas em consequência de *timeouts*.

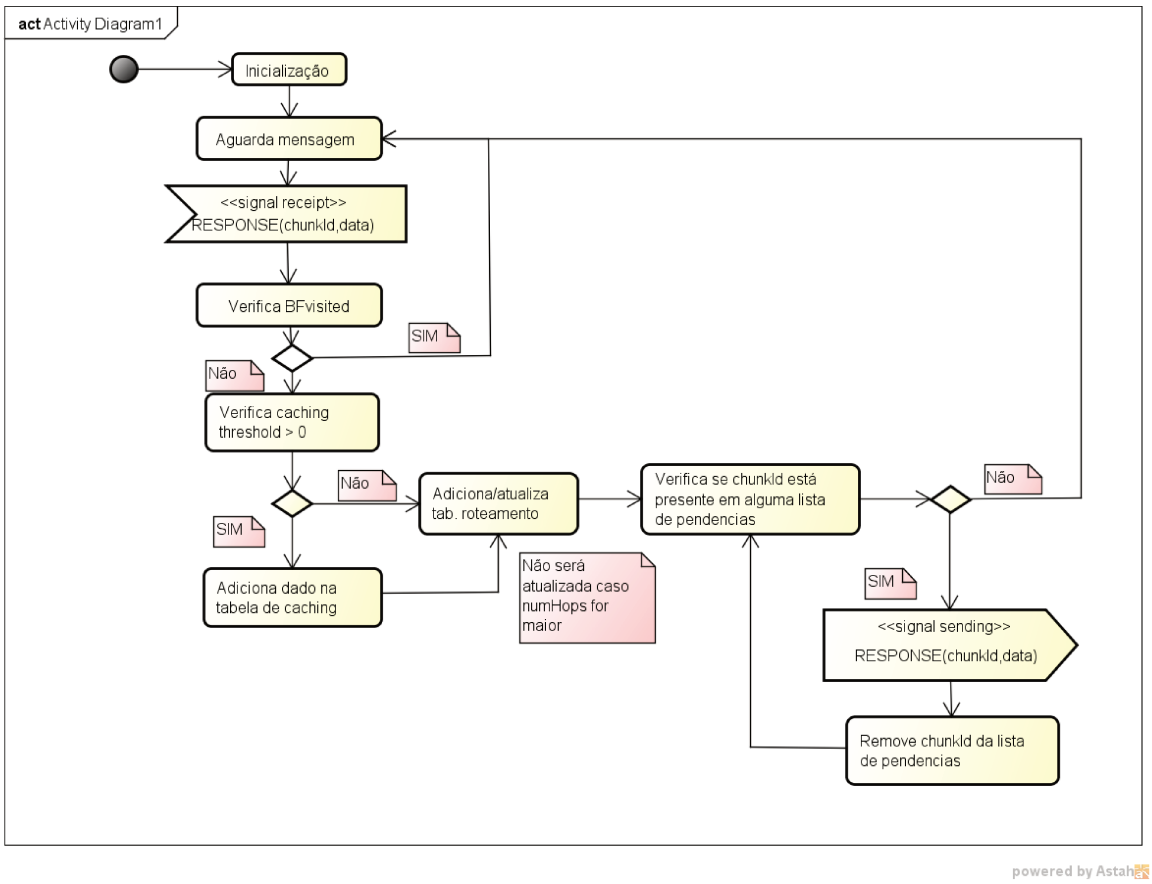



Figura 4.7: CR - Tratamento da mensagem `RESPONSE`

A Figura 4.8 dá uma visão geral dos módulos do Cliente indicando o que é comum na implementação das abordagens no nível IP e nível de enlace.

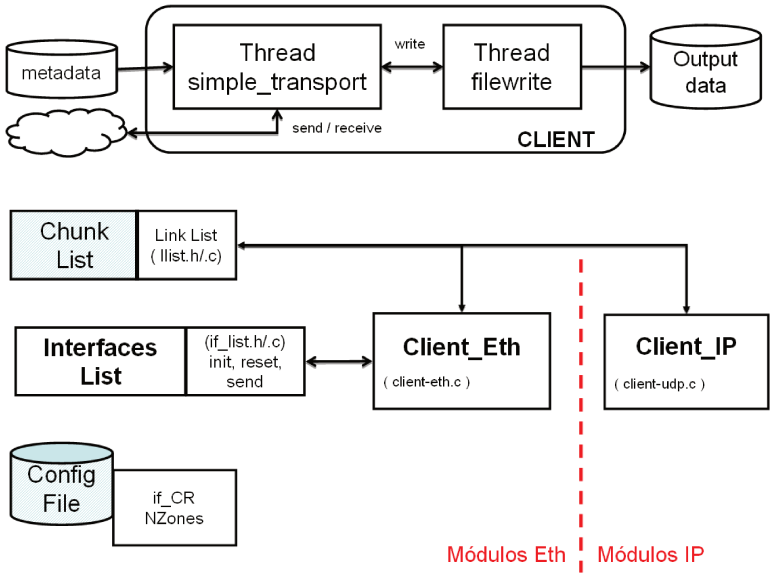



Figura 4.8: Módulos do Cliente

O elemento Cliente é utilizado para o envio de mensagens de `LOG` aos CRs e Servidores da rede para que os elementos armazenem os contadores utilizados para verificações diversas. Como indicado na Figura 4.1, existem opções de execução para `LOG` (opção -l) e para forçar a finalização dos CRs e servidores (opção -s), através do envio da mensagem `STOP`.

As seguintes variáveis de configuração são definidas para o Cliente: if_CR, que é a interface de conexão com um CR; e NZones, valor que será utilizado no processo de encaminhamento da mensagem `REQUEST`.

### 4.1.3 Servidor

Este elemento é executado a partir do módulo **server-eth**, módulo principal no nível de enlace que identifica as interfaces disponíveis, gera uma tabela em memória com todo o conteúdo disponível indexada pelo *chunkId*; aguarda por mensagens `REQUEST` e gera mensagem `RESPONSE` correspondente quando tem o conteúdo disponível. Mensagens de anúncio de um conteúdo, ou seja, de todos os seus *chunkIds*, serão geradas se a execução indicar o nome do conteúdo como argumento na execução do servidor.

A Figura 4.9 dá uma visão geral dos módulos do Servidor indicando o que é comum na implementação das abordagens no nível IP e nível de enlace.

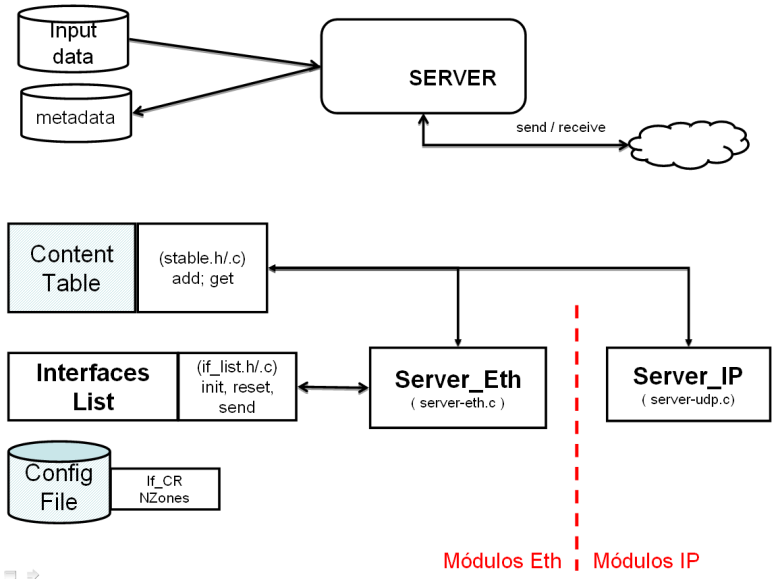



Figura 4.9: Módulos do Servidor

As seguintes variáveis de configuração são definidas para o Servidor: if_CR, que é a interface de conexão com um CR; e NZones, valor que será utilizado no processo de encaminhamento da mensagem `ANNOUNCE_CONTENT`.

### 4.1.4 O protótipo na abordagem IP/*overlay*

O protótipo apresenta algumas diferenças na implementação para o ambiente IP/*Overlay*. A Figura 4.10 apresenta os elementos da arquitetura no nível IP/*overlay*, ou simplesmente IP

em tempo de execução, assim como os vínculos de comunicação entre eles, utilizando uma visão de alto nível.

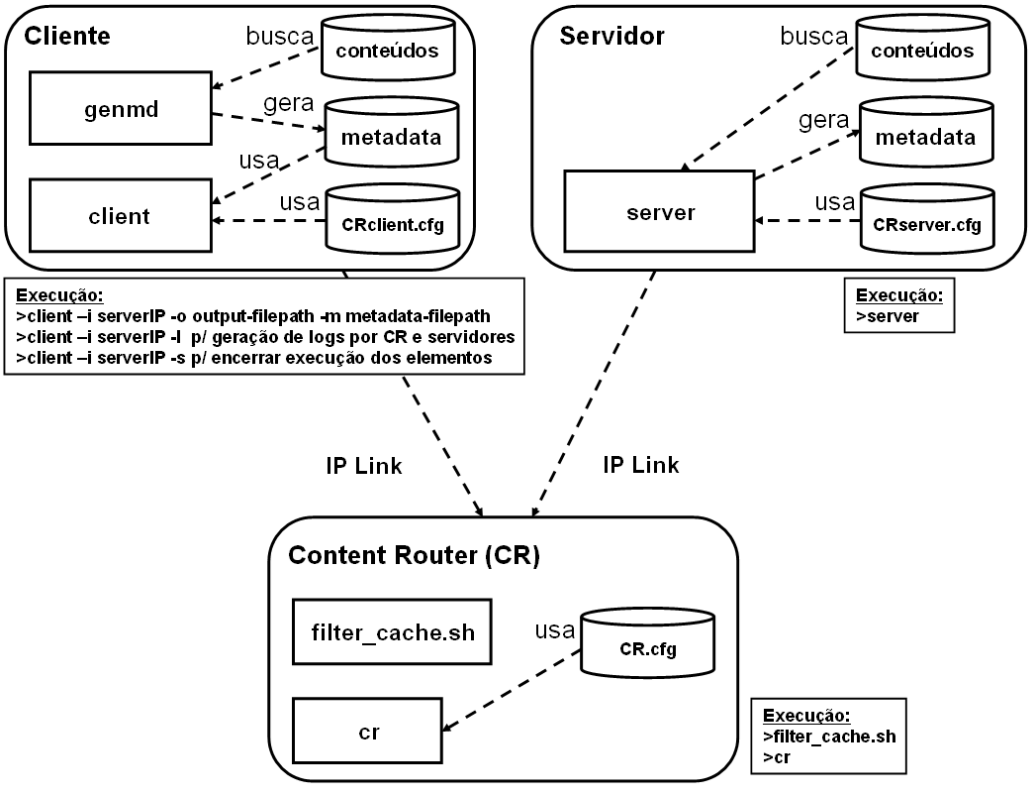



Figura 4.10: Diagrama de Implantação para nível IP/*overlay*

O *Content Router* (CR) no nível IP/*overlay* tem dois módulos de execução:

- **cr-ip:** módulo principal no nível IP/*overlay*
- **filter_cache.sh:** módulo auxiliar no ambiente IP/*overlay*, pré-requisito para o módulo principal pois define regras que fazem com que pacotes UDP recebidos pelos portas 22000 e 22001 sejam direcionados para a aplicação (módulo netfilter)

Cada CR é identificado pelo seu endereço IP indicado em arquivo de configuração. O módulo de controle atual não aplica a função de *hash* para gerar um CRId, e também não utiliza ou verifica o campo BFvisited do cabeçalho das mensagens.

Como apresentado na Figura 4.3, no nível do IP o módulo de controle é composto também pelo netfilter (kernel), responsável por interceptar mensagens na rede e entregar ao módulo de controle, utilizando interface Netlink, datagrama UDP e portas 22000 e 22001. O netfilter executa como uma *thread* separada e se comunica com o módulo principal através de *socket*.

Na tabela de roteamento, a melhor rota é indicada pelo endereço IP de um CR vizinho que encaminhou uma mensagem `RESPONSE`. Esta informação é extraída do campo lastCR no cabeçalho da mensagem implementada por este trabalho.

No tratamento da mensagem `REQUEST`, quando NZones atinge o valor zero, a mensagem é encaminhada ao endereço IP de destino, indicado no cabeçalho da mensagem. A mensagem `RESPONSE` é encaminhada ao endereço IP que originou a mensagem `REQUEST`.

## 4.2 Validação

Este trabalho utiliza método de pesquisa quantitativa e procedimento do tipo experimento (Creswell 2009). Participam do experimento todos os elementos da rede: clientes, servidores e roteadores de conteúdo (CRs). Todos os participantes são designados por conveniência, ou seja, não existe nenhum mecanismo aleatório na seleção dos mesmos. Estes elementos interligados formam as topologias utilizadas para os testes.

Como variáveis independentes, ou seja, as que causam ou influenciam nos resultados, temos as seguintes variáveis já explicadas anteriormente:

- *caching threshold*: relacionado ao armazenamento de conteúdo
- *neighbor zones*: relacionado ao processo de encaminhamento de mensagens
- *flooding flag*: relacionado ao mecanismo de inundação da requisição para o CR no nível de enlace

Além destas variáveis, colaboram na consequência dos resultados a dimensão das tabelas de *caching* e de roteamento. Diferentes valores são aplicados a estas variáveis para verificação do comportamento da arquitetura, utilizando o protótipo.

As variáveis dependentes, ou de resultado, indicam as consequências. Os valores correspondentes são coletados durante os experimentos através de contadores implementados nos elementos da arquitetura. São elas:

- utilização dos servidores e roteadores: quantidade de mensagens recebidas ou enviadas nos diferentes elementos
- distância para a resolução de um conteúdo em número de *hops*
- tempo de transferência do conteúdo (ou tempo de *download* de um arquivo)

O protótipo é usado como instrumento no experimento. Mensagens especiais foram acrescentadas ao protótipo para observação do comportamento da rede através de medidas ou contadores (ex. total de mensagens recebidas, enviadas, *cache-hit*, *cache-miss* etc) Todos os elementos da arquitetura geram arquivos com registro de eventos (arquivos de LOG) e arquivos com as medidas no formato CSV (*comma separated value*) quando da recepção destas mensagens (ex: `LOG`, `STOP`). Esses arquivos são gerados, em geral, ao final de uma solicitação de um conteúdo e ao final do teste. As mensagens têm função operacional sem relação com o protocolo da rede, nunca ocorrem durante a transferência de um arquivo. Como a topologia e as interfaces de cada CR são conhecidas, o envio destas mensagens é feito de forma direcionada para interfaces identificadas como interfaces *gossip* (Kulik, Heinzelman & Balakrishnan 2002) pré-definidas em cada CR garantindo que todos os elementos recebam a mensagem sem ocorrência de *flooding*.

Para a avaliação do protótipo são consideradas duas topologias: (1) topologia em malha, ou *Internet-like*, onde doze CRs estão inter conectados num ambiente de máquinas virtuais, e (2) topologia simples, com um único CR atuando como ponto de acesso e *gateway* Internet num ambiente de máquinas reais. A topologia é similar a uma rede domiciliar (*home network*).

A condição inicial em todos os cenários é que os CRs não têm qualquer informação sobre os conteúdos disponíveis, ou seja, as tabelas de *caching* e roteamento estão vazias ao iniciar o experimento.

### 4.2.1 Testes no Ambiente Virtual

A seguir são descritos os cenários e procedimentos de teste no ambiente virtual que tiveram o objetivo de comparar as duas abordagens: nível de enlace e nível IP/*overlay*. Neste cenário também foram executados experimentos para verificação do comportamento da arquitetura em condições variadas, validação das novas primitivas e do elemento Super CR.

#### Cenário de Teste

Neste cenário de teste, os elementos da rede são um conjunto de máquinas virtuais criadas utilizando a ferramenta XEN [2], que permite a definição de máquinas que atuem como roteadores. Todos os elementos são executados em ambiente Debian GNU/Linux. Para os testes da rede em modo *overlay*/IP, os roteadores executam a pilha de protocolos *open-source* Quagga [3] com o protocolo OSPF (*Open Shortest Path First*).

Uma topologia que inclui 12 CRs, 4 servidores e 8 clientes é utilizada para os testes da arquitetura de CRs nos ambientes IP e nível de enlace, ou Ethernet (Figura 4.11). Os elementos (*hosts*) estão conectados através de dispositivos Ethernet não capturados na figura por questão de simplicidade. Os CRs formam dois domínios interligados em dois pontos. Esta topologia coloca a abordagem no nível de enlace em certa desvantagem comparada ao IP, pois introduz mais pontos potenciais de inundação. No entanto a topologia serve para a comparação entre as duas abordagens.

#### Procedimento de Teste

Um teste é composto de uma sequência de 30 solicitações para diferentes arquivos ou conteúdos (texto, som, imagem), seguindo uma distribuição *power law*: 10 arquivos são solicitados sendo que cada um deles tem um grau de popularidade, ou seja, alguns são solicitados com maior frequência. A distribuição de popularidade por conteúdo é resultado de programa que gera número aleatório e o *script* de execução do teste. A Tabela 4.1 mostra a lista dos arquivos com respectiva quantidade de *chunks* (1 *chunk* = 1350 bytes) e quantas vezes cada arquivo é requisitado no teste.

Os experimentos ou testes são repetidos utilizando a mesma sequência, com variação de valores das variáveis independentes descritas anteriormente: *caching threshold*, *neighbor zones* e *flooding flag*. Os valores são atribuídos através de arquivo de configuração. As tabelas de *caching* e roteamento são configuradas para suportar (i)10.000 e (ii)25.000 entradas. Isso significa que na situação (i) ocorre substituição nas duas tabelas considerando o total de *chunks* durante o teste, ou seja, configura uma situação de limitação no tamanho das tabelas.

[2] http://www.xensource.com
[3] http://www.quagga.org

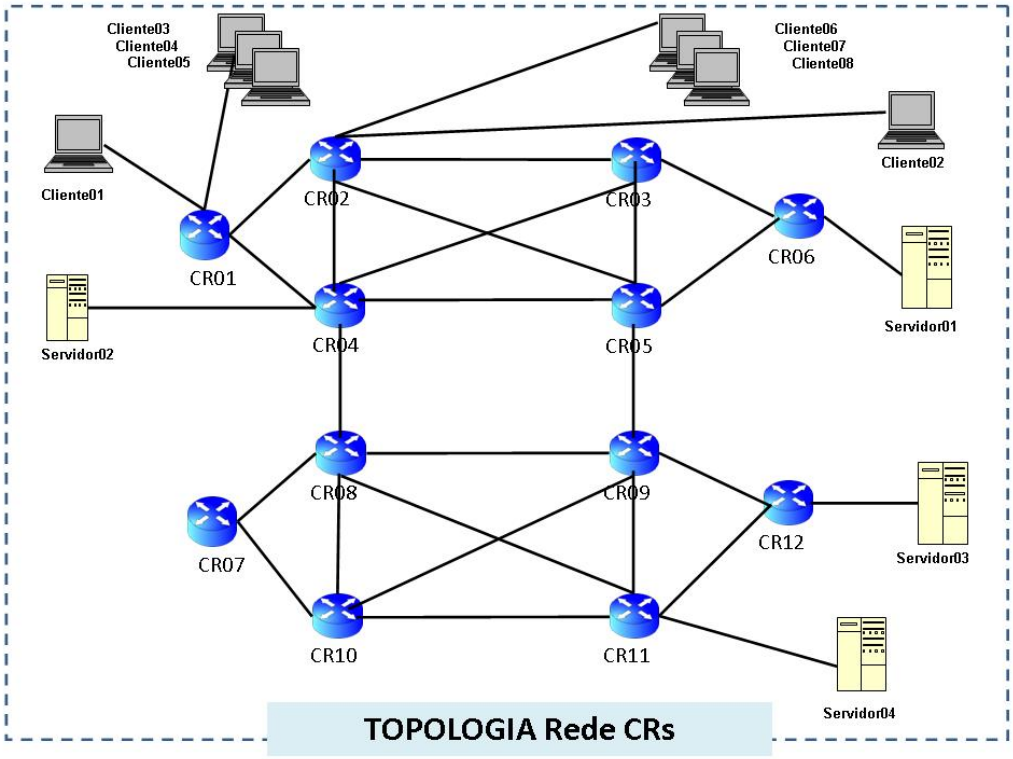



Figura 4.11: Topologia - Ambiente virtual

Tabela 4.1: Arquivos de conteúdo para procedimento de testes.

| # | **Nome do arquivo** | **Qtde. chunks** | **# req.** |
|---|---|---|---|
| 1 | aliancas.jpg | 1059 | 0 |
| 2 | biblioteca.jpg | 563 | 1 |
| 3 | DistributedCachingAlgoritms.pdf | 175 | 2 |
| 4 | EnhanceContentBroadcast.pdf | 118 | 1 |
| 5 | GatewayControlledContentCaching.pdf | 286 | 2 |
| 6 | PacktpubMoodleSecurity.pdf | 3406 | 16 |
| 7 | SantaClausIsComingToTown.mp3 | 5099 | 2 |
| 8 | TeachYourselfFreeBSD.pdf | 7176 | 1 |
| 9 | TikosGrooveFeatGosha.mp3 | 6590 | 1 |
| 10 | towerEiffel.jpg | 411 | 4 |
| | **TOTAL** | **24888** | **30** |

O teste é iniciado por um elemento, o *Cliente*01 conforme Figura 4.11, que faz o papel do solicitante. Considerando as 30 solicitações indicadas, o total de *chunks* requisitados a cada experimento são 81.707, ou seja, em situação normal, sem erros ou re-tentativas, são geradas 81.707 mensagens de `REQUEST` pelo cliente.

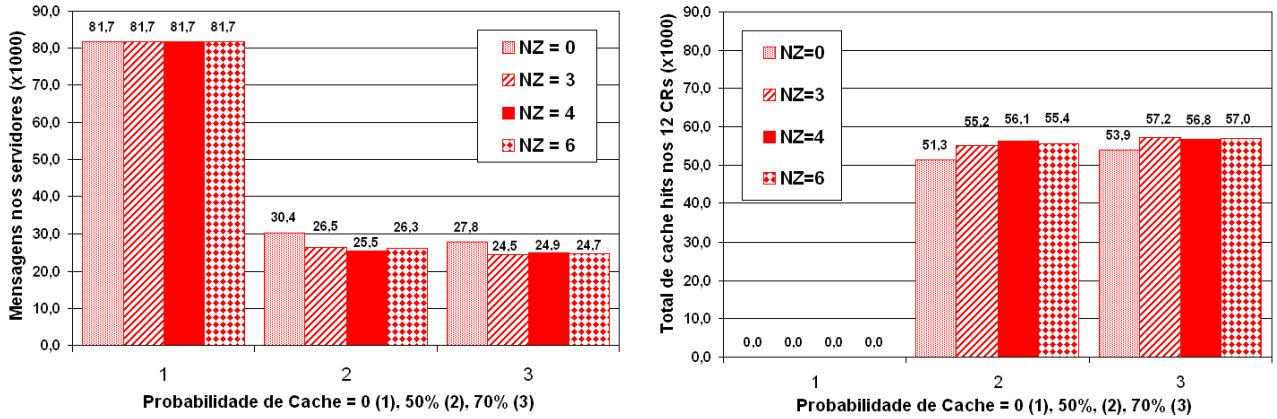
Todos os servidores têm os conteúdos disponíveis, ou seja, os 10 arquivos. No ambiente IP, as 30 requisições são distribuídas pelos 4 servidores, o que não acontece no ambiente no nível de enlace onde não existe o conceito do endereço destino.

Ao final do teste os arquivos de LOG e CSV são coletados nos diversos elementos.

Os resultados de testes a partir de um único cliente são apresentados para facilitar entendimento das variáveis envolvidas. Testes com variação da topologia foram executados e apresentaram comportamento comparáveis.

### Resultados - Ambiente IP/Overlay

Os gráficos na Figura 4.12(a) e na Figura 4.12(b) mostram o comportamento da rede orientada a conteúdo para ambiente IP/overlay nos diferentes cenários: probabilidade de *caching* (0%, 50%, 70%) e valor de *Neighbor Zones, NZ* (0, 3, 4, 6). É possível observar uma complementariedade entre a resolução nos servidores e nos CRs. Não são contabilizadas as mensagens do protocolo OSPF (ex: Hello, LSA, LSU) trocadas periodicamente em todas as interfaces entre os roteadores.




(a) Requisições atendidas pelos servidores

(b) Requisições atendidas pelos CRs (cache hit)

Figura 4.12: Tratamento de requisições (IP)

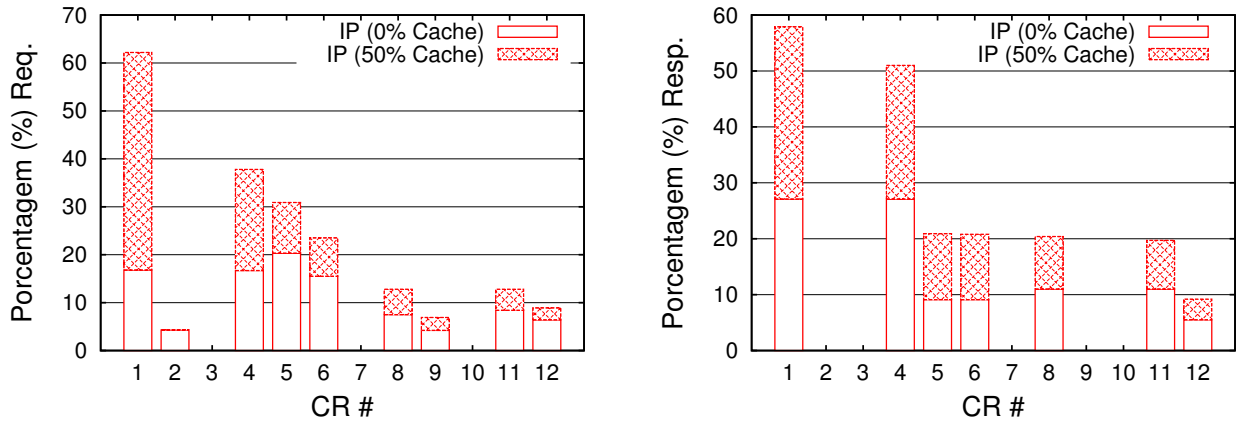
Nos cenários (1) sem disponibilidade de cache nos CRs, a resolução das requisições acontece sempre nos servidores. O valor de $NZ$ não colabora no processo pois a requisição é encaminhada sempre para o endereço do servidor.

Para cache habilitado, observar que a resolução acontece em grande parte nos CRs, diminuindo a carga nos servidores, porém sem muitas vantagens no aumento da probabilidade de *caching* de 50% para 70%. A busca na vizinhança ($NZ$) também colabora para aumentar a resolução na rede.

As Figuras 4.13(a) e 4.13(b) mostram a distribuição das mensagens `REQUEST` e `RESPONSE` tratadas nos CRs quando $NZ = 6$. Alguns CRs não participam do tratamento de mensagens (CRs 3,7 e 10) pois o OSPF não fez a opção pela rota que passa por eles no encaminhamento das mensagens.

### Resultados - Ambiente Ethernet

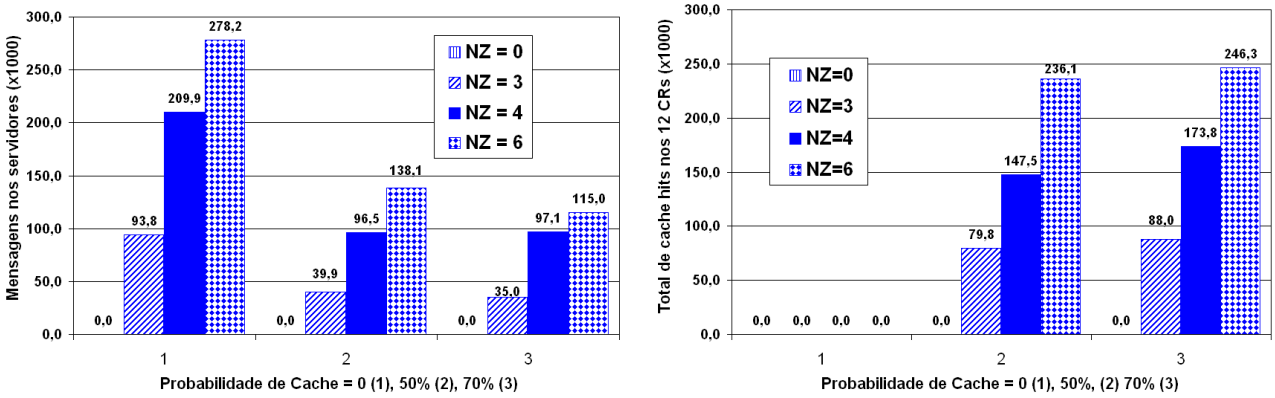
Os gráficos na Figura 4.14(a) e na Figura 4.14(b) mostram o comportamento da rede orientada a conteúdo para a proposta no nível de enlace (ambiente Ethernet) para os diferentes cenários sob avaliação (Probabilidade de *caching*, *# Neighbor Zones – NZ*). O *flooding flag* não está setado, ou seja, a busca pára quando o valor de NZones atinge zero.




(a) Mensagens `REQUEST` processadas

(b) Mensagens `RESPONSE` processadas

Figura 4.13: Distribuição de mensagens nos CRs (IP).




(a) Requisições atendidas pelos servidores

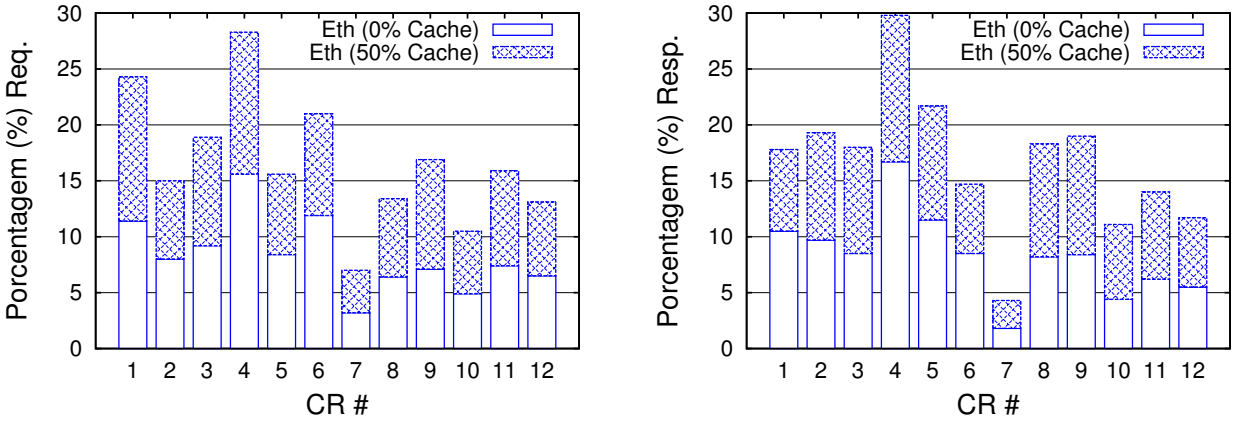
(b) Requisições atendidas pelos CRs (cache hit)

Figura 4.14: Tratamento de requisições (Ethernet)

Nos cenários sem cache (1) onde a resolução das requisições acontece nos servidores, o valor de $NZ$ tem impactos diversos: $NZ = 0$ : não faz busca, pára, conforme esperado. $NZ = 3$ : gera menor número de mensagens; resolução no servidor mais próximo ($Servidor02$). $NZ = 4$ ou $6$ : aumenta o número de mensagens geradas na rede; resolução distribuída nos demais servidores.

Para cache habilitado, verificar que a resolução acontece nos CRs, diminuindo a carga nos servidores, porém sem muitas vantagens no aumento da probabilidade de 50% para 70% como no caso do IP. A busca na vizinhança ($NZ$) colabora para aumentar o número de mensagens na rede.

As Figuras 4.15(a) e 4.15(b) mostram a distribuição das mensagens `REQUEST` e `RESPONSE` nos CRs quando $NZ = 6$. Nesta condição todos os CRs respondem às requisições apesar de nem todas serem "utilizadas": o *flooding* causa várias respostas para uma mesma requisição; estas respostas serão ignoradas pelos CRs que não têm a requisição pendente.




(a) Mensagens `REQUEST` processadas

(b) Mensagens `RESPONSE` processadas

Figura 4.15: Distribuição de mensagens nos CRs (Ethernet).

**Resultados - Ambiente IP vs. Ethernet**

A topologia do tipo Internet utilizada nos testes coloca a solução no nível de enlace em desvantagem comparada à solução IP/overlay pois aumenta o número de pontos de inundação e aumenta a quantidade de mensagens. Nas Figuras 4.12 e 4.14 é possível observar que apesar das quantidades de mensagens geradas na rede serem diferentes (Ethernet precisando de mais mensagens que o IP), o comportamento dos dois ambientes tende a ser o mesmo. As Figuras 4.15(a) e 4.15(b) mostram que a rede Ethernet distribui melhor o processamento entre os diferentes CR sem sobrecarregar nenhum CR em particular.

Já o gráfico na Figura 4.16 faz uma comparação dos dois ambientes em termos de número de *hops* desde o cliente consumidor até a fonte do conteúdo (CR ou servidor). É possível observar a tendência da resolução das requisições no $CR1$, mais próximo do $Cliente01$ que fez as requisições, nos dois ambientes. Vale a pena destacar que o ambiente Ethernet consegue caminhos mais curtos (60% das vezes com 4 ou menos *hops*) que o IP (60% das vezes com 6 ou menos *hops*) e atinge resultados quase comparáveis aos da rede IP com cache ativado (concentração entre 3 e 4 *hops*). Observar que em alguns casos, a distância percorrida é maior que NZones definido para o teste ($NZ = 6$). Isso ocorre como consequência do roteamento híbrido: houve tentativa de busca na vizinhança sem sucesso (roteamento no conteúdo) seguida do encaminhamento para o servidor (roteamento no IP), causando um acréscimo de *hops* na distância normal entre cliente e servidor.

Finalmente, a Tabela 4.2 compara o tempo de transferência do arquivo PacktpubMoodleSecurity.pdf que foi requisitado várias vezes nos experimentos. No *testbed* com os elementos virtualizados em uma máquina física o tempo não é um resultado que possa ser considerado de forma absoluta pelos efeitos da virtualização e o consumo concorrente de CPU, mas apresenta certa coerência. Os números mostram que nos dois ambientes o custo (tempo ou número de mensagens) é muito maior para a primeira requisição, pois não existe informação na tabela de roteamento e nem conteúdo armazenado na rede. A rede Ethernet tem um custo maior para a resolução da primeira requisição devido a busca por *flooding*. Porém, após o conhecimento das interfaces e ocorrência de *caching*, as duas redes se comportam de forma equivalente para

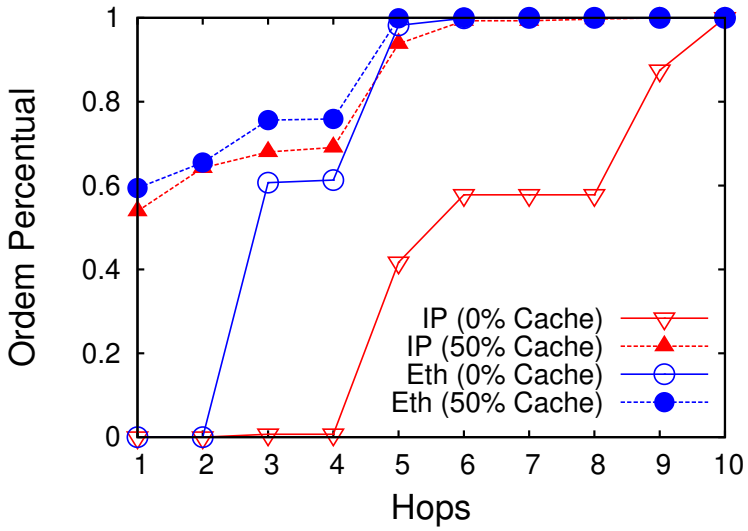



Figura 4.16: IP vs. Ethernet: Distância em saltos para resolução das requisições

as requisições subsequentes. O tempo se refere ao tempo de recepção (*download*) do arquivo (3406 chunks) pelo *Cliente*01 medido desde o tempo de envio do primeiro `REQUEST`. A Tabela 4.3 apresenta o total de mensagens `RESPONSE` geradas pela rede Ethernet, tanto nos CRs como nos servidores, para o mesmo experimento da Tabela 4.2. Os valores mostram que existe uma coerência com os valores relacionados ao tempo e também a tendência da resolução no CR mais próximo ao cliente.

Tabela 4.2: IP vs. Ethernet: Tempo de transferência em segundos de um arquivo de 3406 chunks para $NZ = 4$ e $Cache = 50\%$.

| Requisição # | $1^a$ | $2^a$ | $3^a$ | $4^a$ | $5^a$ |
|---|---|---|---|---|---|
| Tempo - IP | 24,55 | 5,62 | 3,50 | 2,86 | 2,56 |
| Tempo - Ethernet | 42,67 | 4,92 | 3,68 | 2,96 | 2,77 |

Tabela 4.3: Ethernet: Total de mensagens `RESPONSE` na transferência de um arquivo de 3406 chunks para $NZ = 4$ e $Cache = 50\%$.

| Requisição # | $1^a$ | $2^a$ | $3^a$ | $4^a$ | $5^a$ |
|---|---|---|---|---|---|
| Mensagens geradas nos CRs | 13723 | 4090 | 3707 | 3405 | 3406 |
| Mensagens geradas nos Servidores | 11702 | 75 | 28 | 1 | 0 |
| Total de Mensagens | 25425 | 4165 | 3735 | 3406 | 3406 |

### Resultados - Ambiente Ethernet com Variações Diversas

As tabelas a seguir comparam resultados de experimentos onde houve variação no número de entrada nas tabelas de *caching* e roteamento, na probabilidade de *caching*, no uso de *flooding flag* (FFlag) e no anúncio de conteúdos.

As Tabelas 4.4 e 4.5 trazem os resultados quando as tabelas de *caching* e roteamento foram configuradas para 10.000 entradas. Nos experimentos e resultados da Tabela 4.4, os CRs estão configurados para probabilidade de cache=0%, sendo que os piores casos são observados quando se usa a opção de continuar com o processo de inundação após NZones atingir o valor zero. A quantidade de mensagens geradas na rede causa situações de inconsistência nas listas de requisições pendentes dos diversos CRs e erros de *timeout*.

Na Tabela 4.5 os mesmos experimentos são realizados quando os CRs estão configurados para cache=50% e os valores observados são melhores.

O anúncio de conteúdo pelos servidores foram feitos com NZones = 3. Apesar de todos os servidores terem todos os conteúdos, cada um deles anuncia somente uma parte, ou seja, não existe anúncio repetido dos conteúdos, exceto pelo mais solicitado que é anunciado por 2 servidores. As requisições são feitas pelo Cliente01 com NZones = 6.

Com relação aos resultados presentes nas tabelas, a coluna **Msg Serv** traz o total de mensagens `REQUEST` recebidas pelos servidores, que é o mesmo valor das mensagens `RESPONSE` enviadas. A coluna **Msg rec CRs** traz o total de mensagens recebidas pelos CRs, incluindo todos os tipos de mensagens. A coluna **Msg env CRs** se refere ao total de mensagens enviadas pelos CRs. A coluna **Tmp Tx** indica o tempo de transferência (*download*) das 30 requisições (ou 81.707 *chunks*) sob o ponto de vista do cliente que faz a requisição. O valor é dado em segundos. A coluna **Perda** indica o percentual de pacotes perdidos na transferência, também sob o ponto de vista do cliente.

Tabela 4.4: Ethernet: Resultados com tabelas 10.000 entradas e cache=0%.

| # | **Anúncio** | **FFlag** | **Msg Serv** | **Msg rec CRs** | **Msg env CRs** | **Tmp Tx** | **Perda** |
|---|---|---|---|---|---|---|---|
| 1 | Não | 0 | 278.207 | 2.230.912 | 2.293.039 | 763,5 | 0 |
| 2 | Não | 1 | 37.120 | 243.911 | 299.594 | 178,5 | 98,7 |
| 3 | Sim | 0 | 221.648 | 2.067.467 | 2.159.091 | 694,9 | 0 |
| 4 | Sim | 1 | 160.009 | 1.539.723 | 1.765.662 | 267,1 | 94 |

Tabela 4.5: Ethernet: Resultados com tabelas 10.000 entradas e cache=50%.

| # | **Anúncio** | **FFlag** | **Msg Serv** | **Msg rec CRs** | **Msg env CRs** | **Tmp Tx** | **Perda** |
|---|---|---|---|---|---|---|---|
| 1 | Não | 0 | 138.140 | 1.761.045 | 1.839.182 | 610,9 | 0 |
| 2 | Não | 1 | 144.726 | 1.851.267 | 1.958.600 | 638,3 | 0 |
| 3 | Sim | 0 | 120.418 | 1.757.340 | 1.862.815 | 569,7 | 0 |
| 4 | Sim | 1 | 125.680 | 1.832.245 | 1.959.302 | 592,3 | 0 |

Os experimentos com *Flooding Flag* = 1 demonstraram que não há muito ganho e portanto não foram utilizados em outros cenários. Esta opção causa volume de mensagem muito grande na rede.

A Tabela 4.6 traz os resultados quando as as tabelas de *caching* e roteamento foram configuradas para 25.000 entradas. Nesta condição não ocorre substituição pois todos os *chunkIds* podem ser armazenados (ver quantidade total de *chunks* na 4.1). Nos experimentos e resultados, os CRs estão configurados para probabilidade de cache=0% e cache=50%.

Tabela 4.6: Ethernet: Resultados com tabelas 25.000 entradas.

| # | **Anúncio** | **Cache** | **Msg Serv** | **Msg rec CRs** | **Msg env CRs** | **Tmp Tx** | **Perda** |
|---|---|---|---|---|---|---|---|
| 1 | Não | 0% | 238.519 | 1.935.619 | 2.001.081 | 740,9 | 0 |
| 2 | Sim | 0% | 179.349 | 1.611.461 | 1.671.722 | 669,3 | 0 |
| 3 | Não | 50% | 125.916 | 1.579.168 | 1.662.090 | 656,0 | 0 |
| 4 | Sim | 50% | 78.999 | 1.271.560 | 1.335.133 | 472,5 | 0 |

Os resultados mostram as vantagens do armazenamento de conteúdos nos CRs com relação ao *caching* e às condição das tabelas quando as mesmas têm um maior número de entradas. O anúncio do conteúdo também traz benefícios e é possível observar a redução no número de mensagens atendidas pelos servidores, assim como a redução no número de mensagens recebidas e enviadas pelos CRs, mesmo quando não é feito o *caching* nos CRs.

**Resultados - Ambiente Ethernet com Super CR**

A Tabela 4.7 compara resultados de experimentos com relação ao anúncio de conteúdos e à presença de um Super CR na rede. O CR08 foi definido como sendo um Super CR para esta sequência de testes.

As tabelas de *caching* e roteamento foram configuradas para 25.000 entradas. Os CRs estão configurados para probabilidade de cache=50%. O anúncio de conteúdo pelos servidores foram feitos com variação do NZones (coluna NZ, valores 3 ou 4). Apesar de todos os servidores terem todos os conteúdos, cada um deles anuncia somente uma parte, ou seja, não existe anúncio repetido dos conteúdos, exceto pelo mais solicitado que é anunciado por 2 servidores. O anúncio do SCR foi feito pelo CR08 com NZones = 3, configurado com probabilidade de cache=50% e 100%. A coluna SCR indica a presença e o valor de cache do Super CR. As requisições são feitas pelo Cliente01 com NZones = 6. O tempo é dado em segundos e se refere ao tempo de transferência (*download*) das 30 requisições, 81.707 *chunks*.

Tabela 4.7: Ethernet: Uso do Super CR.

| # | **Anúncio** | **NZ** | **SCR** | **Msg Serv** | **Msg rec CRs** | **Msg env CRs** | **Tmp Tx** |
|---|---|---|---|---|---|---|---|
| 1 | Não | - | - | 120.569 | 1.528.579 | 1.605.614 | 631,9 |
| 2 | Sim | 4 | - | 83.283 | 1.347.260 | 1.446.913 | 487,0 |
| 3 | Sim | 4 | 50% | 41.396 | 726.947 | 740.948 | 323,2 |
| 4 | Sim | 3 | - | 77.170 | 1.259.637 | 1.325.119 | 473,9 |
| 5 | Sim | 3 | 50% | 30.520 | 499.522 | 496.528 | 281,8 |
| 6 | Sim | 3 | 100% | 28.564 | 485.047 | 479.525 | 287,6 |

Os resultados mostram melhora no desempenho quando existe o elemento Super CR na rede, mesmo sem uma configuração diferenciada de *caching*. A configuração diferenciada de *caching* melhora ainda mais os resultados. A tendência de resolução no CR mais próximo ao cliente se mantém.

### 4.2.2 Testes no Ambiente Real

A seguir é descrito o cenário e procedimento de teste no ambiente de máquinas reais que tiveram objetivo de avaliar a arquitetura no nível de enlace em ambiente similar a uma rede domiciliar. Tais redes podem se beneficiar com uma arquitetura simples, sem necessidade de configuração e gerenciamento como é o caso das redes IP.

#### Cenário de Teste

O cenário de rede domiciliar é ilustrado na Figura 4.17. O CR funciona como um *gateway* para a Internet. A ferramenta NetEm (Jurgelionis, Laulajainen, Hirvonen & Wang 2011) foi utilizada para emular comportamento onde ocorre perda de pacotes aleatória no *link* entre os clientes e o CR, com diferentes taxas. Os elementos da arquitetura executam em máquinas reais. Os testes consideram a requisição de um mesmo arquivo de vídeo de 92MB por três vezes consecutivas. Este cenário corresponde a uma situação onde um conteúdo popular é compartilhado entre os membros de uma família (ex. via uma rede social) ou acessado através de diferentes dispositivos num curto período de tempo. Não existe outro tráfego na rede.

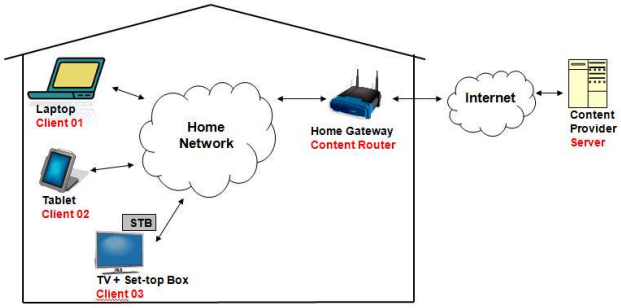



Figura 4.17: Topologia - Ambiente Real / Rede Domiciliar.

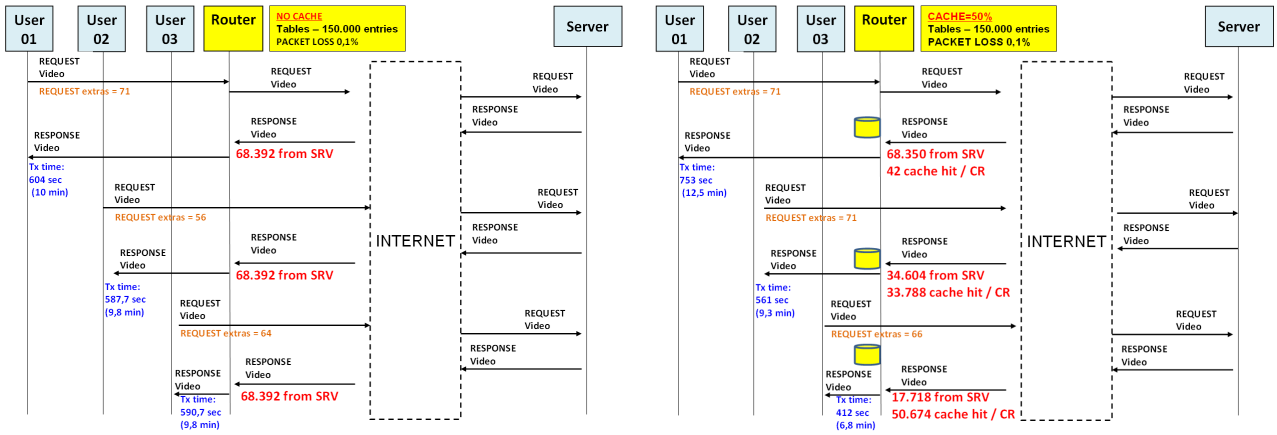
#### Procedimento Teste

O CR está configurado para probabilidade de *caching* de 0% e 50%. É feita a requisição do arquivo de vídeo em sequência. O comando da ferramenta NetEm é executado para provocar a perda de pacotes.

#### Resultados - Ambiente Real

As Figuras 4.18(a) e 4.18(b) mostram o comportamento e os resultados obtidos para o CR configurado com probabilidade de 0% e 50%, com a taxa de perda de pacotes $0,1\%$, o que significa a perda de 1 pacote a cada 1000.

A Tabela 4.8 mostra os resultados obtidos para o CR configurado com probabilidade de *caching* de 50% para diferentes valores da taxa de perda de pacotes. Como esperado, o *caching* reduz a carga do servidor nas segundas e terceiras requisições em todos os casos. Ou seja, o efeito é positivo mesmo com o aumento da taxa de perda de pacotes e situações de *timeouts*




(a) Cache = 0%

(b) Cache = 50%

Figura 4.18: Cenário Rede Domiciliar: Requisição de Vídeo

Tabela 4.8: Resultados - Cenário rede domiciliar. CR-gateway ($Cache = 50\%$).

| | Perda Pct = 0.01% | | | Perda Pct = 0.1% | | | Perda Pct = 1% | | | Perda Pct = 5% | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Requisição # | $1^a$ | $2^a$ | $3^a$ | $1^a$ | $2^a$ | $3^a$ | $1^a$ | $2^a$ | $3^a$ | $1^a$ | $2^a$ | $3^a$ |
| Resolução no CR | 15 | 33536 | 50751 | 42 | 33788 | 50674 | 348 | 33697 | 50644 | 1708 | 35096 | 51955 |
| Resolulção no Servidor | 68377 | 34856 | 17641 | 68350 | 34604 | 17718 | 68044 | 34695 | 17748 | 66684 | 33296 | 16437 |
| Erros *timeout* no cliente | 8 | 8 | 8 | 71 | 71 | 66 | 701 | 668 | 688 | 3526 | 3547 | 3579 |
| Tempo Transf.(segundos) | 722 | 533 | 383 | 753 | 561 | 412 | 1066 | 861 | 724 | 2480 | 2304 | 2170 |

ocorridas no cliente, o que causa reenvio de requisições. O tempo de transferência (*download*) também diminui entre a primeira e a terceira requisição mesmo com o aumento das taxas.

O teste sugere que conteúdos compartilhados em um cenário de rede domiciliar suportado por uma arquitetura orientada a conteúdo de nível de enlace, com característica *plug-and-play*, pode contribuir para reduzir a demanda por banda nas redes de acesso e no núcleo da rede além de acelerar a recuperação do conteúdo.

## 4.3 Conclusão do Capítulo

Este capítulo descreveu alguns aspectos da implementação do protótipo utilizado para validar a arquitetura dos CRs diretamente no nível de enlace e também no nível IP/*overlay*. Também apresentou a metodologia utilizada para os testes de validação assim como os resultados dos testes. As análises realizadas nesse capítulo demonstram a viabilidade do roteamento de conteúdo no nível de enlace e os benefícios de recursos como o Super CR.

Capítulo 5

# Conclusões e Trabalhos Futuros

## 5.1 Conclusões

As Redes Orientadas a Conteúdo(ROC) se apresentam como um novo paradigma tendo o conteúdo como a base da comunicação, independente de sua localização. Esta dissertação apresentou uma proposta da arquitetura de *Content Routers* diretamente no nível de enlace e os resultados obtidos nos experimentos mostraram que é possível trabalhar diretamente no nível de enlace com roteamento de conteúdo e controle de inundação de mensagens, substituindo um roteamento com base na localização.

Duas abordagens para a arquitetura de CRs foram avaliadas: a primeira é efetivamente uma arquitetura híbrida com roteamento no conteúdo e no IP e a segunda é completamente limitada ao nível de enlace com roteamento no conteúdo.

A principal vantagem desta proposta é dispensar o roteamento baseado em localização e operar diretamente na camada de enlace através do roteamento baseado no conteúdo. A contrapartida da abordagem é um maior número de mensagens trocadas na rede. Porém, considerando o desempenho apresentado pela proposta uma vez aprendidos os caminhos até o conteúdo, vale a pena considerar o *trade-off* de sobrecarga em tráfego de controle pela simplicidade de uma arquitetura *plug-and-play* no nível de enlace. Como na arquitetura CCN, a nossa proposta não precisa do IP e faz roteamento com base no conteúdo, porém o CCN não tem uma proposta específica para o nível de enlace.

Abordagens no espírito da proposta deste trabalho podem ser uma opção para redes domiciliares ou outros tipos de redes menores onde não existe necessidade de endereçamento IP ou configurações mais complexas. Com o aumento de dispositivos do tipo *tablets*, *laptops*, *smartphones* e aparelhos de segurança para monitoramento é esperado adição de roteadores nestas redes. O uso de *gateways* não IP como o Super CR é uma opção para aumentar a complexidade da rede sem aumento da complexidade de operação da rede. Essa característica que deve ser considerada principalmente no contexto de redes domiciliares onde os usuários não têm treinamento técnico e, em geral, precisam de ajuda para instalação e configuração de novos dispositivos (Edwards, Grinter, Mahajan & Wetherall 2011).

## 5.2 Trabalhos Futuros

Como possíveis trabalhos futuros pode-se apontar:

**Arquitetura:** o sistema de nomeação pode explorar a ideia de agregação flexível conforme proposto em (Ghodsi, Koponen, Rajahalme, Sarolahti & Shenker 2011) e assim criar a possibilidade de roteamento no nível do nome do conteúdo ou de seu publicador, além do atual roteamento a nível dos *chunks*; a mensagem `NACK` pode ser acrescentada ao protocolo da rede para evitar "lixo" na lista de requisições pendentes; a mensagem `REQUEST` pode ter um *nonce* associado para auxiliar na detecção de *loops*; o Super CR pode ter suas funções ampliadas e funcionar como: (i) *gateway* para conexão das redes Ethernet e IP, (ii) servidor de resolução de nomes, ou (iii) ponto de *rendezvous* para suporte ao modelo *publish/subscribe*.

**Otimização de mecanismos:** as tabelas de roteamento e *caching* podem utilizar outras estratégias além da utilizada atualmente (FIFO); a tabela de roteamento mantém somente a interface que indica a menor distância em *hops* e não uma lista de opções como na proposta original da arquitetura CCN (Jacobson, Smetters, Thornton, Plass, Briggs & Braynard 2009); inclusão do aprendizado de quais são as melhores interfaces (cf. (Yi et al. 2012)), seja pelo tempo ou percentual de respostas; avaliação de processamento instanciado por interface; inserção do mecanismo de resolução de nomes e obtenção do *metadata*; uso de um *proxy* para aplicações legadas; inserção de políticas de *caching*. A introdução do elemento Super CR abre a possibilidade de uso de árvores de espalhamento (*Spanning Tree*) para controle de inundação.

**Otimização na Lista de Requisições Pendentes:** os filtros de Bloom podem causar falso positivo. No caso do BF contador, um falso positivo permite decrementar e zerar valores dos contadores. Esta situação pode resultar em falso negativo e na permanência de "lixo" na estrutura. Artigos na literatura (Wang, Lee, Venkataraman, Lingappa & Rhee 2011), (Tortelli, Grieco & Boggia 2012) discutem o uso de filtros de Bloom no contexto das redes de conteúdo, alguns especificamente para o controle das listas de requisições pendentes ou PIT (Li, Bi, Wang & Jiang 2012).

**Aplicação em cenários de *HomeNets*:** instanciar uma versão do protótipo numa rede domiciliar combinando ambiente sem fio e cabeado onde o Super CR atua como *gateway* para o mundo IP e onde qualquer dispositivo pode atuar como CR provendo ampla capacidade de *caching* e colaborativamente oferecendo um sistema de arquivos distribuído para os usuários da HomeNet; interconexão de vários *gateways* num cenário de rede de condomínio; investigar a utilização de tecnologia Wi-Fi Direct.

# Bibliografia


Ahlgren, B., Aranda, P., Chemouil, P., Oueslati, S., Correia, L., Karl, H., Söllner, M. & Welin, A. (2011). Content, connectivity, and cloud: ingredients for the network of the future, *IEEE Communications Magazine* **49**(7): 62–70.
**URL:** *http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5936156*

Ahlgren, B., D'Ambrosio, M., Marchisio, M., Marsh, I., Dannewitz, C., Ohlman, B., Pentikousis, K., Strandberg, O., Rembarz, R. & Vercellone, V. (2008). Design considerations for a network of information, *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, ACM, New York, NY, USA, pp. 66:1–66:6.
**URL:** *http://doi.acm.org/10.1145/1544012.1544078*

Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D. & Ohlman, B. (2012). A survey of information-centric networking, *Communications Magazine, IEEE* **50**(7): 26–36.
**URL:** *http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6231276*

Ambiel, L., Rothenberg, C. E. & Magalhaes, M. (2013a). Content oriented networking: A promising alternative for home networks, *IWT 2013*, Santa Rita do Sapucai, Brazil.
**URL:** *http://www.inatel.br/iwt/index.php/component/docman/cat_view/5-papers?start=30*

Ambiel, L., Rothenberg, C. E. & Magalhaes, M. (2013b). Redes orientadas a conteudo: Abordagem no nivel de enlace, *SBRC 2013*, Brasilia, Brasil.
**URL:** *http://sbrc2013.unb.br/files/anais/trilha-principal/artigos/artigo-51.pdf*

Bernstein, D. J. (2009). CubeHash specification (2.B.1).
**URL:** *http://cubehash.cr.yp.to/submission2/spec.pdf*

Chiocchetti, R., Rossi, D., Rossini, G., Carofiglio, G. & Perino, D. (2012). Exploit the known or explore the unknown?: hamlet-like doubts in icn, *Proceedings of the second edition of the ICN workshop on Information-centric networking*, ICN '12, ACM, New York, NY, USA, pp. 7–12.
**URL:** *http://doi.acm.org/10.1145/2342488.2342491*

Choi, J., Han, J., Cho, E., Kwon, T. & Choi, Y. (2011). A survey on content-oriented networking for efficient content delivery, *Communications Magazine, IEEE* **49**(3): 121–127.
**URL:** *http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5723809*

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, third edit edn, Sage, Los Angeles.

Dannewitz, C., Golic, J., Ohlman, B. & Ahlgren, B. (2010). Secure Naming for a Network of Information, *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, IEEE, pp. 1–6.
**URL:** *http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5466661*

Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B. & Karl, H. (2013). Network of information (netinf): An information-centric networking architecture, *Computer Communications* **36**(7): 721 – 735.
**URL:** *http://www.sciencedirect.com/science/article/pii/S0140366413000364*

Day, M., Cain, B., Tomlinson, G. & Rzewski, P. (2003). A Model for Content Internetworking (CDI) RFC 3466, *Technical report*, RFC Editor, United States.
**URL:** *http://wiki.tools.ietf.org/pdf/rfc3466.pdf*

de Brito, G. M., Velloso, P. B. & Moraes, I. M. (2012). Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet, *Minicursos do Simpósio Brasileiro de Redes de Computadores*, SBRC 2012, pp. 211–264.
**URL:** *http://sbrc2012.dcc.ufmg.br/app/p-04-g.html*

Detti, A., Blefari Melazzi, N., Salsano, S. & Pomposini, M. (2011). CONET: a content centric inter-networking architecture, *Proceedings of the ACM SIGCOMM Workshop on ICN*, pp. 50–55.
**URL:** *http://dl.acm.org/citation.cfm?id=2018584.2018598*

Detti, A., Pomposini, M., Blefari-Melazzi, N. & Salsano, S. (2012). Supporting the web with an information centric network that routes by name, *Computer Networks* **56**(17): 3705 – 3722.
**URL:** *http://www.sciencedirect.com/science/article/pii/S1389128612002964*

Edwards, W. K., Grinter, R. E., Mahajan, R. & Wetherall, D. (2011). Advancing the state of home networking, *Communications of the ACM* **54**(6): 62–71.
**URL:** *http://doi.acm.org/10.1145/1953122.1953143*

Fan, L., Cao, P., Almeida, J. & Broder, A. Z. (2000). Summary cache: a scalable wide-area web cache sharing protocol, *IEEE/ACM Trans. Netw.* **8**(3): 281–293.
**URL:** *http://dx.doi.org/10.1109/90.851975*

Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, P. & Shenker, S. (2011). Naming in content-oriented architectures, *Proceedings of the ACM SIGCOMM workshop on Information-*

*centric networking*, ICN '11, ACM, New York, NY, USA, pp. 1:1–1:6.
**URL:** *http://doi.acm.org/10.1145/2018584.2018586*

Ghodsi, A., Ohlmann, B., Ott, J., Solis, I. & Wählisch, M. (2013). Information-centric networking – Ready for the real world? (Dagstuhl Seminar 12361), *Dagstuhl Reports* **2**(9): 1–14.
**URL:** *http://drops.dagstuhl.de/opus/volltexte/2013/3787*

Ghodsi, A., Shenker, S., Koponen, T., Singla, A., Raghavan, B. & Wilcox, J. (2011). Information-centric networking: seeing the forest for the trees, *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, ACM, New York, NY, USA, pp. 1–6.
**URL:** *http://doi.acm.org/10.1145/2070562.2070563*

Hofmann, M. & Beaumont, L. R. (2005). Content Networking - Architecture, Protocols and Practice, *in* D. Clark (ed.), *Content Networking: Architecture, Protocols and Practice*, Morgan Kaufmann.
**URL:** *http://www.sciencedirect.com/science/book/9781558608344*

ITU-T Study Group 13 (2011). ITU-T Y.3001 : Future networks: Objectives and design goals.
**URL:** *http://www.itu.int/rec/T-REC-Y.3001-201105-I/en*

Jacobson, V., Mosko, M., Smetters, D. K. & Garcia-Luna-Aceves, J. J. (2007). Content-centric networking whitepaper, Palo Alto Research Center, pp. 2–4.
**URL:** *http://mmlab.snu.ac.kr/courses/2007_advanced_internet/papers/20071010_1.pdf*

Jacobson, V., Smetters, D. K., Briggs, N. H., Plass, M. F., Stewart, P., Thornton, J. D. & Braynard, R. L. (2009). VoCCN: Voice-over content-centric networks, *Proceedings of the 2009 ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT'09 - Co-located 2009 Workshop on Re-Architecting the Internet, ReArch'09*, Rome, Italy, pp. 1–6.
**URL:** *http://dx.doi.org/10.1145/1658978.1658980*

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H. & Braynard, R. L. (2009). Networking named content, *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, ACM, New York, NY, USA, pp. 1–12.
**URL:** *http://doi.acm.org/10.1145/1658939.1658941*

Jain, S. & Pandey, M. (2012). Hash table based word searching algorithm.
**URL:** *http://ijcsit.com/docs/Volume 3/vol3Issue3/ijcsit20120303116.pdf*

Jokela, P., Zahemszky, A., Esteve Rothenberg, C., Arianfar, S. & Nikander, P. (2009). Lipsin: Line speed publish/subscribe inter-networking, *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, ACM, pp. 195–206.
**URL:** *http://dl.acm.org/citation.cfm?id=1592592*

Jurgelionis, A., Laulajainen, J.-P., Hirvonen, M. & Wang, A. (2011). An empirical study of netem network emulation functionalities, *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pp. 1 –6.
**URL:** *http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6005933&tag=1*

Koponen, T., Chawla, M., Chun, B.-G., Ermolinskiy, A., Kim, K. H., Shenker, S. & Stoica, I. (2007). A data-oriented (and beyond) network architecture, *ACM SIGCOMM Computer Communication Review* pp. 181–192.
**URL:** *http://dl.acm.org/citation.cfm?id=1282427.1282402*

Kulik, J., Heinzelman, W. & Balakrishnan, H. (2002). Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks, *Wireless Networks* **8**(2-3): 169–185.
**URL:** *http://link.springer.com/article/10.1023/A%3A1013715909417*

Lagutin, D., Visala, K. & Tarkoma, S. (2010). Publish/Subscribe for Internet: PSIRP Perspective.
**URL:** *http://www.booksonline.iospress.nl/Content/View.aspx?piid=16465*

Li, Z., Bi, J., Wang, S. & Jiang, X. (2012). Compression of pending interest table with bloom filter in content centric network, *Proceedings of the 7th International Conference on Future Internet Technologies*, CFI '12, ACM, New York, NY, USA, pp. 46–46.
**URL:** *http://doi.acm.org/10.1145/2377310.2377326*

Merkle, R. C. (1989). A certified digital signature, *Proceedings on Advances in cryptology*, CRYPTO '89, Springer-Verlag New York, Inc., New York, NY, USA, pp. 218–238.
**URL:** *http://dl.acm.org/citation.cfm?id=118209.118230*

Meyer, D., Zhang, L. & Fall, K. (2007). Report from the IAB Workshop on Routing and Addressing - RFC 4984.
**URL:** *http://tools.ietf.org/pdf/rfc4984.pdf*

Pal, S. K., Sardana, P. & Yadav, K. (2012). Efficient multilingual keyword search using bloom filter for cloud computing applications, *2012 Fourth International Conference on Advanced Computing (ICoAC)*, IEEE, pp. 1–7.
**URL:** *http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6416809*

Pathan, M., Buyya, R. & Vakali, A. (2008). Content Delivery Network: State of the Art, Insights, and Imperatives, *in* R. Buyya, M. Pathan & A. Vakali (eds), *Content Delivery Networks*, Vol. 9 of *Lecture Notes in Electrical Engineering*, Springer Berlin Heidelberg, pp. 3–32.
**URL:** *http://www.springerlink.com/content/lhu512116752359t/*

Perino, D. & Varvello, M. (2011). A reality check for content centric networking, *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, ACM Press, pp. 44 –49.
**URL:** *http://dl.acm.org/citation.cfm?id=2018584.2018596*

Psaras, I., Chai, W. K. & Pavlou, G. (2012). Probabilistic in-network caching for information-centric networks, *Proceedings of the second edition of the ICN workshop*, ACM Press, pp. 55 – 60.
**URL:** *http://dl.acm.org/citation.cfm?id=2342488.2342501*

Qwasmi, N. & Liscano, R. (2013). Bloom filter supporting distributed policy-based management in wireless sensor networks, *Procedia Computer Science* **19**(null): 248–255.
**URL:** *http://dx.doi.org/10.1016/j.procs.2013.06.036*

Ramakrishna, M. V. & Zobel, J. (1997). Performance in practice of string hashing functions, *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, Melbourne, Australia, pp. 215 – 223.

Roberts, J. (2009). The clean-slate approach to future Internet design: a survey of research initiatives, *Annals of Telecommunications* **64**(5-6): 271–276.
**URL:** *http://www.springerlink.com/content/e240776641607136/*

Rothenberg, C. E., Verdi, F. L. & Magalhães, M. F. (2008). Towards a new generation of information-oriented internetworking architectures, *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, ACM, pp. 65:1–65:6.
**URL:** *http://doi.acm.org/10.1145/1544012.1544077*

Tarkoma, S., Ain, M. & Visala, K. (2009). The Publish/Subscribe Internet Routing Paradigm (PSIRP): Designing the Future Internet Architecture, *Towards the Future Internet - A European Research Perspective, 2009*, pp. 102–111.
**URL:** *http://www.booksonline.iospress.nl/Content/View.aspx?piid=12016*

Tarkoma, S., Rothenberg, C. E. & Lagerspetz, E. (2012). Theory and Practice of Bloom Filters for Distributed Systems, *IEEE Communications Surveys & Tutorials* **14**(1): 131–155.
**URL:** *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5751342*

Tortelli, M., Grieco, L. A. & Boggia, G. (2012). Ccn forwarding engine based on bloom filters, *Proceedings of the 7th International Conference on Future Internet Technologies*, CFI '12, ACM, New York, NY, USA, pp. 13–14.
**URL:** *http://doi.acm.org/10.1145/2377310.2377314*

Vakali, A. & Pallis, G. (2003). Content delivery networks: Status and trends, *IEEE Internet Computing* **7**: 68–74.
**URL:** *http://dl.acm.org/citation.cfm?id=1050672.1050717*

Wang, L., Hoque, A. M., Yi, C., Alyyan, A. & Zhang, B. (2012). Ospfn: An ospf based routing protocol for named data networking, *NDN Technical Report NDN-0003, July 2012* .
**URL:** *http://www.named-data.net/techreport/TR003-OSPFN.pdf*

Wang, Y., Lee, K., Venkataraman, B., Lingappa, R. & Rhee, I. (2011). A Step towards Practical Deployment of the Content-Centric Networking Architecture, *Proceedings of The ACM*

*CoNEXT Student Workshop*, CoNEXT '11 Student, ACM CoNEXT Student Workshop.
**URL:** *http://dl.acm.org/citation.cfm?doid=2079327.2079347*

Wong, W. (2011). Security plane for data authentication in information-centric networks.
**URL:** *http://www.bibliotecadigital.unicamp.br/document/?code=000836755&fd=y*

Wong, W., Giraldi, M., Magalhaes, M. F. & Kangasharju, J. (2011). Content Routers: Fetching Data on Network Path, *IEEE International Conference in Communications*, IEEE, pp. 1–6.
**URL:** *http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5963111*

Wong, W., Wang, L. & Kangasharju, J. (2012). Neighborhood Search and Admission Control in Cooperative Caching Networks, *Globecom 2012 - Next Generation Networking and Internet Symposium*, pp. 2876–2882.
**URL:** *http://www-mobile.ecs.soton.ac.uk/home/conference/Globecom_12/papers/p2876-wong.pdf*

Yi, C., Afanasyev, A., Wang, L., Zhang, B. & Zhang, L. (2012). Adaptive forwarding in named data networking, *SIGCOMM Comput. Commun. Rev.* **42**(3): 62–67.
**URL:** *http://dl.acm.org/citation.cfm?id=2317307.2317319*

Zhang, L., Estrin, D., Burke, J., Jacobson, V., Thornton, J. D., Smetters, D. K., Zhang, B., Tsudik, G., Claffy, K., Krioukov, D., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P. & Yeh, E. (2010). Named Data Networking (NDN) Project.
**URL:** *https://www.parc.com/content/attachments/named-data-networking.pdf*

Zhu, Z., Wang, S., Yang, X., Jacobson, V. & Zhang, L. (2011). ACT: audio conference tool over named data networking, *Proceedings of the ACM SIGCOMM workshop on Information-centric networking - ICN '11*, ACM Press, New York, New York, USA, pp. 68 – 73.
**URL:** *http://dl.acm.org/citation.cfm?id=2018584.2018601*