Mini-CCNx: prototipagem rápida para Redes Orientadas a Conteúdo baseadas em CCN

Carlos Manuel Silvestre Cabral[1], **Christian Esteve Rothenberg**[2], **Maurício Ferreira Magalhães**[1]

[1]Faculdade de Engenharia Elétrica e Computação
Universidade Estadual de Campinas
Campinas, SP – Brasil (UNICAMP)

[2]Fundação CPqD – Centro de Pesquisa e Desenvolvimento em Telecomunicações
Campinas, SP – Brasil

{cabral,mauricio}@dca.fee.unicamp.br, esteve@cpqd.com.br

Abstract. *Experimentally driven research in Information-Centric Networking (ICN) is crucial to the evaluation of new proposals that bring named pieces of content as the main element of networks. This paper presents a new fast prototyping tool for ICN based on the CCN (Content-Centric Networking) model, Mini-CCNx, that aims at filling an existing gap in generally available experimental platforms. Using container-based emulation and resource isolation techniques, Mini-CCNx appears as a flexible, scalable, high-fidelity, and low-cost tool that enables experiments on emulated networks with hundreds of nodes in a commodity laptop.*

Resumo. *A pesquisa experimental em Redes Orientadas a Conteúdo (ROCs) é crucial para a validação das novas propostas que trazem o conteúdo como elemento central das redes. Este artigo apresenta uma nova ferramenta de prototipagem rápida para as ROCs baseadas no modelo CCN (Content-Centric Networking), o Mini-CCNx, o qual busca preencher uma lacuna existente entre as plataformas de experimentação da área. Utilizando emulação baseada em contêineres e técnicas de isolamento de recursos, o Mini-CCNx se apresenta como uma ferramenta flexível, escalável, de baixo custo, e com comportamento e fidelidade de desempenho compatíveis ao de um sistema real, podendo criar experimentos em redes emuladas com centenas de nós em um simples laptop.*

1. Introdução

Os princípios que guiaram o desenvolvimento da Internet nas décadas de 1960 e 1970 não são mais tão relevantes atualmente como foram no passado. Se originalmente o intuito principal era o compartilhamento remoto de recursos e interação majoritariamente textual entre estações de trabalho e servidores, atualmente o foco está muito mais centrado na obtenção e acesso a conteúdos e serviços multimídia [Plagemann et al. 2006]. Com a tecnologia de redes atual, quando se deseja obter um conteúdo, é necessário fazer um mapeamento entre *o que* um usuário quer e *onde* este conteúdo está, trazendo inúmeros desafios como mobilidade, segurança, fornecimento de dados de maneira *multicast* e outros, desafios esses que não foram previstos originalmente pelos desenvolvedores da Internet.

Com essa motivação, as Redes Orientadas a Conteúdo (ROCs) trazem propostas para endereçar essa mudança de paradigma, colocando o conteúdo como item central das redes [de Brito et al. 2012]. Propostas como NetInf [Dannewitz et al. 2013], PSIRP [PSIRP] e CCN [Jacobson et al. 2012], dentre outras, introduzem inúmeros conceitos e estratégias nesse intuito. Como em toda nova proposta, a pesquisa experimental para as ROCs é crucial para a avaliação da viabilidade de novas idéias.

Ao tentar avaliar estratégias de encaminhamento e novas técnicas de redução de estado nos roteadores de núcleo das ROCs, identificamos uma lacuna existente entre as ferramentas disponíveis nessa área de pesquisa. Não foi encontrada uma ferramenta de baixo custo, com escalabilidade, flexível e com comportamento e fidelidade de desempenho compatíveis aos de um sistema real para a avaliação de cenários e propostas de ROCs. Essa dificuldade para experimentação motivou o desenvolvimento do Mini-CCNx, apresentado nesse artigo. Inspirado nas práticas bem-sucedidas de prototipagem rápida para Redes Definidas por Software (SDNs - *Software-Defined Networks*) [Lantz et al. 2010], o Mini-CCNx é uma ferramenta capaz de construir uma ROC completa, com centenas de nós, em um simples *laptop*, com altíssima flexibilidade, agilidade e configurabilidade. Além disso, ela provê um ambiente de prototipagem com resultados de grande fidelidade, o que permite que aplicações e conceitos testados utilizando o Mini-CCNx migrem de forma suave para ambientes e redes reais. O Mini-CCNx é baseado no modelo CCN (*Content-Centric Networking*) e utiliza a implementação oficial desse modelo, denominada CCNx[CCNx] e usada no Projeto NDN[NDN Testbed], o que traz maior relevância para a ferramenta e para a comunidade atual de usuários do projeto.

O resto do artigo está organizado da seguinte forma. A seção 2 faz uma análise das ferramentas existentes e detalha os objetivos do Mini-CCNx. A seção 3 descreve em detalhes a arquitetura da ferramenta. A seção 4 apresenta uma análise de performance e fidelidade do Mini-CCNx. A seção 5 explica as duas demonstrações a serem feitas no Salão de Ferramentas do SBRC 2013 e mostra onde encontrar a documentação e o código. A seção 6 finaliza o artigo com a conclusão.

2. Objetivos e Motivação

Existem várias plataformas e ferramentas que podem ser utilizadas na pesquisa e desenvolvimento das ROCs, cada uma com seus prós e contras. Idealmente, seria interessante possuir uma ferramenta que reunisse as seguintes características: *(i)* flexibilidade (capacidade de criação ágil de diversos cenários), *(ii)* escalabilidade (criação de topologias com um número suficientemente alto de nós), *(iii)* baixo custo (possibilidade de ser executada em um *laptop* ou *desktop* comum) e *(iv)* realismo (o comportamento apresentado deve ser compatível com o de um ambiente real e o código utilizado na ferramenta deve ser o mesmo utilizado posteriormente na rede real).

2.1. Análise das ferramentas existentes

Podemos dividir tais plataformas e ferramentas em, ao menos, três categorias:

Simuladores. São flexíveis, escaláveis, e em geral têm baixo custo. Como exemplos de simuladores específicos para ROCs, pode-se citar o *ndnSIM*[ndnSIM] e o ccnSim[ccnSim]. Porém, ambas as ferramentas apresentam as desvantagens de qualquer simulador: *(i)* não apresentam realismo e *(ii)* os modelos de hardware, pilhas de

protocolos e geração de tráfego de um simulador, por melhor que sejam implementados, não representam com total fidelidade todas as características reais dos mesmos.

Testbeds. São infraestruturas experimentais que oferencem recursos reais ou virtualizados para testes em ambientes reais. Podem ser compartilhados entre vários pesquisadores, como o *testbed* oficial do NDN[NDN Testbed] ou especificamente criados para um certo projeto, como por exemplo a criação de um ambiente local com máquinas virtuais representando os *hosts*, *switches* e roteadores. *Testbeds* apresentam realismo, ou seja, executam código real em ambientes com pilha de protocolos e tráfego real. Porém, em geral, apresentam *(i)* alto custo de criação e manutenção e *(ii)* reduzida flexibilidade e escalabilidade na definição de cenários e topologias personalizadas.

Emuladores. Como os simuladores, possuem flexibilidade na definição de topologias, além de um baixo custo de hardware. Como os *testbeds*, apresentam realismo, pois executam código real (aplicações, kernel de SO, etc). O Mini-CCNx, descrito nesse artigo, se encaixa nessa categoria e é especificamente focado nas ROCs.

A Tabela 1 sumariza as características analisadas e inclui duas outras: facilidade de configuração da plataforma e a possibilidade de configuração de parâmetros de *links*, como atraso e perda. Dada a inexistência de uma solução que atenda a todos os requisitos acima, optou-se pelo desenvolvimento de uma nova ferramenta, o Mini-CCNx.

Tabela 1. Resumo das características das plataformas de teste para ROCs

	Simuladores Ex. ndnSIM e ccnSim	***Testbeds*** Ex. *Testbed* NDN	**Emuladores** Ex. Mini-CCNx
Flexibilidade	Alta	Baixa	Alta
Escalabilidade	Alta	Baixa	Alta
Custo	Baixo	Alto	Baixo
Realismo	Não	Sim	Sim
Facilidade de Configuração	Média	Baixa	Alta
Configuração de Parâmetros de *Links*	Sim	Com Restrições	Sim

3. Arquitetura

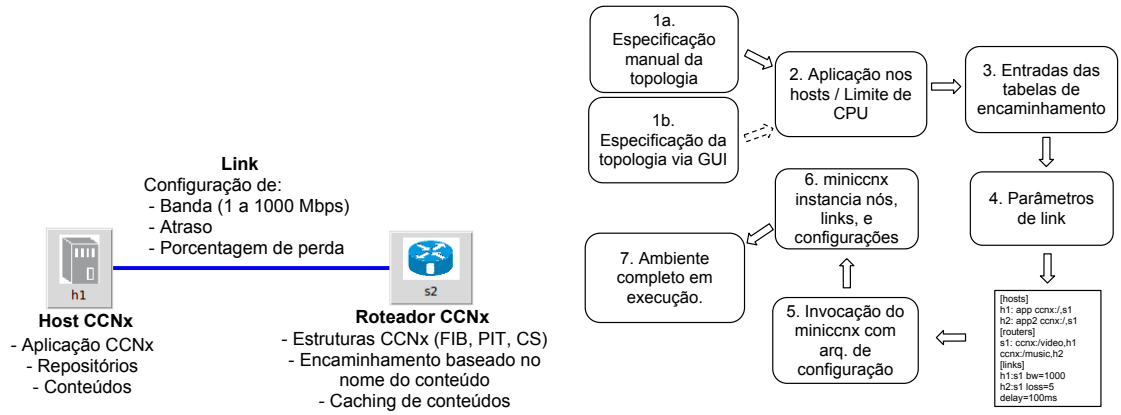
O Mini-CCNx reúne todas as características analisadas na seção anterior. Através da edição de um arquivo texto é possível criar as mais diferentes topologias e cenários CCN de maneira flexível e rápida. O custo para executá-lo é baixo, sendo possível levantar centenas de nós em um simples *laptop*. O mesmo código executado na ferramenta poderá ser posteriormente utilizado em uma rede real. Além disso, é possível configurar rapidamente, com um arquivo de configuração, parâmetros como atraso, banda e perda dos *links*.

O Mini-CCNx é um *fork* do Mininet-HiFi [Handigol et al. 2012] (originalmente proposto para redes OpenFlow) que adiciona uma série de mecanismos e novas classes para instanciar ROCs baseadas no modelo CCN utilizando o código oficial do Projeto CCNx. Estruturas de dados específicas do modelo como tabelas de encaminhamento e repositórios foram implementadas. Também foi implementado todo um mecanismo automático de criação dinâmica de conectividade IP sobre a qual as conexões CCNx operam, tornando isso totalmente transparente ao usuário. Além disso, várias ferramentas de suporte, configuração e construção de topologias também foram desenvolvidas com o intuito de auxiliar o pesquisador no desenvolvimento de seus cenários de teste.

O Mini-CCNx utiliza emulação baseada em contêineres no nível de Sistema Operacional [Soltesz et al. 2007]. Isso permite que grupos de processos tenham visões independentes dos recursos do sistema (*ID*s de processos, sistemas de arquivos e interfaces de rede) apesar de compartilharem um único kernel, alcançando assim uma escalabilidade maior do que um sistema baseado em máquinas virtuais em um único *hypervisor*. O Mini-CCNx também incorpora o conceito de isolamento de desempenho e de provisão de recursos utilizando os *cgroups*[1] do kernel Linux. Com isso, é possível configurar, para cada nó, a porcentagem máxima de uso de CPU. Todo esse isolamento de recursos traz ao Mini-CCNx a garantia de uma grande fidelidade nos experimentos.

3.1. Visão Geral

O Mini-CCNx conta com três componentes principais: *hosts* CCNx, roteadores CCNx e *links*. A Figura 1(a) mostra um resumo das funcionalidades de cada componente.



(a) Componentes Mini-CCNx (b) Fluxo de trabalho de experimentação

Figura 1. Visão geral e fluxo de trabalho

***Host* CCNx:** Responsável por executar as aplicações CCNx. Pode ser implementado pela classe *CCNHost* (sem limitação de recursos) ou pela classe *CPULimitedCCNHost* (com limitação de recursos). Uma aplicação executando em um *host* pode se comportar como cliente quando expressa o interesse por um conteúdo através de uma mensagem *Interest*, ou como servidor, quando retorna um *Content Object* em resposta a um *Interest* recebido. Os *hosts* também são responsáveis por armazenar conteúdos em seus repositórios.

Roteador CCNx: Implementado pela classe *CCNRouter*. Responsável por fazer o encaminhamento dos pacotes *Interest* e *Content Object* e também o *caching* de dados.

***Links*:** Responsáveis por ligar, ponto-a-ponto, os nós CCNx (*hosts* e roteadores). A classe *Link* foi adaptada para suportar a conexão entre nós CCN. Pode-se inserir, para cada *link*, os seguintes parâmetros: *(i) bw=<1-1000>* (banda, em Mbps, do *link*), *(ii) delay=<0-10000>ms* (atraso, em ms, do *link*) e *(iii) loss=<0-100>* (porcentagem de perda de pacotes no *link*).

Após configurar e iniciar o Mini-CCNx, é possível obter um terminal para cada *host* ou roteador, examinar e alterar suas tabelas de encaminhamento, adicionar conteúdos, executar aplicações personalizadas baseadas em conteúdo e coletar dados e

[1] http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

métricas desejados. É importante notar que *hosts* e roteadores utilizam a implementação oficial mais recente do Projeto CCNx, executando o *daemon ccnd*. Caso o usuário atualize a versão desse código, o Mini-CCNx transparentemente utilizará essa nova versão, sem necessidade de atualização ou recompilação.

3.2. Fluxo de trabalho utilizando o Mini-CCNx

A Figura 1(b) exemplifica o típico fluxo de utilização do Mini-CCNx.

1. A especificação da topologia é feita editando o arquivo de configuração do Mini-CCNx especificando *hosts*, roteadores e os *links* entre eles (etapa *1a* na Figura 1(b)). Opcionalmente, pode-se utilizar a ferramenta gráfica `miniccnxedit` fornecida para gerar um template desse arquivo (etapa *1b* na Figura 1(b)).

2. No próprio arquivo de configuração, é possível especificar qual a aplicação será executada automaticamente em cada *host*. Opcionalmente, pode-se definir o limite de uso de CPU, em porcentagem, para cada nó.

3. Ainda no arquivo de configuração, pode-se inserir entradas nas tabelas de encaminhamento dos *hosts* e roteadores.

4. Pode-se especificar ainda parâmetros para cada *link* (banda, perda ou atraso). Ao fim dessa etapa, o arquivo de configuração estará concluído.

5. A ferramenta `miniccnx` deve ser invocada, tendo como argumento o arquivo de configuração criado.

6. A ferramenta faz o *parsing* das configurações, instancia os nós, configura as entradas nas tabelas de encaminhamento, aplica os parâmetros especificados para cada *link* e executa as aplicações nos *hosts*.

7. Agora o ambiente está criado e executando, permitindo ao usuário a interação dinâmica com as aplicações baseadas em conteúdo. Opcionalmente, ele pode coletar quaisquer métricas e *logs* que sejam necessários para a avaliação de seus cenários.

4. Análise de Desempenho e Fidelidade

Todos os testes foram feitos utilizando um *laptop* mediano, representante típico de um hardware de uso pessoal (processador Intel Core I5 2410M e com 4GB de memória RAM). A idéia é justamente analisar o desempenho e a fidelidade do Mini-CCNx ao se utilizar recursos de baixo custo. Serão analisadas as seguintes dimensões de desempenho da ferramenta: *(i)* escalabilidade, *(ii)* coerência e *(iii)* fidelidade ante experimentos reais[2].

4.1. Escalabilidade

Primeiramente, será analisada a escalabilidade da ferramenta em termos de quantos nós e enlaces ela é capaz de instanciar simultaneamente. Para isso, foram escolhidas duas topologias representativas, *full mesh* e linear. Na topologia *full mesh* todos os nós possuem conexões para todos os outros nós. Já na topologia linear (Tabela 2(b)), os nós são ligados linearmente entre si. Tais configurações foram escolhidas justamente por representarem os dois extremos de conectividade entre nós - qualquer outra topologia apresentará um nível de conectividade que estará entre esses dois extremos.

As Tabelas 2(a) e 2(b) mostram o número de nós instanciados (*CCN Routers*), a memória utilizada especificamente pelo Mini-CCNx, a memória utilizada especificamente

[2]Análise de isolamento em https://www.dropbox.com/s/zeyk0q25jotn512/miniccnx.pdf

Tabela 2. Escalabilidade

(a) Topologia *full mesh*

Nº de Nós	Mem. Mini-CCNx (MB)	Mem. *ccnd* (MB)	Mem. Total (MB)	Nº de *Links*	Tempo de inic. (s)
4	15.1	7.2	22.3	6	<1
8	15.3	13.7	29.0	28	3
16	15.7	28.9	44.6	120	11
32	18.2	57.1	75.3	496	48
64	26.0	114.5	140.6	2016	118
128	62.0	229.8	291.8	8128	753

(b) Topologia linear

Nº de Nós	Mem. Mini-CCNx (MB)	Mem. *ccnd* (MB)	Mem. Total (MB)	Nº de *Links*	Tempo de inic. (s)
4	15.0	7.2	22.2	3	<1
16	15.1	28.9	44.0	15	1
64	15.7	113.8	129.5	63	6
256	18.1	458.6	476.7	255	36
512	21.6	918.5	940.1	511	95
1024	27.8	1835.4	1863.2	1023	228
1536	35.3	2754.2	2789.6	1535	320

pelas instâncias do *daemon ccnd* que executam em cada nó, a memória total utilizada (Mini-CCNx + *daemons ccnd*), o número de *links* instanciados e o tempo de iniciação de cada cenário. Pode-se notar que a principal parcela do uso de memória se refere ao *daemon ccnd*, cuja memória cresce linearmente com o número de nós. O tempo de iniciação está fortemente relacionado ao número de *links* instanciados. Por isso, nota-se um tempo maior na topologia *full mesh* quando comparada com a topologia linear.

4.2. Coerência

A coerência ante a variação de parâmetros de experimento (atraso, banda dos *links*, número de *hops*) será analisada. A ferramenta pode ser considerada coerente se, por exemplo, o atraso total aumentar linearmente com o número de *hops*, como o esperado.

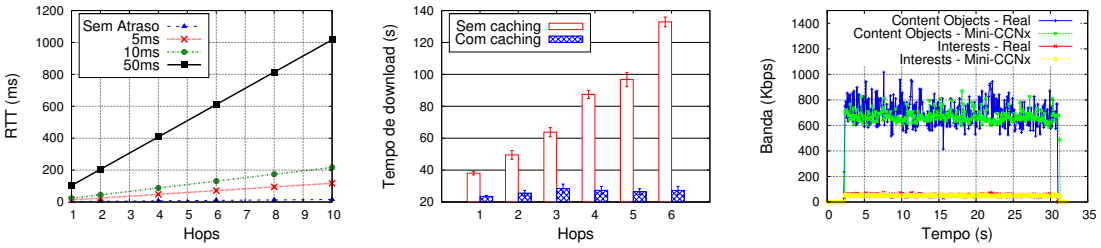
Dois cenários simples utilizando nós CCN foram utilizados. No primeiro (Figura 2(a)), o RTT (*round-trip time*) foi medido para diferentes números de *hops* e diferentes valores de atrasos configurados nos *links* da topologia. No segundo (Figura 2(b)), foi medido o tempo médio de *download* de um arquivo de 100 MB para vários números de *hops*, ou seja, fazendo com que o produtor do conteúdo ficasse cada vez mais distante do cliente. Ainda neste cenário, foram analisados os casos sem *caching* (no qual as requisições pelo arquivo precisam passar por todos os *hops* até chegar ao produtor) e com *caching* (no qual partes do arquivo poderão estar localizadas no *cache* dos nós CCN mais próximos ao requisitante). Ambos os gráficos apresentam valores com intervalo de confiança de 95%.

Na Figura 2(a), pode-se notar o crescimento linear do RTT com o aumento do número de *hops*, como esperado. Além disso, para os diferentes valores de atraso, o comportamento do RTT é consistente. Por exemplo, se todos os *links* têm atraso de 10ms e o número de *hops* é 2, o *ping* deve levar 20 ms na ida, mais 20 ms na volta, mais o tempo de processamento em cada nó, o que é justamente o observado. No cenário da Figura 2(b), nota-se o aumento do tempo de *download* conforme aumenta a distância entre produtor e cliente do conteúdo para o caso sem *caching*, como esperado. Já no caso com *caching*, percebe-se que o tempo médio de *download* é menor do que no caso anterior e pouco relacionado ao número de *hops*. Nesse caso, o tempo de *download* é mais influenciado pela quantidade de conteúdo em *cache* nos nós mais próximos ao requisitante.

4.3. Fidelidade ante Experimentos Reais

A ferramenta possuirá fidelidade se reproduzir o comportamento de experimentos reais. Para a análise de fidelidade, foi criada uma simples topologia com dois *desktops* reais com placas de 100Mbps ligados diretamente entre si, ambos com instalações nativas do código oficial do Projeto CCNx. Em seguida, utilizando o gerador de tráfego

`ccntraffic`[3], o primeiro *desktop* gera constantemente mensagens *Interest* requisitando diferentes conteúdos enquanto o segundo responde com *Content Objects* de 1024 *bytes*. O mesmo cenário foi reproduzido utilizando o Mini-CCNx, com 100Mbps e 200 μs de atraso como parâmetros de *link*. A Figura 2(c) mostra o tráfego de *Interests* e *Content Objects* para ambos os cenários. Pode-se notar que o comportamento da banda utilizando o Mini-CCNx é muito próximo ao comportamento do ambiente real.



(a) Coerência: RTT x nº de hops (b) Coerência: Download (c) Fidelidade: Real x Mini-CCNx

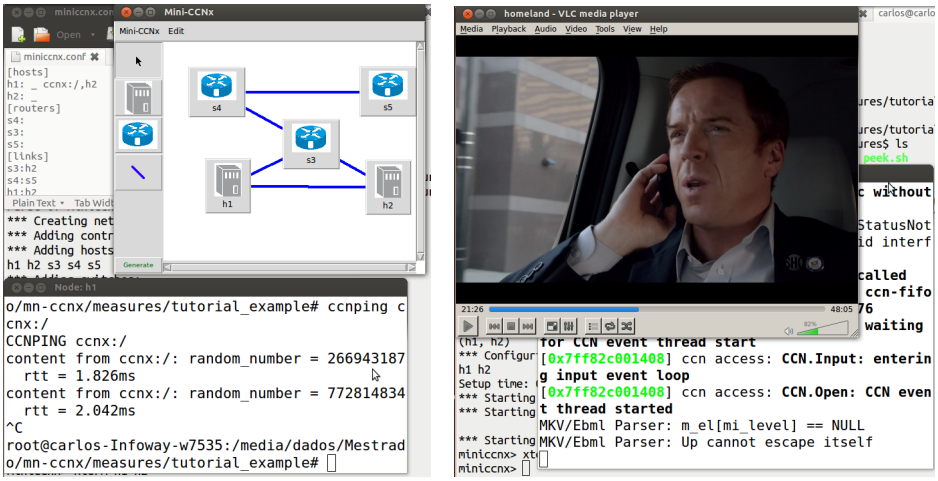
Figura 2. Análise de coerência e fidelidade

5. Documentação, código e demonstração

O Mini-CCNx é um projeto de código livre. Ele está atualmente disponível em *https://github.com/carlosmscabral/mn-ccnx*. Como tal, os autores encorajam o *download*, teste e contribuições para a ferramenta. Documentação, tutorial e exemplos podem ser encontrados em *https://github.com/carlosmscabral/mn-ccnx/wiki*.

A demonstração será feita em duas partes. A primeira (Figura 3(a)) mostrará um passo-a-passo de como utilizar a ferramenta para testar uma aplicação orientada a conteúdo: criação de topologia, definição de parâmetros de *links*, inserção de entradas nas tabelas de encaminhamento e execução de aplicações serão explicados e demonstrados. A segunda parte (Figura 3(b)) mostrará um exemplo prático de como uma aplicação

[3]http://wiki.arl.wustl.edu/onl/index.php/CCNx:_traffic_generation



(a) Fluxo de trabalho (b) Aplicação de vídeo

Figura 3. Demonstrações para o Salão de Ferramentas

de *streaming* de vídeo totalmente orientada a conteúdo se comporta ante a alteração de parâmetros de *link*, como atraso e perda, utilizando as facilidades do Mini-CCNx.

6. Conclusão

O Mini-CCNx, tendo como base a experiência de sucesso das ferramentas de prototipagem rápida para Redes Definidas por Software, apresenta-se como uma ferramenta inovadora, que vem preencher uma lacuna atualmente existente na experimentação de ROCs. A ferramenta apresenta realismo, é flexível e tem baixo custo: toda uma rede orientada a conteúdo com centenas de *hosts* é capaz de ser executada em um simples *laptop* utilizando o Mini-CCNx, com alta configurabilidade e fidelidade dos resultados. Assim, o Mini-CCNx pode ser útil para o desenvolvimento e teste de aplicações orientadas a conteúdo, estratégias de encaminhamento, protocolos de roteamento e técnicas de *caching* dentre outros desafios de pesquisa em ROCs.

Referências

[ccnSim] ccnSim. Scalable chunk-level CCN simulator. http://perso.telecom-paristech.fr/ drossi/index.php?n=Software.ccnSim.

[CCNx] CCNx. Official implementation of the CCN model. https://www.ccnx.org/.

[Dannewitz et al. 2013] Dannewitz, C., Kutscher, D., Ohlman, B., Farrell, S., Ahlgren, B., and Karl, H. (2013). Network of Information (NetInf) - An Information-Centric Networking Architecture. *Computer Communications*.

[de Brito et al. 2012] de Brito, G. M., Velloso, P. B., and Moraes, I. M. (2012). Redes Orientadas a Conteúdo: Um Novo Paradigma para a Internet. *SBRC 2012*.

[Handigol et al. 2012] Handigol, N., Heller, B., Jeyakumar, V., Lantz, B., and McKeown, N. (2012). Reproducible network experiments using container-based emulation. *CoNEXT '12*, page 253.

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

[Lantz et al. 2010] Lantz, B., Heller, B., and McKeown, N. (2010). A network in a laptop. In *Hotnets '10*, pages 1–6, New York, New York, USA. ACM Press.

[NDN Testbed] NDN Testbed. NDN Routing Topology. http://netlab.cs.memphis.edu/script/htm/topology.html.

[ndnSIM] ndnSIM. NS-3 based NDN simulator. http://ndnsim.net/.

[Plagemann et al. 2006] Plagemann, T., Goebel, V., Mauthe, A., Mathy, L., Turletti, T., and Urvoy-Keller, G. (2006). From content distribution networks to content networks - issues and challenges. *Computer Communications*, 29(5):551–562.

[PSIRP] PSIRP. Publish-Subscribe Internet Routing Paradigm. http://www.psirp.org/.

[Soltesz et al. 2007] Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. (2007). Container-based operating system virtualization. *ACM SIGOPS Operating Systems Review*, 41(3):275.