SDCCN - Cache de Conteúdo Programável Definido por Software para Redes Orientadas a Conteúdo

Sergio R. C. Junior [1], Magnos Martinelo [1], Rodolfo da Silva Villaça [1] *

[1]Programa de Pós-Graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES)

{magnos, scjunior}@inf.ufes.br, rodolfo.villaca@ufes.br

Abstract. *Content Centric Networking (CCN) is a novel approach in the context of Future Internet research. These networks represent a major change in the current operation of the Internet, priorizing content over the communication between end nodes. The key components in these networks (i.e. , the routers), which receive content interest requests, have its own forwarding strategies and different caching policies for the most popular contents. However, the experimental evaluation of different forwarding algorithms and caching policies may require a huge effort in reprogramming the routers. Thus, this paper proposes an SDN approach to CCN networks, providing programmable forwarding strategies and caching policies in CCN routers to enable fast prototyping and experimentation in CCNs. Experimental results, obtained through implementation in the Mininet environment, are presented and evaluated.*

Resumo. *As Redes Orientadas a Conteúdo (Content Centric Networking – CCN) são um novo paradigma de redes no contexto da Internet do Futuro. Essas redes trazem uma mudança importante em relação ao funcionamento atual da Internet, colocando o conteúdo como item central em detrimento da comunicação entre nós terminais. Os nós centrais destas redes (roteadores), que recebem pedidos de interesse por conteúdos, possuem suas próprias estratégias de encaminhamento e diferentes políticas de armazenamento em cache para os conteúdos mais populares. Entretanto, a avaliação experimental de diferentes algoritmos de encaminhamento e políticas de cache exige muito esforço na reprogramação dos roteadores. Desta forma este artigo propõe uma abordagem SDN para redes CCN, provendo estratégias de encaminhamento e políticas de cache programáveis nos roteadores facilitando a experimentação de CCNs. Resultados experimentais são apresentados e foram obtidos através de implementação realizada no ambiente Mininet.*

1. Introdução

A engenharia da arquitetura e dos princípios da Internet de hoje foi criada nas décadas de 60 e 70 para resolver problemas de compartilhamento de recursos disponibilizados por dispositivos caros e escassos. O modelo de comunicação resultante foi de conversação entre dois únicos dispositivos [Jacobson et al. 2009] e, desde então, esta engenharia não mudou. Novos ”remendos”vêm sendo adicionados constantemente para suportar novas aplicações e diferentes necessidades de tráfego.

*Agradecimentos à CAPES, CNPq e FAPES pelo financiamento parcial deste trabalho.

Uma das perspectivas das Redes de Entrega de Conteúdo (*Content Delivery Networks* – CDNs), é que essas redes irão transportar mais da metade do tráfego da Internet em 2017 [Cisco 2013], e o tráfego de vídeo dos usuários corresponderá a 69% [Cisco 2013]. O relatório ainda ressalta que o usuário estará somente interessado no conteúdo ("*what*"), independente de sua localização, enquanto hoje a comunicação continua baseada na localização ("*where*") dos mesmos.

Nesse contexto, as Redes Orientadas a Conteúdo (*Content Centric Network* - CCN) surgiram para fornecer uma nova abstração para os problemas de comunicação atuais, fornecendo endereçamento, roteamento e segurança baseados no conteúdo e *cache* nativo fornecido pela rede. Entretanto, novos problemas surgem nessas redes, tais como a escalabilidade dos endereços, que são muito maiores que o número de hospedeiros e a necessidade de substituição dos equipamentos de rede, em especial os roteadores.

Um roteador CCN típico possui uma FIB (*Forwarding Information Base*) para roteamento de requisições de dados, uma tabela PIT (*Pending Interest Table*) para encaminhamento de conteúdo e uma *Content Store* para *cache* de conteúdo. Um dos problemas nessa arquitetura é que o algoritmo de substituição de *cache* e estratégia de encaminhamento de pacotes não são flexíveis o suficiente para serem modificados em tempo de execução, ou até mesmo permitir a execução de diferentes estratégias em multiplos nós sem a necessidade de modificação de código dos roteadores CCN.

Desta forma este artigo apresenta uma abordagem utilizando Redes Definidas por Software (*Software Defined Networking* – SDN) visando adicionar flexibilidade em roteadores CCN. O SDCCN (*Software Defined Content Centric Network*) suporta roteamento e cache programáveis e um ambiente de prototipação para a fácil realização de experimentos e soluções inovadoras orientadas à conteúdo. A solução elimina a necessidade de mapemanto entre nomes de conteúdo e identificadores, processando nativamente nomes de conteúdo com tamanho variável com base no protocolo POF (*"Protocol Oblivious Forwarding*) [Song 2013]. O Mininet [Lantz et al. 2010] foi escolhido como ambiente para prova de conceito e adaptado para suportar a solução. Esse ambiente de experimentação foi utilizado para comparação de políticas de *cache* e demonstração dos benefícios da abordagem SDN em CCN.

O restante deste artigo divide-se na seguinte forma: a Seção 2 discute e compara alguns trabalhos relacionados com a proposta do SDCCN. Na Seção 3 é apresentada a arquitetura da solução SDCCN. Na Seção 4 são mostrados detalhes da implementação realizada e na Seção 5 é mostrado os resultados experimentais obtidos com a implementação. Por fim, a Seção 6 conclui o artigo e apresenta propostas de trabalhos futuros.

2. Trabalhos Relacionados

O COntent NETwork (CONET) [Detti et al. 2011] estende a abordagem proposta por [Jacobson et al. 2009] (CCNx) em vários aspectos, tais como a integração com IP, escalabilidade de roteamento, uso de novos mecanismos para transporte, roteamento interdomínio e integração com a arquitetura SDN utilizando OpenFlow [McKeown et al. 2008]. Para contornar o problema de processamento de nomes de conteúdo de tamanho variável, o CONET faz um mapeamento entre o nome do conteúdo e um identificador (TAG) e utiliza esta TAG para roteamento de conteúdo. Além disso, estende o protocolo OpenFlow para suportar operações de roteamento baseado nas TAGs,

segurança e *cache*. A escalabilidade no roteamento é alcançada armazenando-se somente parte da FIB (*Forwarding Information Base*) nos roteadores e a FIB completa no plano de controle, denominado *Name Routing System* (NRS). O CONET apresenta um novo protocolo CCN e não suporta o CCNx, ao contrário do SDCCN, que suporta a proposta original do CCNx.

Nguyen [Nguyen et al. 2013] propõe uma abordagem SDN para implementação de "*off-path caching*" em CCN. Uma camada intermediária denominada "*wrapper*" faz a integração entre os protocolos CCN e OpenFlow, mapeando nomes de conteúdo para identificadores gerados por funções de *hashing*. Apesar de o tamanho do espaço de nomes de conteúdo ser grande, ainda existe a possibilidade de haverem colisões, o que não ocorreria se fosse mantido o nome original do conteúdo, como na proposta do SDCCN.

Syrivelis [Syrivelis et al. 2012] discute como criar uma arquitetura SDN para CCN e propõe um projeto de nó CCN utilizando OpenFlow. Vahlenkamp [Vahlenkamp et al. 2013] propõe uma solução para integração gradual de nós CCN em redes IP através de uma abordagem SDN. Em nenhuma dessas propostas são discutidos aspectos relacionados a programabilidade do *cache*.

A abordagem proposta neste artigo se difere das demais principalmente por: (i) eliminar a necessidade de mapeamentos entre nomes de conteúdo e identificadores, suportando nativamente o processamento de nomes do conteúdo com tamanhos variáveis; (ii) permitir que os protocolos CCN sejam implementados de maneira agnóstica nos elementos de rede; (iii) utilizar um ambiente para prototipação de redes CCN programáveis tornando fácil e escalável a realização de experimentos; (iv) fornecer uma implementação com roteamento e *cache* programáveis em redes CCN.

3. Arquitetura SDCCN

A arquitetura SDCCN é formada por usuários (fornecedores e consumidores de conteúdo), *switches* CCN (CCN SW) e um controlador CCN (logicamente centralizado). Existem dois tipos de pacotes: pacotes de Interesse (*Interest*), utilizados para requisitar um conteúdo, e pacotes de Conteúdo (*Content*), utilizados para fornecer um conteúdo. Consumidores requisitam conteúdo enviando pacotes do tipo Interesse na rede. *Switches* encaminham os pacotes e mantém em *cache* os conteúdos mais populares. Controladores gerenciam os *switches* programando estratégias de encaminhamento para os pacotes de Interesse e o armazenamento de conteúdos em *cache*. Fornecedores de conteúdo respondem as requisições de Interesse enviando pacotes de Conteúdo na rede CCN e também divulgam prefixos de nomes de conteúdos que produzem.

A Figura 1 ilustra a arquitetura da rede SDCCN. Cada *switch* é formado por: (i) uma FIB, que denominaremos de Tabela de Encaminhamento (TE), para armazenar os fluxos referentes ao encaminhamento de pacotes de Interesse; (ii) uma Tabela de Regras de *Cache* (TRC) para armazenar regras de *cache* que informam quais conteúdos devem ser armazenados em *cache*; (iii) uma *content store* (CS), que denominaremos de Tabela de Conteúdo (TC), para o armazenamento de conteúdo; e (iv) uma PIT (*Pending Interest Table*), que denominaremos de Tabela de Interesses (TI), utilizada para agrupar requisições de Interesse para um mesmo conteúdo que ainda encontram-se pendentes, ou seja, não receberam um pacote de Conteúdo correspondente ao Interesse requisitado.

O controlador além de poder alterar o conteúdo destas tabelas, também pode obter

informações sobre elas, a qualquer momento, através de consultas dos CCN SW. Entre os vários benefícios resultantes dessa separação dos planos de dados e de controle em redes CCN, destacamos: (i) algoritmos de substituição de *cache* podem ser modificados em tempo de execução e definidos de forma diferenciada em diferentes pontos da rede; (ii) conteúdos podem ser pré-instalados na TC dos *switches* de tal forma a permitir engenharia de tráfego de conteúdo nas redes CCN; (iii) considerando que o controlador possui uma visão geral da rede, ele pode tomar decisões de roteamento baseadas na localização dos *caches* que contém os conteúdos mais populares.

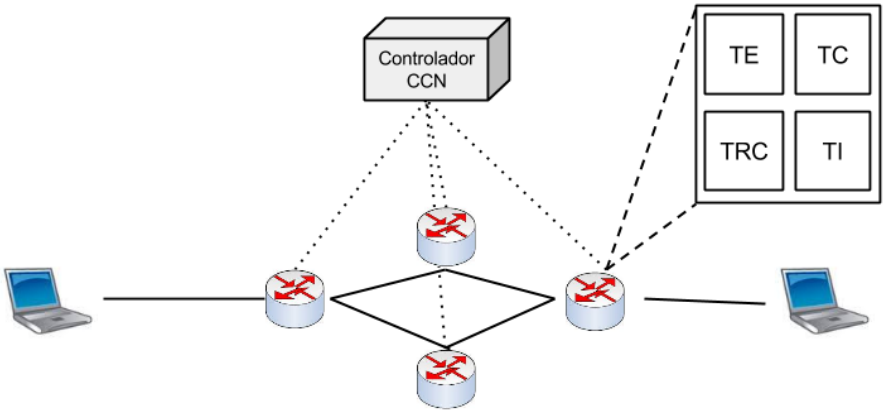


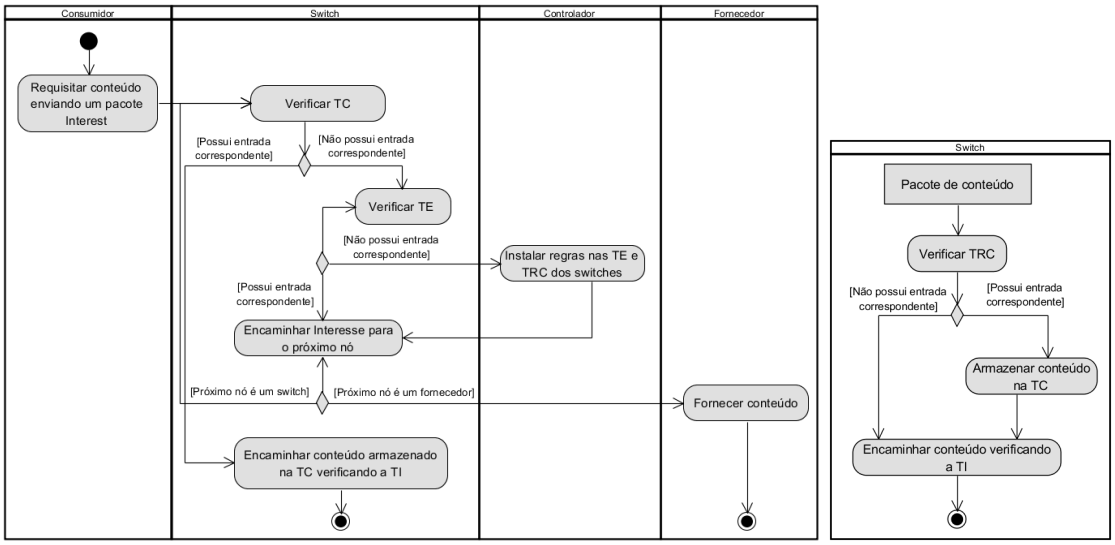
Figura 1. Arquitetura da solução proposta.

3.1. Encaminhamento

O diagrama de atividades da Figura 2 dá uma visão geral do encaminhamento de pacotes na arquitetura SDCCN. Um usuário requisita conteúdo enviando pacotes do tipo Interesse na rede. Os *switches*, ao receberem esse pacote, verificam primeiramente se o conteúdo requisitado está armazenado em sua TC. Caso esteja, enviam o pacote do tipo Conteúdo correspondente ao pacote Interesse recebido. Caso contrário, encaminham o pacote Interesse para o próximo *switch* (de acordo com alguma estratégia de encaminhamento) e armazenam uma entrada em sua TI contendo o nome do conteúdo e a porta em que o pacote foi recebido. O encaminhamento é feito consultando a TE, que faz um mapeamento entre nome de conteúdo e portas de saída. Caso não encontre uma entrada correspondente na TE, o *switch* consulta o controlador. Este determina como deve ser o encaminhamento e instala regras na TE dos *switches*.

O encaminhamento do pacote de Conteúdo em resposta ao Interesse é feito consultando-se as TIs dos *switches* e, portanto, terá a rota inversa ao pacote de Interesse correspondente. Antes de encaminhar o pacote de Conteúdo, os *switches* consultam sua TRC para verificar se devem armazenar esse conteúdo em sua TC.

A camada de Estratégia (*Estrategy*) é responsável por fazer escolhas otimizadas no encaminhamento de pacotes Interesse, por exemplo, o envio de pacotes por diferentes portas simultaneamente a fim de se obter um menor tempo de resposta ao Interesse. Além disso, também é responsável pela retransmissão de pacotes Interesse pendentes na TI. Na proposta original do CCN [Jacobson et al. 2009] é reconhecido que não existe uma estratégia de encaminhamento que seja melhor para todos os casos. Assim, a intenção



(a) Encaminhamento de Interesse.

(b) Encaminhamento de Conteúdo.

Figura 2. Diagrama de atividades ilustrando o encaminhamento de pacotes na rede SDCCN.

inicial era de acrescentar em cada entrada da TE (FIB) um programa, escrito em uma linguagem especializada em decisões sobre encaminhamento de pacotes, que determinaria como o encaminhamento seria feito.

Um dos grandes benefícios de utilizar uma abordagem SDN é a fácil implementação da camada de Estratégia, promovendo grande flexibilidade no encaminhamento de pacotes. Essa facilidade existe porque a ideia da camada de Estratégia é muito similar ao da abordagem SDN. Em ambos, o *switch* suporta uma linguagem especializada em decisões sobre encaminhamento (por exemplo, OpenFlow e POF) e um agente externo define como será o encaminhamento, instalando regras nas tabelas dos *switches* utilizando essa linguagem.

Na proposta apresentada neste artigo, um conjunto de mensagens e ações do protocolo OpenFlow são extendidas e utilizadas para fornecer roteamento programável. Uma nova mensagem foi adicionada ao protocolo OpenFlow para implementação da camada de Estratégia. Seu formato pode ser observado na Figura 3. Os números abaixo dos campos identificam seus respectivos tamanhos em *bytes*. A mensagem é detalhada a seguir:

- INTEREST_INFO: Mensagem utilizada pelo *switch* para sinalizar ao controlador a respeito de uma requisição Interesse que não foi correspondida dentro de algum intervalo de tempo configurável. O campo “cmd” atualmente possui somente o comando OFPII_TIMEDOUT, indicando que o pacote de Interesse não foi correspondido durante esse intervalo. O campo “nconteudo” identifica o nome do conteúdo cujo interesse não foi correspondido.

3.2. Cache

Vários estudos vêm sendo feitos para encontrar a melhor solução de *cache* em redes CCN ([Rossi and Rossini 2011], [Chai et al. 2012], [Psaras et al. 2012]). Muitos desses estu-

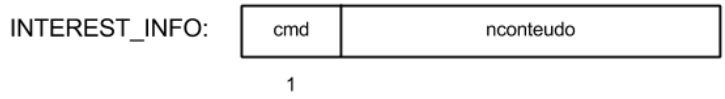


Figura 3. Formato da mensagem de encaminhamento adicionada.

dos propõem a utilização de políticas de substituição de *cache* diferentes para nós localizados nas bordas ou no núcleo das redes CCN. Nas redes CCN, entretanto, a alteração de políticas de *cache* é custosa, pois o código embarcado nos roteadores CCN precisa ser alterado. A abordagem SDN facilita essa implementação pois o controlador possui uma visão geral da topologia da rede e da distribuição de conteúdos. Mover as decisões de *cache* para o plano de controle, através de uma interface padronizada, facilita a experimentação e a comparação das diferentes políticas de *cache*.

Conforme ilustrado no diagrama de atividades da Figura 2, o *switch* somente armazenará um conteúdo na TC se possuir uma regra equivalente na TRC. Essa equivalência é feita comparando o nome do conteúdo do pacote e das entradas da TRC. A TRC também suporta o uso de regras coringa. Caso a TC do *switch* fique cheia, novos conteúdos não poderão ser armazenados. Quando a TC ficar cheia ou atingir um limite configurável, o *switch* avisará ao controlador através de uma mensagem padronizada. O controlador poderá remover conteúdos armazenados na TC a seu critério, possibilitando assim a definição de algoritmos de substituição de *cache* no plano de controle.

O controlador também pode instalar conteúdos na TC dos *switches* de maneira pró-ativa, ou seja, sem que um pacote Interesse correspondente tenha trafegado pelo *switch*. Essa abordagem é interessante, por exemplo, quando ocorre a previsão de que um determinado conteúdo terá grandes chances de ser requisitado por múltiplos usuários da rede, como nos lançamentos de um conteúdo "*blockbuster*", ou durante eventos de mobilidade de usuários que estejam consumindo conteúdos de forma contínua (vídeo, por exemplo). No primeiro caso, estratégias de balanceamento e distribuição de carga na rede podem ser adotadas, afim de aproximar o conteúdo e os usuários interessados nele, levando para a rede o conceito das CDNs. No segundo caso, usuários de dispositivos móveis podem ter acesso à rede por pontos distintos na sua borda, e, com isso, perder o *cache* do conteúdo que estava consumindo, ocasionando uma perda de desempenho, por exemplo. Como o controlador é centralizado e tem a visão da rede, ele pode, proativamente antecipar-se a esse movimento e ordenar o *caching* do conteúdo no novo ponto de acesso à rede desses usuários. Um raciocínio inverso pode ser aplicado quando usuários de dispositivos móveis são os geradores de algum conteúdo, por exemplo durante a transmissão de um *twitcasting* em uma manifestação popular.

Novas mensagens foram adicionadas para suportar a programabilidade de *cache*, ou seja, possibilitar que o plano de controle gerencie as tabelas TC e TRC dos *switches*. O formato dessas mensagens pode ser observado na Figura 4. Os números abaixo dos campos identificam seus respectivos tamanhos em *bytes*. As mensagens são detalhadas a seguir:

- CACHE_MOD: Mensagem utilizada para modificar a TRC, possibilitando adicionar, remover e modificar regras de *cache*. O campo "cmd" identifica um dos possíveis comandos: OFPCAC_ADD, OFPCAC_MODIFY e OFPCAC_DELETE.

O campo “ext” é utilizado para a criação de regras coringa. Se “ext” for igual a 1, a busca na TRC utilizando o nome de conteúdo será exata. Se “ext” for igual a 0, serão selecionadas todas as entradas da TRC que possuírem nome do conteúdo como um prefixo do nome buscado. Os campos “stimeout” e “htimeout” identificam o tempo até que a regra seja removida. O primeiro é o tempo máximo sem uso da regra (*soft timeout*) e o segundo o tempo máximo, independente do uso (*hard timeout*). O campo “index” identifica o número da entrada da TRC. O campo “nconteudo” contém o nome do conteúdo.

- CS_MOD: Mensagem utilizada para modificar a TC, possibilitando adicionar, remover e modificar conteúdos. Possui funcionamento similar ao CACHE_MOD.
- CACHE_FULL: Mensagem utilizada para que o *switch* possa indicar ao controlador que sua TC está cheia ou que está quase cheia. O comando OFPCFAC_CRIT é usado para o primeiro caso e OFPCFAC_WARN para o segundo. O campo “cmd” identifica um dos possíveis comandos: OFPCFAC_CRIT OFPCFAC_WARN. O campo “tentradas” identifica o total de entradas. O campo “nentradas” identifica o número de entradas ocupadas.
- CACHE_INFO: Mensagem utilizada para que o controlador requisite dados sobre a TRC do *switch*. O campo “cmd” identifica um dos possíveis comandos: OFPCIAC_REQUEST e OFPCIAC_REPLY. O comando OFPCIAC_REQUEST é utilizado para a solicitação do controlador para o *switch* e o comando OFPCIAC_REPLY para resposta do *switch* para o controlador. O campo “nentradas” identifica o número de entradas em uso na TRC do *switch*. O campo “entradas” possui informações sobre todas as entradas, tais como índice, nome do conteúdo e data de criação.
- CS_INFO: Mensagem utilizada para que o controlador requisite dados sobre a TC do *switch*. Possui funcionamento similar ao CACHE_INFO.

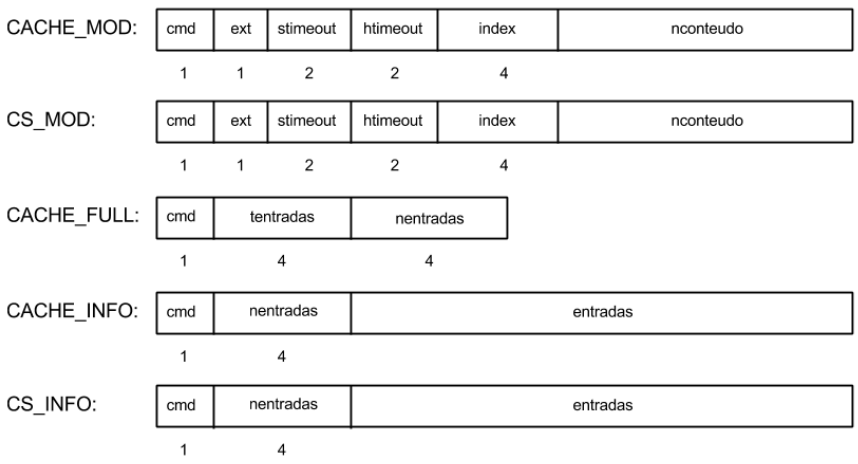


Figura 4. Formato das mensagens de cache.

4. Implementação

Um ambiente de experimentação para rápida prototipação e experimentação em redes CCN foi desenvolvido a partir do mininet. Nele é possível criar nós CCN, que rodam o

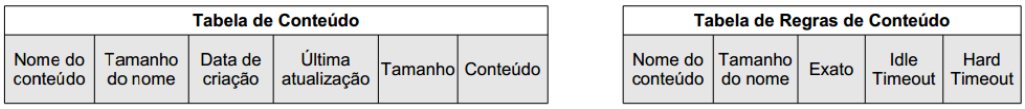


Figura 5. Formato das tabelas TC e TRC.

código oficial do projeto CCNx, e os *switches* da solução desenvolvida. A implementação atual suporta todas as mensagens de encaminhamento e *cache*, descritos na Seção 3. As funcionalidades relacionadas à segurança serão incluídas em versões futuras.

A comunicação entre os planos de dados e de controle é feita através do protocolo *Protocol-oblivious Forwarding* (POF). A escolha do POF foi feita com base na necessidade de roteamento por nomes de conteúdo de tamanhos variáveis. O POF faz com que o plano de dados não necessite reconhecer nenhum formato de pacote. Seu processamento de fluxos é similar ao OpenFlow, entrentando no OpenFlow a busca é realizada utilizando campos de protocolos conhecidos. No POF, a busca é feita pela tupla {deslocamento, tamanho, *bytes*}. O POF também suporta todas as mensagens do protocolo OpenFlow.

Com a utilização do POF as mensagens CCN podem ser implementadas de maneira agnóstica nos elementos de rede. Ou seja, é possível mudar o formato dos pacotes CCN sem alterar o código dos *switches* CCN. Mensagens como descoberta de vizinhos, troca de chaves, sincronização de repositório, etc, poderão ser alteradas sem modificar o plano de dados.

No plano de dados, o *switch* POF foi estendido para suportar o novo processamento de fluxo, representado pelo diagrama de atividades da Figura 2. Além da TE já existente na implementação do POF, as tabelas TC e TRC foram adicionadas. A tabela TI não foi implementada e o encaminhamento de pacotes de conteúdo é realizado instalando regras na TE. Um estudo maior sobre a melhor forma de encaminhamento de conteúdo na arquitetura SDCCN é necessário, levando em consideração questões como desempenho, flexibilidade e suporte a algorítmos de roteamento. O formato atual das duas tabelas pode ser observado na Figura 5.

5. Prova de Conceito

Como prova de conceito da proposta SDCCN utilizamos nosso ambiente de experimentação para realizar dois experimentos:

- O primeiro experimento foi realizado para **demonstrar a programabilidade do cache e o impacto da popularidade do conteúdo**. Comparamos dois algoritmos de substituição de cache: LRU e FIFO variando a popularidade dos conteúdos. Esses algoritmos foram programados no controlador utilizando as mensagens descritas na Seção 3.
- O segundo experimento visa mostrar os **benefícios da programabilidade do *cache* na rede em um cenário de mobilidade**.

Ressaltamos que durante estes experimentos, nenhuma modificação foi feita no código dos *switches* CCN.

5.1. Programabilidade do Cache e popularidade do conteúdo

A topologia mostrada na Figura 6 foi criada no ambiente de experimentação desenvolvido para a arquitetura SDCCN. Os enlaces entre o *switch* CCN e os *hosts* foram definidos

com atraso de 10ms. Um *host* `h1` roda uma aplicação *ccnpingserver* e o *host* `h2` faz as solicitações *ccnping* na rede CCN. Devido aos atrasos dos enlaces, uma requisição de *ping* que passa pelos dois *hosts* deve ter um RTT próximo de 40ms. Caso o *switch* tenha o conteúdo do *ping* requisitado armazenado em sua TC, a requisição deve ter um RTT próximo de 20ms.

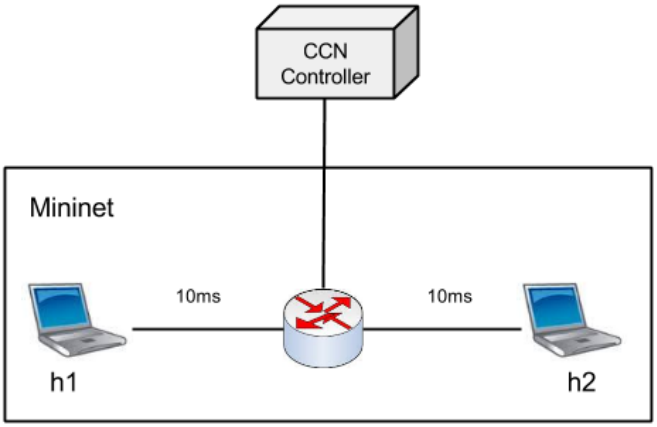


Figura 6. Arquitetura do ambiente de experimentação utilizando *ccnping*.

Para execução deste experimento, a TC do *switch* foi limitada em 50 entradas e o limite para envio de alertas de aviso (OFPCFAC_WARN) foi configurado como 45 entradas. Foram adicionadas regras de encaminhamento na TE e regras para armazenamento de qualquer conteúdo com o prefixo "ccnx:/ufes" na TRC. O prefixo "ccnx:/ufes" foi utilizado como prefixo do *ccnping*. Os nomes dos conteúdos gerados pelo *ccnping* seguiram o padrão ccnx:/ufes/[N], onde [N] é um número inteiro positivo gerado aleatoriamente. Esse número foi gerado utilizando uma distribuição *Zipf-like* para caracterizar a popularidade dos conteúdos com parâmetros α igual a 0.6, 0.7, 0.8, 0.9 e 1.0. Os valores de α foram selecionados de acordo com [Breslau et al. 1999]).

No início do experimento, a TC do *switch* encontra-se vazia, sem conteúdos armazenados, e a TRC possui uma entrada para armazenamento dos conteúdos do *ccnping*. Assim, cada conteúdo que passa pelo *switch* será armazenado em sua TC. Quando o número de conteúdos armazenados chegar ao limite (45 entradas no caso desse experimento), o *switch* envia uma mensagem CACHE_FULL para o controlador. O controlador, ao receber a mensagem CACHE_FULL, envia uma mensagem CS_INFO para obter dados dos conteúdos armazenados na TC do *switch*, tais como data de criação e data da última referência de cada conteúdo. Em seguida, remove-se uma entrada da TC seguindo os algoritmos de substituição de *cache* FIFO e LRU, enviando uma mensagem CS_MOD.

Para cada algoritmo de substituição foram geradas 1000 requisições de Interesse. O experimento foi repetido 5 vezes para cálculo de média e intervalo de confiança (consideramos o nível de confiança de 95%). A Tabela 1 mostra a taxa de acerto do *cache* para cada valor de α para os algoritmos FIFO e LRU respectivamente. É possível observar que o algoritmo LRU obteve uma taxa de acerto ligeiramente maior para todos os valores de α. Entretanto, o impacto da popularidade do conteúdo (quanto maior o α, maior a popularidade do conteúdo) foi muito mais significativo do que os algoritmos de *cache*. A diferença entre os algoritmos de *cache* foi por volta de 4% para todos valores

Tabela 1. Comparação do RTT médio e taxa de acerto das políticas de cache LRU e FIFO.

α	Taxa de acerto – FIFO	Taxa de acerto – LRU
0,6	0,173 ± 0,011	0,189 ± 0,010
0,7	0,215 ± 0,010	0,240 ± 0,009
0,8	0,284 ± 0,009	0,312 ± 0,011
0,9	0,362 ± 0,014	0,415 ± 0,016
1,0	0,451 ± 0,022	0,506 ± 0,009

de α, enquanto que houve um aumento de 12% com a variação de α = 0.6 a α = 1.0 para ambos os algoritmos de *cache*.

A Figura 7 mostra a quantidade de requisições que tiveram RTT no intervalo de 0–30ms para todos valores de α. É possível observar que o desempenho do algoritmo LRU foi ligeiramente superior, como era esperado. No entanto, confirmou-se que se o conteúdo for muito popular, por exemplo variando-se α (α = 0.6 a α = 1.0), há um aumento consideravel no número de requisições com RTTs pequenos (*cache hits*). Considerando a política FIFO como exemplo, esse numero vai de 150 para 450. Esses resultados mostram que o impacto da popularidade do conteúdo tende a ser mais efetivo no ganho de desempenho do que a politica de substituição de cache. Entretanto, ressaltamos aqui que a principal contribuição deste experimento está na facilidade com que o mesmo pôde ser realizado usando-se a arquitetura SDCCN, sem necessidade de reprogramação do CCN SW.

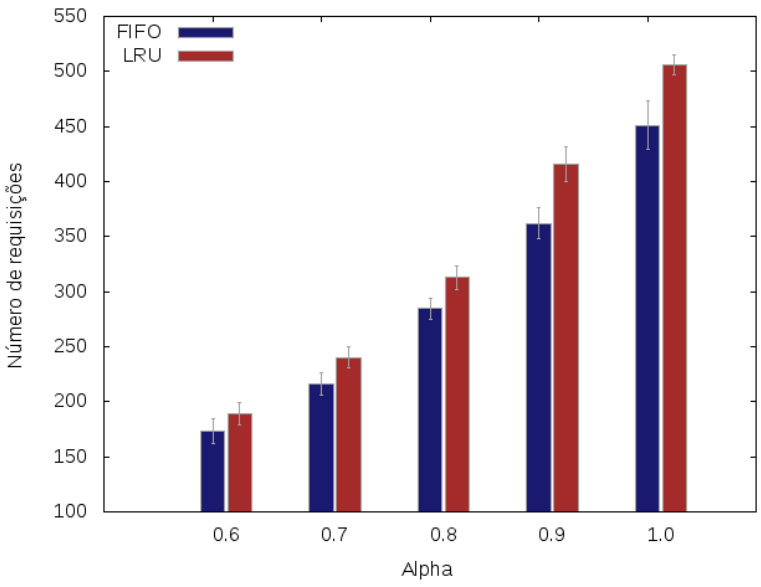
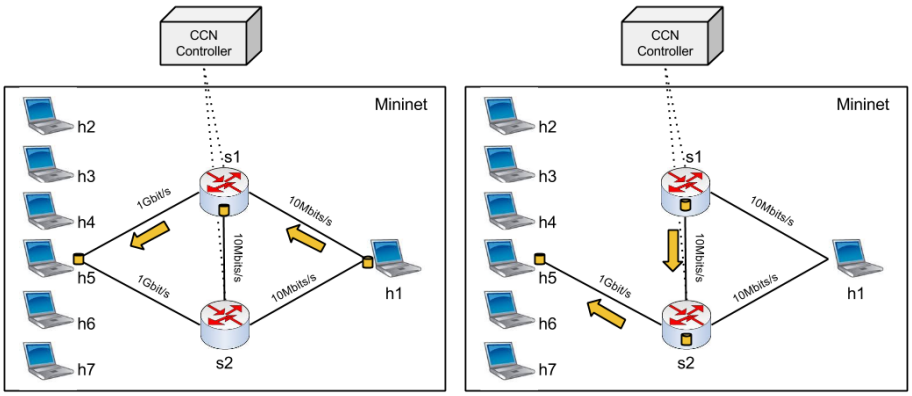


Figura 7. Comparação do RTT nas políticas de cache LRU (vermelho) e FIFO (azul).

5.2. Benefícios do cache na rede

Para ilustrar os benefícios do cache na rede utilizando o SDCCN, nos baseamos no experimento *Content Distribution Efficiency* de [Jacobson et al. 2009]. A Figura 8 ilustra a



(a) Arquitetura inicial do ambiente SDCCN.

(b) Arquitetura do ambiente SDCCN após a migração dos cliente para `s2`.

Figura 8. Arquitetura do ambiente de experimentação para comparação de tempo de *download* em redes TCP/IP e CCN.

topologia da rede. O nó `h1` atua como um gerador de conteúdos e os demais nós da rede fazem *download* (Interesse) desse conteúdo de forma concorrente. Inicialmente todos os nós estão conectados no *switch* `s1`, foram realizadas seis rodadas seguindo o experimento Content Distribution Efficiency [Jacobson et al. 2009]:

- Na primeira rodada apenas um nó manifesta interesse pelo conteúdo;
- Na segunda rodada dois nós manifestam interesse pelo conteúdo de forma simultânea;
- Na terceira rodada tres nós manifestam interesse pelo conteúdo de forma simultânea; e assim por diante.

O conteúdo da TC dos *switches* `s1` e `s2` não foi apagado entre as rodadas, mantendo-se o conteúdo em *cache* durante sucessivas requisições de Interesse. Para cada rodada foi calculado o tempo acumulado de *download* do conteúdo requisitado por todos os interessados, ou seja, desde o início dos *downloads* da rodada até que o último interesse da rodada seja satisfeito.

Neste experimento simulamos um cenário de mobilidade, conforme descrito na seção anterior. Antes da última rodada, os consumidores perdem a conexão com o *switch* `s1`, e reconectam-se à rede conexão apenas com `s2`. Na rede CCNx, a TC de `s2` iniciar-se-á vazia e as requisições deverão passar por um enlace de 10Mbps. No ambiente SDCCN, o controlador antecipa-se a essa migração e proativamente ordena a instalação do conteúdo na TC do switch `s2`. É importante ressaltar aqui que não necessariamente faz-se uma cópia de todo o conteúdo. No exemplo deste experimento, como o conteúdo é um arquivo que está sendo requisitado sequencialmente (em pedaços, ou *chunks*, de acordo com a arquitetura CCN), se antes da mobilidade o usuário estiver consumindo o pedaço x, pode-se, proativamente, ordenar a `s2` o *caching* a partir do pedaço x+1. Como os pedaços têm tamanho limitado, o tempo de cópia de um pedaço entre um cache e outro será muito pequeno, o que viabiliza sua cópia durante o evento de mobilidade.

A topologia foi configurada para três modelos de arquitetura: (i) CCNx, utilizando o ambiente de experimentação criado mas utilizando um nó CCNx no lugar do *switch*; (ii)

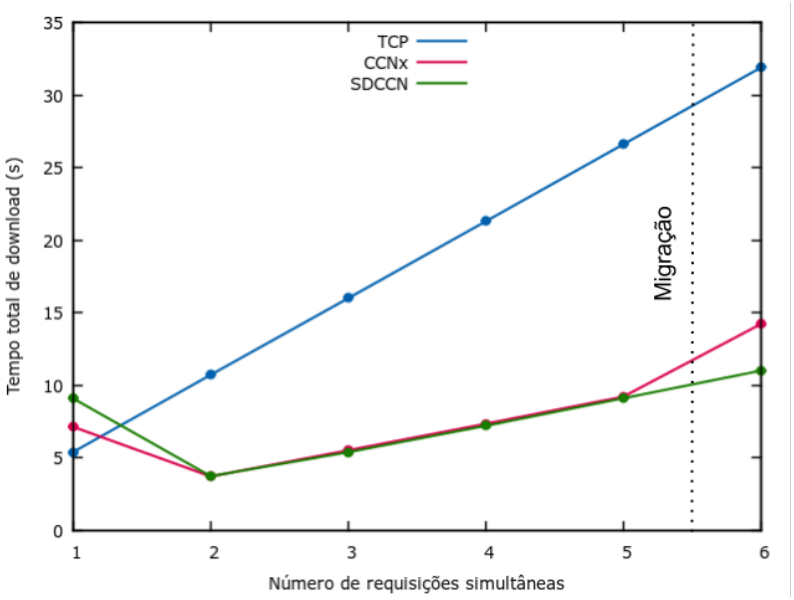


Figura 9. Comparação de tempo e número de *downloads* simultâneos em redes TCP/IP e CCN.

SDCCN, utilizando o ambiente de experimentação criado com nós CCNx e *switches* SDCCN; (iii) TCP/IP, em uma rede simulada com suporte a pilha TCP/IP. Nos ambientes CCN foi utilizado o CCNx 0.8.2 e as ferramentas `ccnr`, `ccnputfile` e `ccngetfile` para realizar a transferência de um arquivo de 6MB entre o nó produtor e os nós consumidores. Na rede TCP/IP, o nó `h1` serve um arquivo de 6MB através de um servidor HTTP com suporte a SSL. A largura de banda do enlace entre os *switches* e o nó servidor de arquivos foi configurado com 10 Mbps para que houvesse uma saturação proposital do enlace.

O experimento foi repetido 5 vezes para cálculo de média e intervalo de confiança (consideramos o nível de confiança de 95%). A comparação entre o tempo de *download* e número de *downloads* simultâneos para os ambientes TCP, CCNx e SDCCN pode ser vista na Figura 9. O intervalo de confiança é muito pequeno e por isso não pode ser observado no gráfico na escala utilizada. No ambiente TCP, os conteúdos têm que percorrer um caminho maior para todas as requisições, atravessando o enlace de 10Mbps para acessar o conteúdo do provedor hospedado em `h1`. Assim, quanto maior o número de *downloads* simultâneos, mais esse enlace ficará saturado e maior será o tempo necessário para a conclusão do *download*. Nas redes CCN, durante a primeira requisição os conteúdos são armazenados na TC (Tabela de Conteúdo) do *switch* `s1`. Assim, as demais requisições para aquele conteúdo que passam por `s1` não atravessam o enlace de 10Mbps. Como o enlace de 1Gbps/s não fica saturado, o tempo de *download* se mantém constante.

Na última rodada, o tempo de *download* aumenta no ambiente CCNx pois o conteúdo que originalmente estava em `h1` é migrado para `s2` e a primeira requisição deve atravessar o enlace de 10Mbps. No ambiente SDCCN, o tempo de *download* se mantém constante pois o controlador SDCCN ordena, proativamente, q cópia do conteúdo do *cache* de `s1` para o *cache* de `s2` (Figura 9(b)).

Podemos observar que na primeira requisição nossa solução SDCCN obteve um tempo de resposta ligeiramente pior do que o CCNx original. Isso ocorre porque o *switch*

SDCCN, baseado no POF, roda em espaço de usuário no sistema operacional e não foram feitas otimizações referentes ao encaminhamento dos conteúdos. As demais requisições que foram feitas antes momento da migração tiveram resultados iguais, pois o conteúdo é retornado diretamente a partir do *cache*. O aumento de aproximadamente 1,3s a cada rodada, para ambos CCN e SDCCN, ocorre devido à implementação das ferramentas de repositório e à geração dos pedaços de conteúdo. Porém, ainda assim, este é um aumento consideravelmente menor que na arquitetura TCP/IP.

6. Conclusão

As Redes Orientadas à Conteúdo (CCN) promovem uma melhor abstração para o uso da Internet. Entretanto, incorporaram alguns dos problemas existentes na arquitetura atual: a experimentação ainda é custosa, já que muitas vezes envolve a reprogramação do código dos *switches*, as decisões de *cache* e as estratégias de encaminhamento são rígidas por serem amarradas ao código dos *switches*. Com a separação dos planos de dados e controle, é possível prover a programabilidade do *cache* e fornecer uma camada de estratégia flexível.

Este artigo propôs uma abordagem SDN para Redes Orientadas à Conteúdo (CCN). O plano de controle é composto por um controlador logicamente centralizado e o plano de dados é composto por *switches* CCN. O controlador programa as tabelas de regras de encaminhamento e estratégias de *cache* presentes nos roteadores CCN. Decisões tais como quais conteúdos cada roteador irá armazenar e como será a política de substituição de *cache* são programáveis. Assim, nós presentes no núcleo e nas bordas da rede podem utilizar diferentes políticas de substituição de cache, que ainda podem ser alteradas dependendo do comportamento do tráfego. Conteúdos também podem ser instalados dinamicamente em um switch CCN a fim de otimizar a entrega de um determinado conteúdo. Por fim, um ambiente de experimentação foi criado a partir do Mininet para prover fácil experimentação.

Resultados experimentais foram realizados para demonstrar os benefícios da arquitetura proposta. Primeiramente foi feita a comparação de dois algorítmos de substituição de *cache* (LRU e FIFO), definidos no plano de controle. Como esperado, o LRU obteve um desempenho melhor. A instalação de conteúdos de forma pró-ativa também foi explorada em um cenário contendo a mobilidade de usuários consumidores de um conteúdo da rede. Nossa solução acrescenta flexibilidade ao CCN e pode apresentar um melhor desempenho em alguns cenários em relação ao CCNx e ao TCP/IP.

Como trabalhos futuros, iremos abordar questões de segurança na implementação e estudar melhor a utilização da TI. Pretendemos também estudar estratégias de fragmentação dos pacotes e algoritmos para roteamento inter-domínio, fornecer implementações de algoritmos intra-domínio atualmente utilizados (como o NSLR [Hoque et al. 2013]) e propor estratégias de localização de *cache* mais inteligentes, explorando a vantagem do controlador possuir uma visão global da rede.

Referências

Breslau, L., Cue, P., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web caching and zipf-like distributions: Evidence and implications. In *Proceedings of the Eigh-*

teenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, pages 126–134. IEEE.

Chai, W. K., He, D., Psaras, I., and Pavlou, G. (2012). Cache "less for more"in information-centric networks. In *Proceedings of the 11th International IFIP TC 6 Conference on Networking - Volume Part I*, IFIP'12, pages 27–40. Springer-Verlag.

Cisco (2013). Cisco visual networking index: Forecast and methodology, 2013–2018.

Detti, A., Blefari Melazzi, N., Salsano, S., and Pomposini, M. (2011). Conet: A content centric inter-networking architecture. In *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking*, ICN '11, pages 50–55. ACM.

Hoque, A. K. M. M., Amin, S. O., Alyyan, A., Zhang, B., Zhang, L., and Wang, L. (2013). Nlsr: Named-data link state routing protocol. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Information-centric Networking*, ICN '13, pages 15–20. ACM.

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N. H., and Braynard, R. L. (2009). Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '09, pages 1–12. ACM.

Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6. ACM.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74.

Nguyen, X. N., Saucez, D., and Turletti, T. (2013). Efficient caching in content-centric networks using openflow. In *INFOCOM, 2013 Proceedings IEEE*, pages 1–2.

Psaras, I., Chai, W. K., and Pavlou, G. (2012). Probabilistic in-network caching for information-centric networks. In *Proceedings of the Second Edition of the ICN Workshop on Information-centric Networking*, ICN '12, pages 55–60. ACM.

Rossi, D. and Rossini, G. (2011). Caching performance of content centric networks under multi-path routing (and more). *Technical Report, Telecom ParisTech*.

Song, H. (2013). Protocol-oblivious forwarding: Unleash the power of sdn through a future-proof forwarding plane. In *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, HotSDN '13, pages 127–132. ACM.

Syrivelis, D., Parisis, G., Trossen, D., Flegkas, P., Sourlas, V., Korakis, T., and Tassiulas, L. (2012). Pursuing a software defined information-centric network. In *Proceedings of the 2012 European Workshop on Software Defined Networking*, EWSDN '12, pages 103–108. IEEE Computer Society.

Vahlenkamp, M., Schneider, F., Kutscher, D., and Seedorf, J. (2013). Enabling information centric networking in ip networks using sdn. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN for*, pages 1–6.