UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE REDES DE COMPUTADORES

Adalberto do Amaral Silva

Análise de desempenho em Redes Centradas em Conteúdo: uma análise comparativa de ferramentas de experimentação

QUIXADÁ

2018

ADALBERTO DO AMARAL SILVA

ANÁLISE DE DESEMPENHO EM REDES CENTRADAS EM CONTEÚDO: UMA ANÁLISE COMPARATIVA DE FERRAMENTAS DE EXPERIMENTAÇÃO

Monografia apresentada no curso de Redes de Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do título de tecnólogo em Redes de Computadores. Área de concentração: Computação.

Orientador: Prof. Dr. Arthur de Castro Callado

QUIXADÁ - CEARÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S578a Silva, Adalberto do Amaral.
Análise de desempenho em redes centradas em conteúdo : uma análise comparativa de ferramentas de experimentação / Adalberto do Amaral Silva. – 2018.
63 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2018.
Orientação: Prof. Dr. Arthur de Castro Callado.

1. Análise. 2. Rede centrada em conteúdo. 3. Ferramentas. I. Título.

CDD 004.6

ADALBERTO DO AMARAL SILVA

ANÁLISE DE DESEMPENHO EM REDES CENTRADAS EM CONTEÚDO: UMA ANÁLISE COMPARATIVA DE FERRAMENTAS DE EXPERIMENTAÇÃO

Monografia apresentada no curso de Redes de Computadores da Universidade Federal do Ceará, como requisito parcial à obtenção do título de tecnólogo em Redes de Computadores. Área de concentração: Computação.

Aprovada em: ____/____/____

BANCA EXAMINADORA

Prof. Dr. Arthur de Castro Callado (Orientador)
Universidade Federal do Ceará – UFC

Prof. Dr. Jeandro de Mesquita Bezerra
Universidade Federal do Ceará - UFC

Prof. Me. Marcos Dantas Ortiz
Universidade Federal do Ceará - UFC

RESUMO

As redes orientadas a conteúdo representam um paradigma que vem ganhando espaço na pesquisa em internet do futuro. Entre as diferentes possibilidades, destaca-se a abordagem das Redes Centradas em Conteúdo (CCN) que passou a ser adotada como um dos padrões para esse modelo de rede proposto devido a seu estado avançado de evolução. Uma forma de contribuir para a evolução desse conceito é através de sua divulgação e prototipagem da tecnologia abrindo espaço para que surjam novas contribuições vindas de acadêmicos, pesquisadores e demais interessados. Foi levantada nesse trabalho a necessidade da existência de uma referência que pudesse indicar e direcionar a esse público as principais opções de ferramentas de experimentação em redes centradas em conteúdo disponíveis, selecionadas a partir de critérios preestabelecidos. Foram selecionadas então as ferramentas Mini-CCNx, ndnSIM e ccnSim. Neste trabalho, são apresentadas as ferramentas, são listados seus principais recursos e detalhes de experimentação e em seguida as ferramentas são submetidas a comparações e análises qualitativa e quantitativa dos recursos consumidos.

Palavras chave: Análises. Redes Centradas em Conteúdo. Ferramentas.

ABSTRACT

Content Oriented Networks represent a paradigm that has been gaining ground within of the internet of the future research field. Among the different approaches, we highlight the Content Centric Networks (CCN) that have been adopted as standards for this proposed network model due to its advanced state of evolution. One way to contribute to the evolution of this concept is through the disclosure and prototyping of the technology, opening space for new contributions that come from academics, researchers and other interested parties. This work answers the need for a reference that would indicate and direct to this public the main options of experimentation tools in content-centric networks available, selected from pre-established criteria. The Mini-CCNx, ndnSIM and ccnSim tools were then selected. In this work these tools are presented, their main features and experimental details are listed and then they are submitted to comparisons and qualitative and quantitative analyses of the resources consumed.

Keywords: Analyses. Content Centric Networks. Tools.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

1. INTRODUÇÃO

Quando o conceito de Internet ainda estava para se formar, havia somente um aglomerado de poucas redes. As necessidades de comunicações existentes consistiam basicamente no compartilhamento de recursos (JACOBSON *et al. 2012*). Segundo Brito (2014), operações como troca de dados textuais, acesso remoto e acesso a recursos compartilhados, tais como impressoras, eram tarefas comumente realizadas pelas estações interconectadas.

Com o crescente avanço tecnológico, tornou-se possível a existência de um *hardware* mais potente a um baixo custo (JACOBSON *et al. 2012*) e consequentemente a existência de *softwares* cada vez mais complexos. Com isso foram surgindo diversas aplicações interativas e interconectadas por rede, gerando diversos tipos de dados, serviços e tráfegos, como arquivos de áudio, vídeo e páginas web, transformando a forma de comunicação existente até chegar ao estado de utilização vivenciado na última década. Portanto, a distribuição de conteúdo na Internet sofreu um processo de evolução, desde um sistema de informação textual para um sistema de informação multimídia, no qual dados, serviços e aplicações são consumidos como conteúdos (PLAGEMANN, 2005 apud BRITO; VELLOSO; MORAES, 2012, p. 212).

Para suportar o novo tipo de tráfego incrementado na rede ao longo do tempo, novas tecnologias adaptativas foram anexadas à arquitetura de rede existente de forma que fosse possível prover suporte a esse novo tipo de tráfego. Analisando brevemente a história da Internet, percebe-se que sua arquitetura atual é a mesma que foi projetada nos primórdios de seu surgimento, tendo sofrido apenas algumas adaptações para que fosse possível atender às novas exigências. Características como ausência de qualidade de serviços, arquitetura centrada em sistemas finais e segurança voltada para o meio e estações são intrínsecas a este tipo de rede regido pelo TCP/IP (BRITO; VELLOSO; MORAES, 2012).

A crescente diversidade de aplicações e protocolos revolucionou a forma como as pessoas passaram a fazer uso da Internet, tornando-as cada vez mais interessadas nos conteúdos disponibilizados na rede, sem nenhum interesse nas informações de localização do conteúdo. Apesar da mudança no uso das aplicações, os protocolos mais utilizados para se obter conteúdo na Internet são, ainda, orientados à localização (BRITO; VELLOSO; MORAES, 2012). Nesse contexto surgem as Redes Orientadas a Conteúdo como uma possível solução para buscar atender a essas exigências, colocando-se como um dos

paradigmas mais bem aceitos entre as áreas de pesquisa de internet do futuro.

Várias abordagens de redes orientadas a conteúdo são propostas, sendo que uma abordagem em particular se destaca como um padrão da tecnologia, as Redes Centradas em Conteúdo (CCN) (BRITO, *2014*). Por se tratar de uma tecnologia ainda emergente e com certo grau de relevância, uma vez que promete transformar a arquitetura atual da internet é necessário que haja ampla divulgação e estímulo ao seu desenvolvimento. Nesse contexto, as ferramentas de experimentação assumem um papel importante auxiliando na compreensão dos conceitos e deixando espaços para a elaboração de soluções voltadas para a área. Ocorre que, entre os trabalhos existentes com foco em prototipagem baseado em ferramentas, seja de análises ou implementação como é possível constatar em Brito (2014) e Cabral (2013), os autores se concentram em uma única ferramenta o que acaba não sendo suficiente para chegar a uma conclusão sobre qual ferramenta é melhor para determinados tipos de experimentos.

Assim, torna-se necessário uma referência comparativa que avalie essas ferramentas de experimentação a fim de determinar suas principais características e capacidades, colocando-as frente-a-frente para extrair o que cada uma tem a oferecer e o que apresentam de melhor em relação às suas concorrentes. Na busca por tentar solucionar essa problemática, este trabalho tem como objetivo propor uma avaliação das principais ferramentas de experimentação em redes centradas em conteúdo através de uma abordagem comparativa, explorando suas principais características e capacidades, inter-relacionando e distinguindo os elementos presentes ou ausentes em cada ferramenta.

Com a execução do trabalho proposto, espera-se que seja possível caracterizar e quantificar as capacidades das ferramentas analisadas em função do que elas proporcionam em termos de recursos. Com essa contribuição, espera-se também que se possa definir a melhor ferramenta a ser utilizada em determinado tipo de trabalho, de acordo com o que se deseja considerar e avaliar. Assim as comunidades acadêmicas, pesquisadores e outros interessados na tecnologia poderão utilizar este trabalho como referência na escolha da ferramenta mais adequada para determinado cenário de redes centradas em conteúdo que desejem estudar.

2. TRABALHOS RELACIONADOS

Diversos trabalhos realizam uma abordagem avaliativa de redes centradas em conteúdo. Dentre os principais temas investigados, pode-se destacar a (i) análise de desempenho em redes centradas em conteúdo e a (ii) apresentação de ferramentas de experimentação de redes centradas em conteúdo.

Brito *(2014)* realiza uma análise de desempenho em redes sem fio orientadas a conteúdo, avaliando o impacto de diversos aspectos de mobilidade no desempenho destas redes. A ideia principal do trabalho consiste em avaliar através de simulações o funcionamento das redes orientadas a conteúdo sem fio, com diferentes níveis de mobilidade, variando a densidade de nós, utilizando diversas métricas como eficiência de entrega, atraso, número de tentativas, carga da rede e número de colisões. Dessa forma, o trabalho busca não somente identificar o comportamento da rede mas também identificar possíveis problemas. Para isso, os autores utilizam como ambiente de simulação o simulador *Network Simulator 3 – (NS-3)*, versão 3.16.

Chiocchetti, Rossi e Rossini *(2013)* apresentam a ferramenta de simulação ccnSim destacando sua alta escalabilidade. A ferramenta é descrita como um pacote em C++ construído no topo do framework OMNET++. Os autores descrevem inicialmente a arquitetura da ferramenta abordando elementos como o modelo de popularidade, mensagens e *chunks*, a arquitetura dos nós e as estatísticas de simulação. Em seguida, descrevem o modelo de referência do simulador, abordando os modelos do cenário, o perfil do cnnSim, a paralelização no ccnSim e o seu desempenho geral.

Cabral *(2013)* apresenta uma ferramenta desenvolvida por ele, o Mini-CCNx. O objetivo do trabalho é o desenvolvimento de uma ferramenta experimental especificamente focada nas Redes Orientadas a Conteúdo (ROCs). Para atender esse objetivo, ele define 5 requisitos que a ferramenta proposta deverá atender, que são: flexibilidade, escalabilidade, baixo-custo, realismo e facilidade de uso. Ao longo do trabalho ele apresenta a ferramenta desenvolvida em função dos requisitos estabelecidos.

Os trabalhos abordados acima representam um padrão dos tipos e modelos de trabalhos existentes de experimentação em redes centradas em conteúdo. Grande parte dos trabalhos de experimentação existentes se concentram exclusivamente na apresentação de ferramentas de experimentação ou mesmo em análises de desempenho diversas da tecnologia de redes centradas em conteúdo utilizando uma única ferramenta e submetendo o experimento às capacidades, recursos e limitações da ferramenta utilizada. Este trabalho assume uma

abordagem diferente dos trabalhos de experimentação de redes centradas em conteúdo existentes, voltando-se para a comparação entre as ferramentas de experimentação colocando-as frente-a-frente e detalhando os recursos e capacidades apresentadas por cada uma delas.

3. OBJETIVOS

Este capítulo detalha os objetivos do trabalho.

3.1 Objetivo geral

O objetivo deste trabalho é avaliar as principais ferramentas de experimentação em redes centradas em conteúdo explorando suas características e capacidades, inter-relacionando e distinguindo os elementos presentes ou ausentes em cada ferramenta.

3.2 Objetivos específicos

- Identificar as ferramentas existentes;
- Definir critérios de seleção das ferramentas para experimentação;
- Selecionar as ferramentas com base nos critérios de seleção estabelecidos;
- Comparar e avaliar os resultados para formar uma conclusão sobre as experiências de experimentação em cada ferramenta.

4. FUNDAMENTAÇÃO TEÓRICA

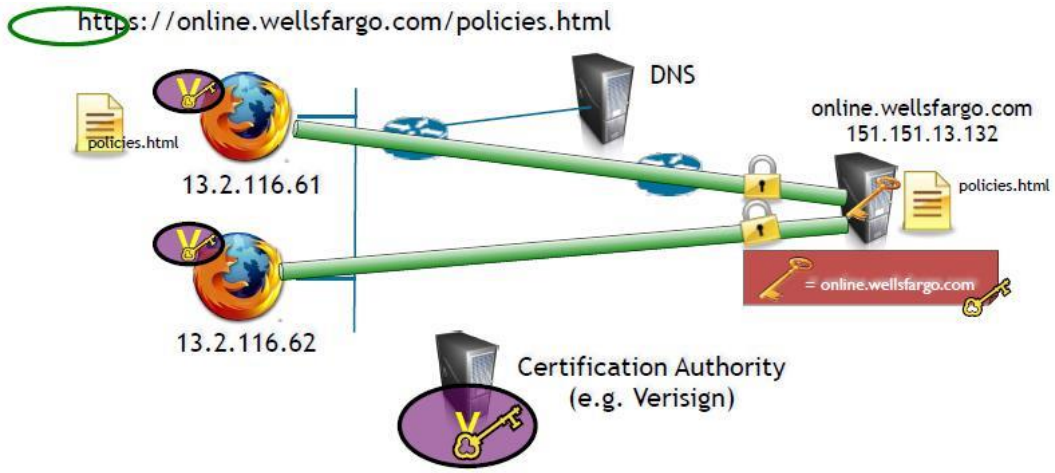
Alguns conceitos são de extrema relevância para a compreensão deste trabalho. Dentre os principais conceitos destacam-se as Redes Orientadas a Conteúdo (ROCs), as Redes Centradas em Conteúdo (*Content Centric Network* – CCN) e análise de desempenho. Nas seções a seguir esses conceitos serão abordados na ordem citada aqui e brevemente descritos.

4.1 Redes Orientadas a Conteúdo

As redes orientadas a conteúdo (ROCs) são um dos principais e mais bem aceitos paradigmas de Internet do Futuro. Nas ROCs o conteúdo é tido como o elemento fundamental da rede, de forma que praticamente todas as operações em rede a serem realizadas, como identificação, roteamento e armazenamento são voltadas para o conteúdo (BRITO; VELLOSO; MORAES, 2012).

Considerando que a arquitetura subjacente da rede atual é essencialmente centrada em sistemas finais, é necessário identificar a estação ou o sistema final detentor do conteúdo que se deseja obter para somente então obter o conteúdo (JACOBSON *et al*., 2012). Com isso, tecnologias adaptativas como o DNS e o HTTP assumem um papel bastante relevante na distribuição dos conteúdos na rede. Por outro lado, a necessidade dessas tecnologias atuarem como intermediárias para a obtenção dos conteúdos gera uma carga extra de dados na rede, causando um certo atraso para a obtenção do conteúdo. Além disso, esse modelo de rede apresenta graves problemas de segurança, uma vez que a segurança é implementada sobre o canal de comunicação ao invés de aplicá-la explicitamente sobre o conteúdo (BRITO; VELLOSO e MORAES, 2012; BRITO, 2014; JACOBSON, 2014) conforme ilustra a Figura 1.

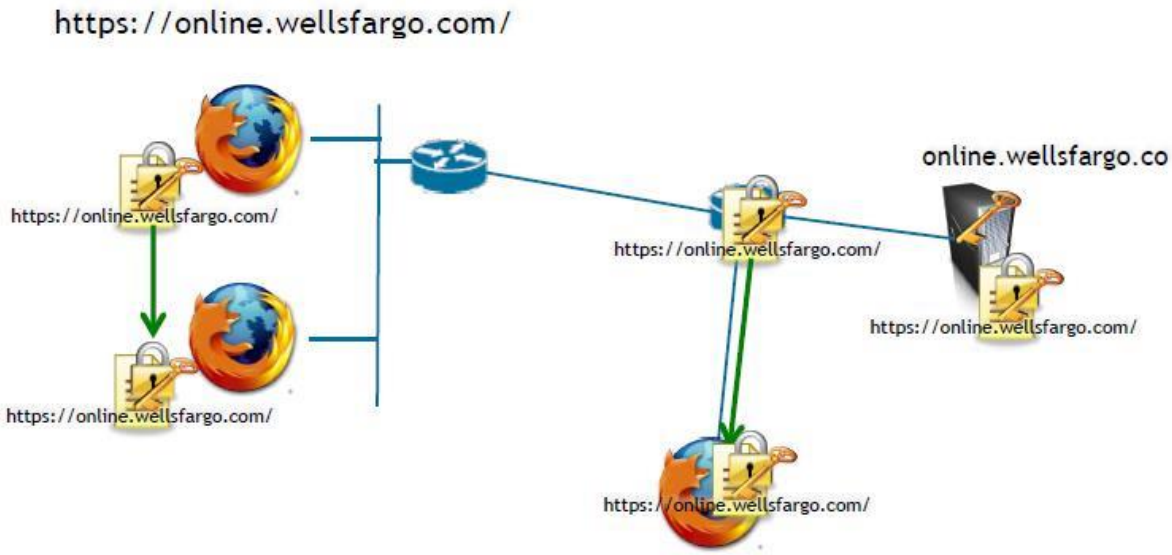
Figura 1 - Problemas existentes no modelo de rede centrado em sistemas finais.



Fonte: Jacinto (2014).

Os paradigmas de redes orientadas a conteúdo surgem nesse contexto como um modelo de rede que busca ser mais eficiente na distribuição de conteúdo e que possa atender às demandas e exigências existentes atualmente. Como pode ser verificado na Figura 2, nas redes orientadas a conteúdo o fluxo de mensagens é dirigido aos nós que manifestam o seu interesse através de identificadores de nomes da informação, em vez de nomes de interfaces de hosts. A segurança é aplicada diretamente no conteúdo, garantindo sua autenticidade, integridade e confiabilidade (JACINTO, 2014).

Figura 2 - Modelo de comunicação e segurança em uma rede orientada a conteúdo.



Fonte - Jacinto (2014).

As ROCs possuem várias abordagens de modelos de distribuição de conteúdo, sendo que as redes centradas em conteúdo se destacam como uma abordagem de grande relevância dentre as demais abordagens e que se apresentam em um estágio evolutivo crescente, colocando-se, portanto, como um dos padrões para a tecnologia (BRITO, *2014*).

4.2 Redes Centradas em Conteúdo

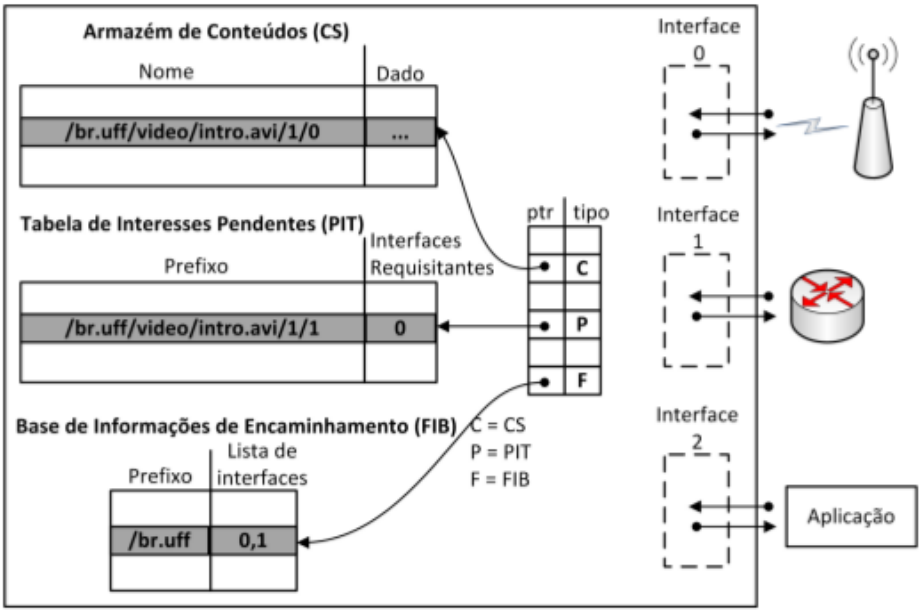
Tendo nascido como um projeto de pesquisa no Centro de Pesquisas de Palo Alto (PARC), liderada por Van Jacobson, tal modelo de rede proposto enfatiza o conteúdo como elemento fundamental da rede. Isso significa que todas as operações realizadas nessa arquitetura de rede como a segurança, o roteamento e a própria distribuição são realizadas exclusivamente com foco no conteúdo, dispensando detalhes da localização do mesmo como é feito na arquitetura atual. Essa abordagem apresenta características básicas como nomeação dos conteúdos e armazenamento nos elementos da rede (MOSKO, 2014).

Mosko (2014) define basicamente dois tipos de mensagens: os interesses e os objetos de conteúdo. Os interesses são mensagens lançadas na rede demonstrando o interesse em receber determinado conteúdo. Objetos de conteúdo são os conteúdos publicados na rede. Objetos de conteúdo são devolvidos em resposta a interesses. Um objeto de conteúdo satisfaz um interesse quando o nome do interesse corresponde ao nome do objeto de conteúdo. Nessa relação de correspondência podem ser considerados outros campos existentes nos pacotes.

Mosko (2014) apresenta três componentes principais para uma CCN que são: Tabela de interesses pendentes (*Pendent Interest Table* - PIT), base de encaminhamento de informações (*Forward Information Base* - FIB) e o cache de conteúdos (*Content Store* - CS). A PIT é uma tabela que mantém entradas para interesses pendentes, isto é, interesses que ainda não obtiveram um objeto de conteúdo correspondente. A FIB é uma tabela responsável por mapear os nomes de caminhos a uma interface apropriada. O CS é o cache de armazenamento dos nós e é responsável por guardar objetos de conteúdo. Quando um nó recebe uma mensagem de interesse ele verifica se o objeto de conteúdo correspondente àquele interesse está presente no seu CS. Caso esteja, ele irá responder à mensagem com um objeto de conteúdo. Caso não esteja ele irá criar uma entrada na PIT associando o interesse à interface de entrada do interesse. Em seguida ele transmite o interesse por todas as interfaces disponíveis exceto a interface pela qual o interesse chegou. Esse processo irá se repetir até que um nó contenha o objeto de conteúdo correspondente e retorne-o pelo caminho inverso, fazendo com que todos os nós que tenham uma entrada na PIT para o objeto de conteúdo

obtenham o conteúdo. Os componentes de uma rede centrada em conteúdo e o funcionamento descrito acima podem ser identificados a partir da figura abaixo.

Figura 3 - Arquitetura e componentes de uma CCN



Fonte: Brito; Velloso; Moraes, 2012.

4.3 Análise de desempenho

Normalmente, quando se deseja compreender o comportamento de um determinado fenômeno, comparar soluções ou tomar alguma decisão realiza-se uma análise de desempenho. Uma análise de desempenho apresenta um conjunto de etapas bem definidas que podem ser desde a definição dos objetivos até a avaliação e apresentação dos resultados. Elementos comumente associados à análise de desempenho são as métricas, as técnicas e as ferramentas utilizadas. Segundo JAIN (1991) as métricas são critérios usados para avaliar o desempenho do sistema, tais como: tempo de resposta, *throughput* e utilização dos recursos. Em relação às técnicas, JAIN (1991) define três técnicas para avaliação de desempenho que são modelagem analítica, simulação e medição. Segundo ele, medições são possíveis somente se algo similar ao sistema proposto já existe, como na criação de uma melhoria para um produto. Já a modelagem analítica e a simulação são para casos onde a medição não é possível. Como ferramentas, ele destaca habilidades de medição, linguagens de simulação e instrumentos de medição.

5. TÉCNICAS DE EXPERIMENTAÇÃO

Motivada pelo constante desenvolvimento tecnológico, o baixo custo de acesso dos usuários e consequentemente a crescente demanda, a Internet cresceu de forma assustadora ao longo das últimas décadas. Atualmente suas formas de uso se estendem desde a realização de operações comuns como a simples acesso a uma página web até a realização de operações mais complexas como transações bancárias (BRITO; VELLOSO; MORAES, 2012). Porém o surgimento de novas aplicações e serviços necessitam da criação de novos protocolos de forma que a evolução e extensão da arquitetura da Internet atual depende fortemente da experimentação em larga escala (MASTORAKIS *et al.*, 2016). Ocorre que a realização de testes de novas soluções nessa arquitetura se mostra algo inviável uma vez que qualquer desvio de funcionamento que leve a sua inoperância pode impactar de forma bastante negativa, podendo causar prejuízos gigantescos na economia mundial ou até mesmo no fornecimento de serviços essenciais como controle de tráfego aéreo ou até mesmo em sistemas médicos interconectados a rede. Nesse contexto surgem ambientes de experimentação que permitem reproduzir com um certo nível de fidelidade o comportamento de cenários de redes diversos se mostrando como alternativas altamente eficazes para a elaboração de novas propostas para a Internet atual. Entre os principais mecanismos para prototipagem de redes de computadores estão os simuladores, emuladores e testbeds.

Simuladores atuam como aplicações que tentam imitar o comportamento de determinada tecnologia gerando uma sequência de eventos padronizados de acordo com a programação ao longo de um intervalo de tempo. Uma vantagem de se utilizar esse tipo de técnica é que ela permite a realização de experimentos com um número significativamente alto de nós. Por outro lado, o realismo acaba não sendo um ponto forte desse tipo de técnica de experimentação. Já o termo emulador pode ser definido como um ambiente sintético que abrange um conjunto de elementos do sistema que atuam de forma sincronizada para representar uma determinada arquitetura. Um ponto forte dessa técnica de prototipagem é que ela possibilita a experimentação com a pilha de protocolos e tráfego real de uma rede, o que certamente levará a resultados mais consistentes e condizentes com o modelo real da tecnologia avaliada. Já os testbeds são plataformas de testes formados pelo agrupamento de equipamentos e dispositivos físicos para a construção de uma infraestrutura controlada voltada para prototipagem. Uma característica marcante dessa técnica é o realismo, uma vez que assim como os emuladores, possibilitam a realização de experimentos com pilha de protocolos e tráfegos reais. Uma de suas principais desvantagens é o alto custo para criação e,

manutenção e gerenciamento do ambiente (CABRAL, 2013)

6. DEFINIÇÃO DOS CRITÉRIOS DE SELEÇÃO

Esta seção apresenta os critérios que permitiram selecionar as ferramentas que analisadas neste trabalho. Por existirem diversos métodos e técnicas de experimentação, nesta seção serão definidos alguns critérios de seleção até mesmo como forma de manter algum alinhamento com o objetivo estabelecido que busca essencialmente manter o foco em ferramentas de estudo voltadas para a tecnologia de redes centradas em conteúdo. Dessa forma, os critérios são listados abaixo.

- Ambiente de experimentação local: buscando atender o objetivo, foi definido que as ferramentas devem se portar como um ambiente de execução local. Portanto, testbeds passam a ser excluídos automaticamente com a especificação desse critério. Acontece que testbeds se comportam um pouco diferentes de ferramentas de estudo, atuando como plataformas de testes, até mesmo pela sua própria definição. Esse comportamento os leva a serem extremamente burocráticos deixando o experimentador muitas vezes refém da infraestrutura fornecida por terceiros e preso a uma série de restrições, fazendo com que em um primeiro momento o usuário se preocupe mais com detalhes da plataforma do que na tecnologia a ser experimentada. Com isso, esse trabalho se restringe a análise de simuladores e emuladores.

- Estabilidade: no momento da execução deste trabalho, a ferramenta deve apresentar pelo menos uma versão estável de forma que seja possível realizar os experimentos nas condições estipuladas sem interrupções no funcionamento.

- Documentação ativa: para que seja possível compreender seu funcionamento e seus detalhes de implementação e suas principais características se faz necessário a existência de uma documentação que possibilite extrair essas informações. Uma análise simplesmente intuitiva da ferramenta, embora pudesse ser eficiente, também poderia deixar passar detalhes importantes de uso da ferramenta.

- Flexibilidade de execução: neste trabalho serão avaliadas somente ferramentas que executem em diferentes arquiteturas de hardware independente do modelo do processador e em diferentes versões de um mesmo sistema operacional.

7. FERRAMENTAS SELECIONADAS

Nesse tópico são listadas as ferramentas selecionadas a partir dos critérios preestabelecidos e definidos no tópico anterior. Este tópico é de grande relevância uma vez que os conceitos apresentados aqui servirão de base para a compreensão de termos abordados posteriormente.

7.1 ccnSim

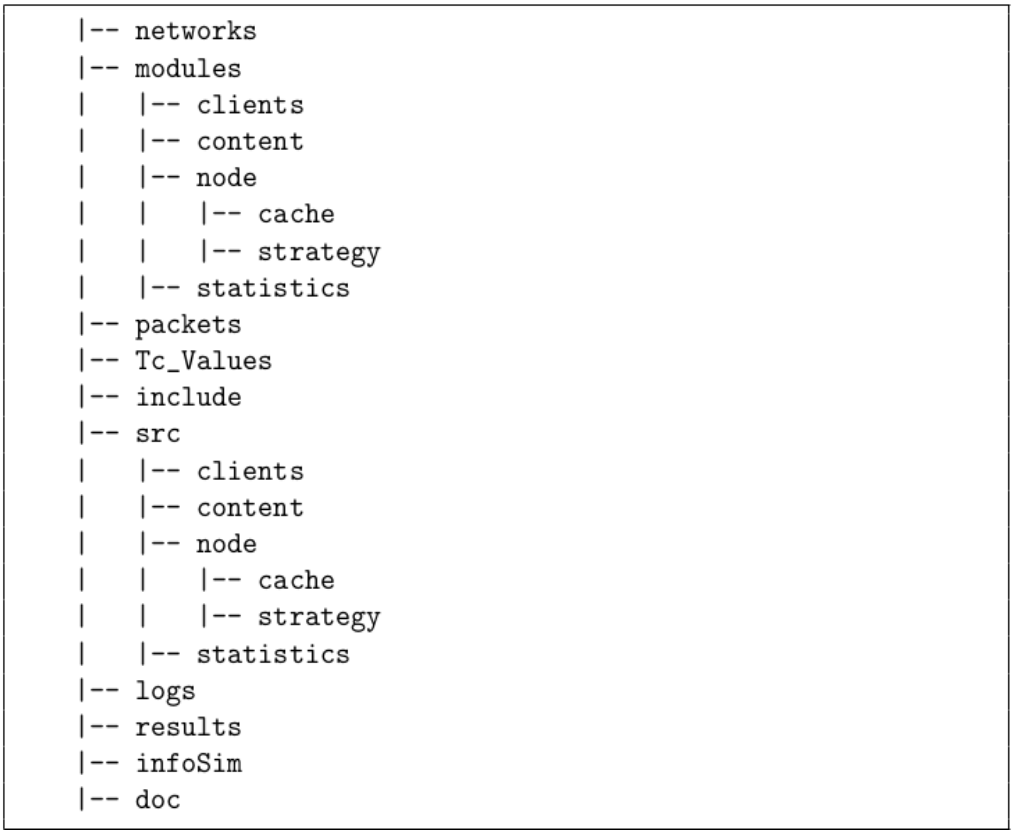
O ccnSim é um simulador escalável em nível de *chunks* para Redes Centradas em Conteúdo, cujo desenvolvimento começou pelo projeto Connect ANR. Ele é escrito em C++ e foi desenvolvimento sobre a estrutura do Omnet++, uma plataforma baseada em eventos usada em simulações de rede. O Omnet++ é caracterizado por: i) um conjunto de classes principais em C++, que podem ser estendidas para customizar o ambiente simulado; ii) uma simples linguagem de descrição de rede (**ned**) usada para descrever as interações entre os módulos; iii) uma linguagem **msg** que define as mensagens trocadas entre os nós da rede.

Ele fornece todas as APIs usadas para simular recursos chave de redes CCN como estratégias de encaminhamento, armazenamento em cache e políticas de decisão de cache. O ccnSim permite realizar simulações clássicas orientadas a eventos (ED) de redes CCN de larga escala (10^9 conteúdos) com uma quantidade moderada de CPU e memória. Ele apresenta também o modelo de simulação ModelGraft, uma nova técnica de *downscaling* baseada em caches TTL, que reduz o consumo de CPU e memória (mais de 100 × simulações de ED), permitindo a simulação de redes CCN em escala Web (10^{12} conteúdos). Outra técnica de simulação é a de modelos analíticos que consiste em uma rede de caches LRU hierárquicos – modelo em que os pontos assumem um posicionamento hierárquico com a aplicação de algoritmos de cache LRU - que pode ser analisada através de um modelo analítico generalizado.

A última versão desse simulador é nomeada de ccnSim-Parallel e traz uma série de melhorias e abordagens de simulação. No entanto, neste trabalho será utilizada a versão anterior, o ccnSim-v0.4 por diversos motivos, detalhados a seguir. O primeiro deles é que existem mais informações e uma documentação mais detalhada sobre o uso da ferramenta para essa versão. O segundo motivo é que o ccnSim-Parallel utiliza como base o ccnSim-v0.4, de forma que a abordagem dessa versão auxiliará na compreensão do ccnSim-Parallel e até

mesmo de versões anteriores. E por fim, o método de simulação adotado neste trabalho será o modelo padrão orientado a eventos (ED) presente nas duas versões, e por isso a utilização de qualquer uma das versões nos experimentos a serem realizados não influenciará nos resultados uma vez que as atualizações incrementadas pelo ccnSim-Parallel não se aplicam à esta abordagem de simulação. A estrutura do simulador é representada na Figura 4.

Figura 4 - Estrutura de diretórios do ccnSim



Fonte: Tortelli *et al.*(2017)

7.2 ndnSIM

O ndnSIM é um simulador de redes baseado no NS-3, voltado para a realização de experimentos com redes orientadas a conteúdo. Tendo sua primeira versão lançada em junho de 2012, o simulador passou por uma série de atualizações ao longo de suas versões posteriores com o acréscimo de novas funcionalidades, recursos e alterações/atualizações pontuais relacionadas a detalhes de implementação da arquitetura *Named Data Networking*

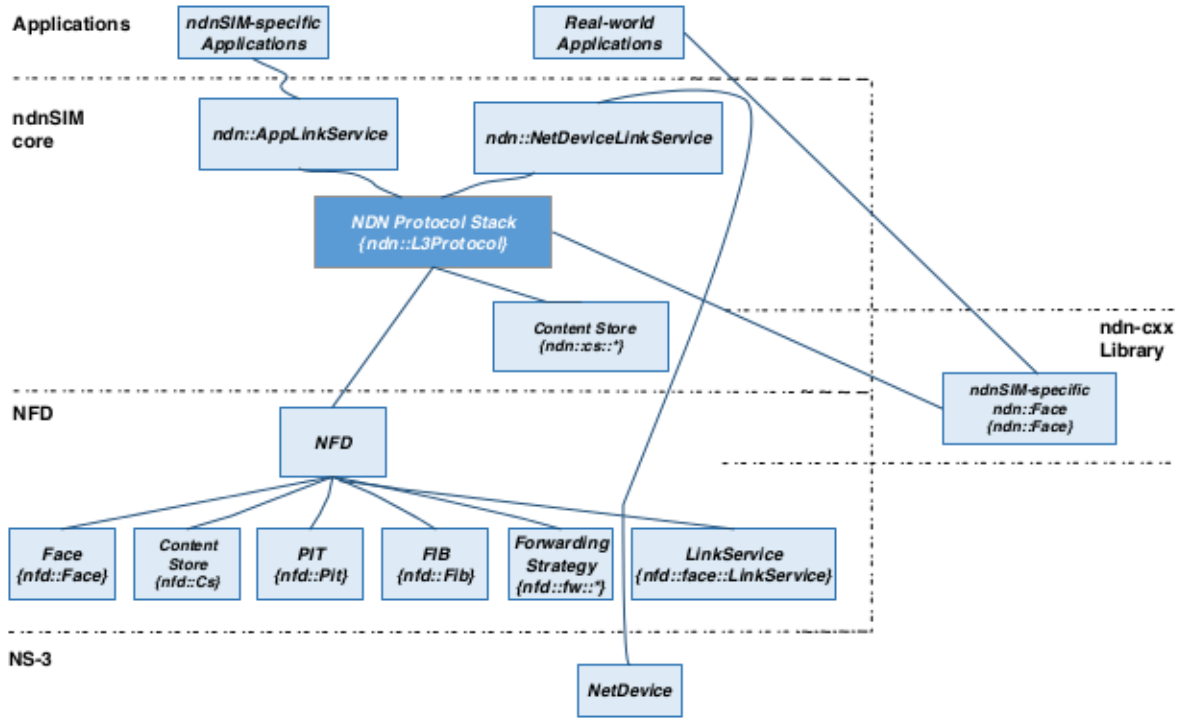
(NDN)[1]. Essa ferramenta de simulação se mostra bastante eficaz na reprodução de experimentos de redes orientadas a conteúdo, sendo utilizada em diversos trabalhos tanto de redes orientadas a conteúdo como em trabalhos de abordagens derivadas (MASTORAKIS et al., 2016).

Embora tenha passado por algumas modificações arquiteturais ao longo de suas versões, tendo sofrido alterações acentuadas principalmente na atualização da versão 1.0 para 2.0, o componente núcleo do ndnSIM continua o mesmo, desempenhando as mesmas funções dentro da arquitetura. Esse componente, implementado pela classe *ndn::L3Protocol* é responsável pela instalação da pilha do protocolo NDN em cada um dos nós simulados. A grande novidade na arquitetura tem sido a integração com o *Named Data Networking Forwarding Daemon* (NFD), a partir da versão 2.0 do ndnSIM, permitindo que novos recursos associados a esse encaminhador de rede sejam incrementados/atualizados a cada nova versão (MASTORAKIS et al., 2016).

O *Named Data Networking Forwarding Daemon* (NFD) é um encaminhador de rede que implementa e evolui junto com o protocolo Named Data Networking (NDN). O principal objetivo do NFD é dar suporte a diversas experimentações da arquitetura NDN. O projeto NDN enfatiza modularidade e extensibilidade para permitir experimentos fáceis com a apresentação de novos protocolos, algoritmos e aplicações. A principal funcionalidade do NFD é encaminhar pacotes *Interest* e *Data.* Para isso, ele abstrai os mecanismos de transporte de rede a um nível mais baixo nas interfaces NDN, mantendo estruturas de dados básicas como CS (Armazenador de Conteúdos) , PIT (Tabela de Interesses Pendentes) e FIB (Base de Informações de Encaminhamento), implementando o processamento lógico dos pacotes. Além do encaminhamento básico de pacotes, ele também suporta múltiplas estratégias de encaminhamento, e um gerenciamento de interface para configurar, controlar e monitorar o NFD (AFANASYEV et al., 2014). A Figura 5 apresenta todos os componentes presentes na arquitetura do ndnSIM, bem como a hierarquia e o relacionamento existente entre eles.

[1] <https://named-data.net/>.

Figura 5 - Componentes do ndnSIM

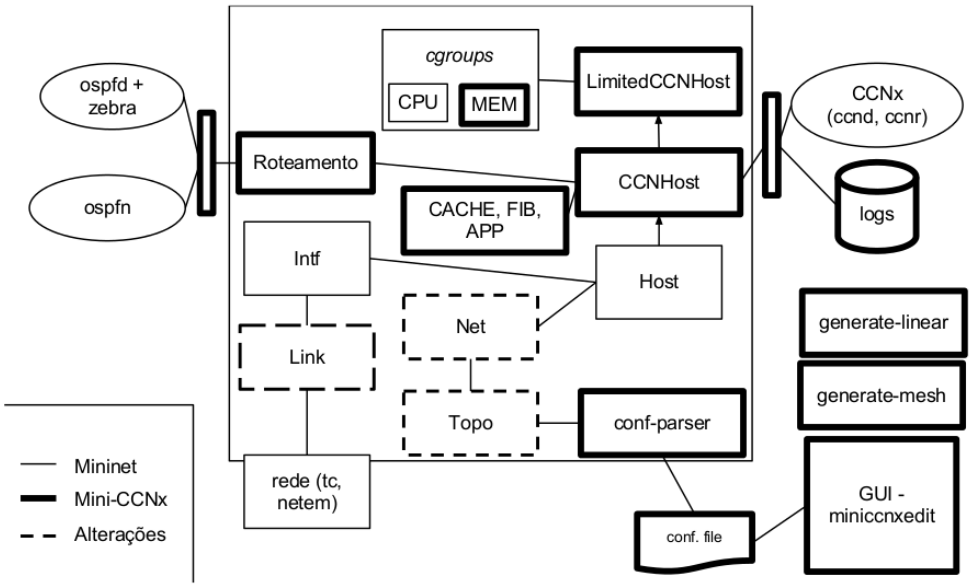


Fonte: ndnsim.net (2017)

7.3 Mini-CCNx

O Mini-CCNx é uma ferramenta de prototipagem de redes centradas em conteúdo que utiliza emulação baseada em contêineres (EBC) para a criação do ambiente de rede emulado. Ele foi implementado a partir do Mininet, um emulador focado em Redes Definidas por Software baseadas em OpenFlow. No entanto, o Mini-CCNx não possui nenhum recurso de emulação para Redes Definidas por Software, como switches e controladores programáveis. O intuito de estender o Mininet foi apenas para aproveitar a infraestrutura já existente de criação dos contêineres (CABRAL, 2013). A Figura 6 mostra a estrutura do Mini-CCNx como uma extensão desse emulador. Os blocos com bordas em negrito representam as partes introduzidas pelo Mini-CCNx, os blocos com bordas pontilhadas representam as partes aproveitadas, mas com alterações significativas e os blocos com linhas simples representam partes que foram aproveitadas sofrendo apenas alterações pontuais.

Figura 6 - Estrutura do Mini-CCNx

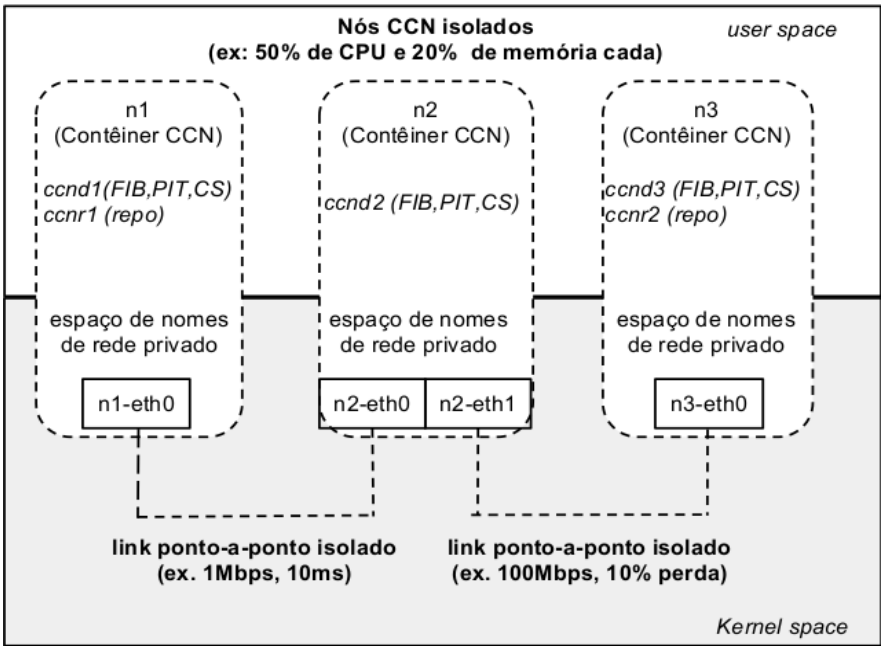


Fonte: CABRAL (2013)

Conforme afirma Cabral (2013, p.21):

> [...] a EBC permite que diferentes grupos de processos, cada qual em seu contêiner Linux, tenham visões independentes (diferentes espaços de nome) de recursos do sistema como IDs de processos, nome de usuários, sistemas de arquivos e interfaces de rede mas, ainda assim, sejam executados no mesmo kernel.

Figura 7 - Contêineres como nós no Mini-CCNx



Fonte: CABRAL (2013)

Isso pode ser visualizado na Figura 7, onde cada contêiner no Mini-CCNx é representado como um nó CCN.

Segundo Cabral (2013) cada nó tem seu próprio espaço de rede privado, com suas próprias interfaces e todas as estruturas associadas como cache ARP e tabela de roteamento. Ainda segundo o autor, cada nó também possui as estruturas características do modelo CCN, como o CS, PIT e FIB implementadas pelo *daemon* ccnd e opcionalmente repositórios de dados implementados pelo *daemon* ccnr. Por ser um emulador, o Mini-CCNx também pode se beneficiar do uso do código base oficial do modelo CCN que implementa esses *daemons*. Os nós, por sua vez, são conectados entre si utilizando links Ethernet virtuais (veth) ponto-a-ponto no espaço de kernel.

Em uma rede real os componentes não interferem no comportamento dos outros presentes na rede. Como os contêineres fazem uso do mesmo kernel e dos mesmos recursos na máquina, existe a possibilidade de que os processos de um contêiner venham a interferir no consumo de outros presentes em outros contêineres. Para garantir esse isolamento foram utilizados os *Control Groups* (cgroup), permitindo a alocação de recursos como tempo de CPU, memória de sistema ou capacidade de rede especificamente para um grupo de processos em execução no sistema (CABRAL, 2013).

8. RECURSOS E DETALHES DE EXPERIMENTAÇÃO

Esse tópico tem por objetivo apresentar os recursos fornecidos pelas ferramentas selecionadas e apresentar alguns detalhes de experimentação. Para facilitar a compreensão será utilizada uma topologia de exemplo que servirá como base para a apresentação de conceitos de grande relevância nos tópicos a seguir. Esse cenário de exemplo é ilustrado na Figura 8.

Figura 8 - Cenário de exemplo

Fonte: Elaborado pelo Autor

8.1 ndnSIM

Como já se tem conhecimento a partir das seções anteriores, o ndnSIM é um simulador baseado no NS-3 e por isso as ações realizadas nele são todas efetuadas pela linha de comando. Recursos gráficos estão disponíveis apenas para a visualização dos experimentos graficamente, tendo ainda que gerar e executar o cenário experimentado pela linha de comando.

Para a realização de um experimento no ndnSIM, é necessário definir o cenário e suas configurações fazendo uso das classes e funções C++ apropriadas. Também é possível criar alguns cenários com a linguagem de programação *python*, embora a quantidade de recursos fornecidos ao se fazer uso dessa linguagem e a própria documentação a respeito sejam mais escassos. O quadro abaixo mostra o código fonte para o cenário de exemplo ilustrado na Figura 8.

Algoritmo 1 - Código fonte de topologia linear no ndnSim

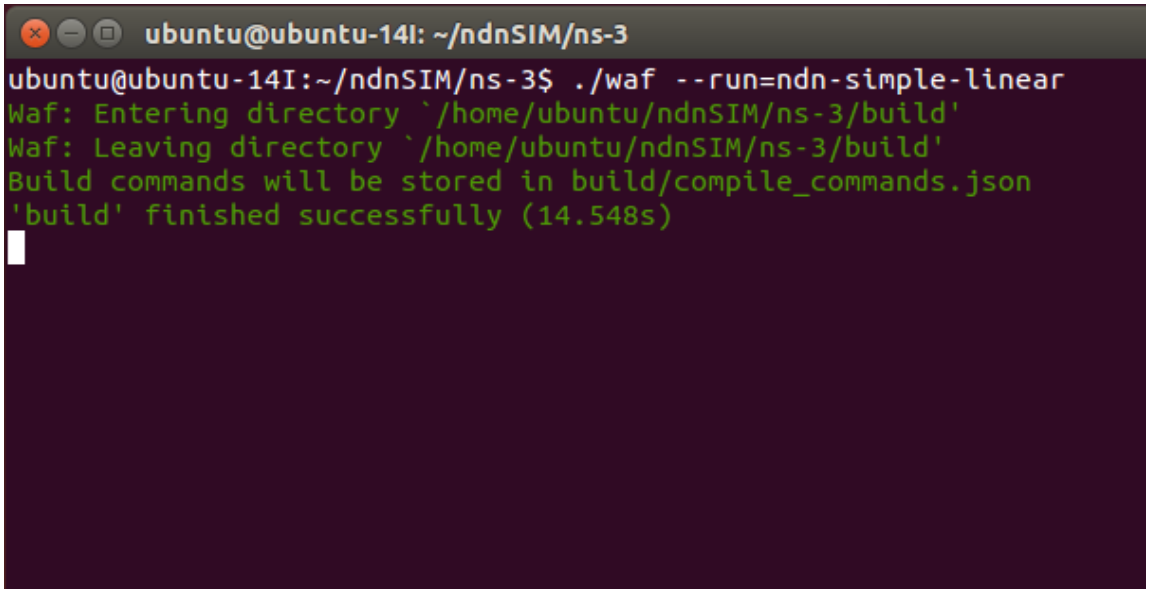
```
#include “ns3/core-module.h”
#include “ns3/network-module.h”
#include “ns3/point-to-point-module.h”
#include “ns3/ndnSIM-module.h”

namespace ns3 {

int main(int argc, char* argv[]) {

  // Configurando parâmetros padrão para enlaces e canais ponto a ponto
  Config::SetDefault(“ns3::PointToPointNetDevice::DataRate”,
StringValue(“1Mbps”));
  Config::SetDefault(“ns3::PointToPointChannel::Delay”, StringValue(“10ms”));
  Config::SetDefault(“ns3::DropTailQueue::MaxPackets”, StringValue(“20”));

  // Ler parâmetros de linha de comando opcionais (por exemplo, ativar o
visualizador com ./waf –run=<> --visualize
  CommandLine cmd;
  cmd.Parse(argc, argv);

  // Criando nós
  NodeContainer nodes;
  nodes.Create(5);

  // Conectando nós usando dois enlaces
  PointToPointHelper p2p;
  p2p.Install(nodes.Get(0), nodes.Get(1));
  p2p.Install(nodes.Get(1), nodes.Get(2));
  p2p.Install(nodes.Get(2), nodes.Get(3));
  p2p.Install(nodes.Get(3), nodes.Get(4));

  // Instalando pilha NDN em todos os nós e configurando o cache

  ndn::StackHelper ndnHelper;

  ndnHelper.setCsSize(50000);
  ndnHelper.setPolicy(“nfd::cs::lru”);

  ndnHelper.SetDefaultRoutes(true);
  ndnHelper.InstallAll();

  // Escolhendo a estratégia de encaminhamento
  ndn::StrategyChoiceHelper::InstallAll(“/prefix”, “/localhost/nfd/strategy/multicast”);

  // Instalando aplicações

  // Consumidor
  ndn::AppHelper consumerHelper(“ns3::ndn::ConsumerCbr”);
```

```
  // Consumidor irá solicitar /prefix/0, /prefix/1, ...
  consumerHelper.SetPrefix("/prefix");
  // 10 interesses em um segundo
  consumerHelper.SetAttribute("Frequency", StringValue("100"));
  consumerHelper.Install(nodes.Get(0));                        // primeiro nó

  // Produtor
  ndn::AppHelper producerHelper("ns3::ndn::Producer");
  // Produtor irá responder todas as solicitações começando por /prefix
  producerHelper.SetPrefix("/prefix");
  producerHelper.SetAttribute("PayloadSize", StringValue("1024"));
  producerHelper.Install(nodes.Get(4));                        // último nó

  Simulator::Stop(Seconds(20.0));
  Simulator::Run();
  Simulator::Destroy();

  return 0;
}

} // namespace ns3

int main(int argc, char* argv[]) {
  return ns3::main(argc, argv);
}
```

Fonte: Elaborado pelo autor

Para executar a simulação, utiliza-se o comando **./waf –run=nome_do_arquivo_fonte**. A Figura 9 mostra a saída desse comando.

Figura 9 - Simulação em Execução no ndnSIM



Fonte: Elaborado pelo Autor

O formato dos pacotes do ndnSIM é o TLV (tipo, tamanho e valor), o mesmo utilizado para o formato dos pacotes do protocolo NDN. Também usa a biblioteca ndn-cxx (*NDN C++ library with experimental extensions*) que implementa primitivas NDN que podem ser usadas para desenvolver várias aplicações NDN, permitindo inclusive a realização de experimentos utilizando aplicações reais. Além disso, como já visto anteriormente, tem acoplado na sua estrutura o NFD, um encaminhador de pacotes que é o elemento central do protocolo NDN. Enfim, todo o código da pilha de protocolo do NFD e seus elementos de desenvolvimento auxiliares foram incorporados à ferramenta, permitindo um alto grau de realismo nas simulações.

A capacidade de configuração (*configurability*) das estruturas características de redes centradas em conteúdo é algo presente no ndnSIM. Para o cache de dados (*Content Store*) é possível configurar parâmetros como seu tamanho e qual política de substituição de cache será utilizada. Na versão 1.0 do ndnSIM, o cache de dados apresentava uma possibilidade maior de configurações e funcionalidades possíveis, por isso ele foi mantido ao longo das versões posteriores e persiste até a versão atual. Com essa versão mais antiga do cache, é possível selecionar entre várias implementações de armazenamento de conteúdo e aplicá-las na simulação. Também é possível estabelecer rotas entre os nós individualmente ou utilizar uma classe para que ela configure automaticamente as rotas entre os nós da rede. Outro recurso, é a possibilidade de definir uma ou mais estratégias de encaminhamento que permitirão que as decisões de encaminhamento sejam tomadas a partir delas. Além disso, é possível que o usuário implemente sua própria estratégia de encaminhamento a partir de classes abstratas.

A Tabela 1 apresenta as estratégias de encaminhamento e as políticas de substituição de cache disponíveis nesta versão do simulador. Para utilizá-las é necessário somente passar como parâmetro para a função que faz a configuração dessas funcionalidades.

Tabela 1 - Estratégias e políticas do ndnSIM

Estratégias de Encaminhamento	**Políticas de Substituição de cache**
/localhost/nfd/strategy/best-route	*nfd::cs::lru*
/localhost/nfd/strategy/ncc	*nfd::cs::priority_fifo*
/localhost/nfd/strategy/multicast	
/localhost/nfd/strategy/client-control	

Fonte: Elaborado pelo Autor

Há uma diversidade de aplicações com diferentes características de geração de tráfego com parâmetros que permitem configurá-las da forma que for mais conveniente ao usuário, proporcionando extrema flexibilidade na experimentação. Além disso é possível escrever aplicações de geração de tráfego personalizadas, acrescentando ou criando as características de tráfego que se deseja obter. A Tabela 2 apresenta as aplicações disponíveis para nesta versão do simulador.

Tabela 2 - Aplicações do ndnSIM

Aplicações	Descrição
ConsumerCbr	Gera um tráfego de *Interests* com um padrão predefinido
ConsumerZipfMandelbrot	Solicita conteúdos seguindo o padrão Zipf-Mandelbrot
ConsumerBatches	Gera um número de *Interests* em pontos específicos da simulação
ConsumerWindow	Gera *Interests* baseado em janela deslizante

Fonte: Elaborada pelo Autor

Em relação à visualização das saídas, existe um conjunto de classes auxiliadoras que permitem a visualização de informações pertinentes às camadas da rede, ao cache de conteúdo e ao tráfego gerado pelas aplicações. Para usar esse recurso, basta adicionar na função específica os parâmetros adequados no código fonte. A Tabela 3 mostra a descrição de algumas dessas classes.

Tabela 3 - Tracers do ndnSIM

Tracer	Descrição
L3RateTracer	Permite visualizar o número de pacotes *Interest/Data* encaminhados em cada nó
L2RateTracer	Permite visualizar pacotes perdidos na camada 2
CsTracer	Permite obter estatísticas do cache

Fonte: Elaborada pelo Autor

8.2 Mini-CCNx

Por utilizar emulação baseada em contêineres, todas as operações de experimentação em redes centradas em conteúdo são possíveis no Mini-CCNx. Para realizar um experimento, existem duas formas possíveis. A primeira é através do arquivo de configuração, que é usado para definir todos os elementos do cenário, como nós, enlaces, parâmetros associados a redes centradas em conteúdo e demais configurações da rede.

Outros elementos que podem ser definidos são a porcentagem de consumo e memória dos componentes da rede, pois como já visto em tópicos anteriores essa configuração é utilizada como forma de isolar os recursos evitando interferências entre eles no consumo dos recursos. Caso não sejam definidos, processos de algum componente da rede poderão solicitar mais recursos do que outros, prejudicando assim o realismo do experimento. Para realizar essas operações é necessário ter conhecimento da sintaxe que representa todos esses elementos, o que é extremamente fácil. O quadro abaixo mostra um exemplo de como o cenário da Figura 8 pode ser representado a partir do arquivo de configuração do Mini-CCNx.

Algoritmo 2 - Definição de topologia linear no Mini-CCNx

```
[preferences]
getMetrics: 1
ipBase: 10.0.0.0/8
metricsTimer: 30
dbPort: 8086
dbHost: localhost
dbPass: root
dbName: miniccnx_data
dbUser: root
terminalType: xterm
startCLI: 0
templatePath: miniccnx.conf
[hosts]
h1: appName='./script1.sh' cache=50000 cpu=0.1 mem=0.5 getMetrics=1 ccnx:/h2,r1 ccnx:/r1,r1 ccnx:/r2,r1 ccnx:/r3,r1
h2: appName='./script2.sh' cache=50000 cpu=0.1 mem=0.5 getMetrics=1 ccnx:/h1,r3 ccnx:/r1,r3 ccnx:/r2,r3 ccnx:/r3,r3
[routers]
r1: cache=50000 cpu=0.1 mem=0.2 getMetrics=1 ccnx:/h1,h1 ccnx:/h2,r2 ccnx:/r2,r2 ccnx:/r3,r2
r2: cache=50000 cpu=0.1 mem=0.2 getMetrics=1 ccnx:/h1,r1 ccnx:/h2,r3 ccnx:/r1,r1 ccnx:/r3,r3
r3: cache=50000 cpu=0.1 mem=0.2 getMetrics=1 ccnx:/h1,r2 ccnx:/h2,h2 ccnx:/r1,r2 ccnx:/r2,r2
[links]
h1:r1 bw=1 delay=10ms
r1:r2 bw=1 delay=10ms
r2:r3 bw=1 delay=10ms
r3:h2 bw=1 delay=10ms
```

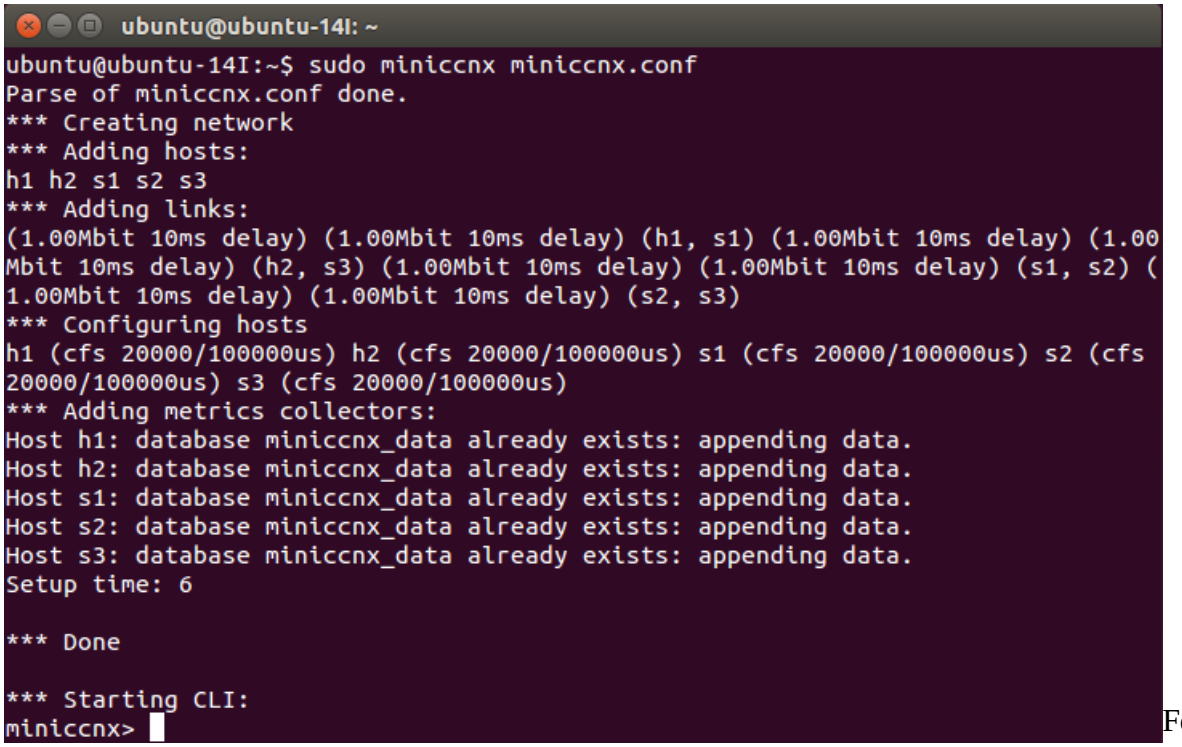
Fonte: Elaborado pelo autor

Em redes centradas em conteúdo, não há distinção entre os nós quanto a sua classificação, sendo que qualquer nó pode atuar como produtor e consumidor ao mesmo tempo. No Mini-CCNx existe um recurso que permite diferenciar os nós entre *hosts* e *routers*,

bastando apenas separá-los em seções diferentes. Nós do tipo *host* têm um parâmetro *appName* que permite especificar uma aplicação ou comando que se deseja executar assim

Figura 10 - Emulação no Mini-CCNx

que ele for criado. Com esse recurso é possível, caso desejável, especificar um *script* de comandos ou aplicações que pode ser utilizado para executar uma série de instruções e configurações no nó após sua criação. Já os nós do tipo *router* se diferenciam em relação aos nós do tipo *host* por não apresentarem esse parâmetro. A Figura 10 mostra a saída que será observada ao ser realizado um experimento no Mini-CCNx com o arquivo de configuração.

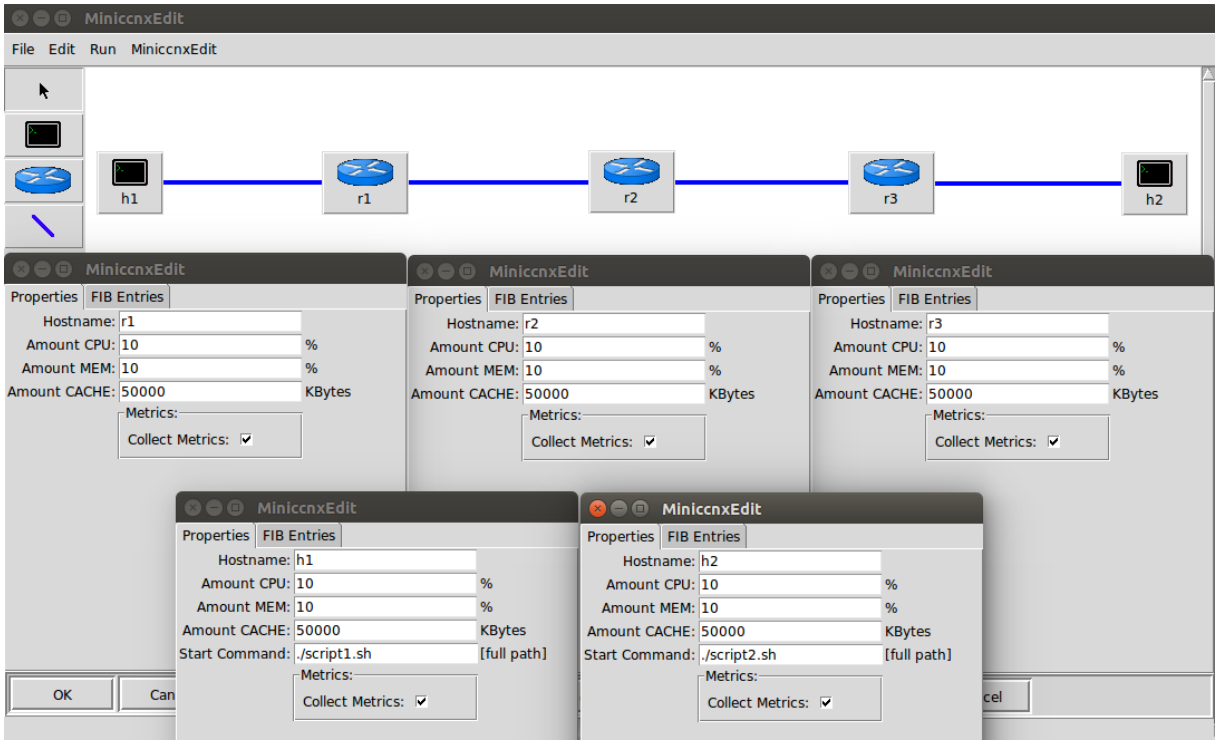


Fonte: Elaborada pelo Autor

A segunda forma é através da ferramenta gráfica *miniccnxedit*, que torna essa tarefa ainda mais simples e intuitiva para o usuário. Com o miniccnxedit é possível criar todo o cenário com o simples ato de arrastar e soltar os componentes. Através de janelas gráficas associadas aos componentes que se deseja configurar, é possível definir os valores dos parâmetros de redes centradas em conteúdo assim como o arquivo de configuração. O cenário desenhado no miniccnxedit com as configurações dos componentes serão transformados em um arquivo de configuração ao se clicar no botão *generate.* Dessa forma, o usuário pode realizar os experimentos sem precisar conhecer a sintaxe do arquivo de configuração,

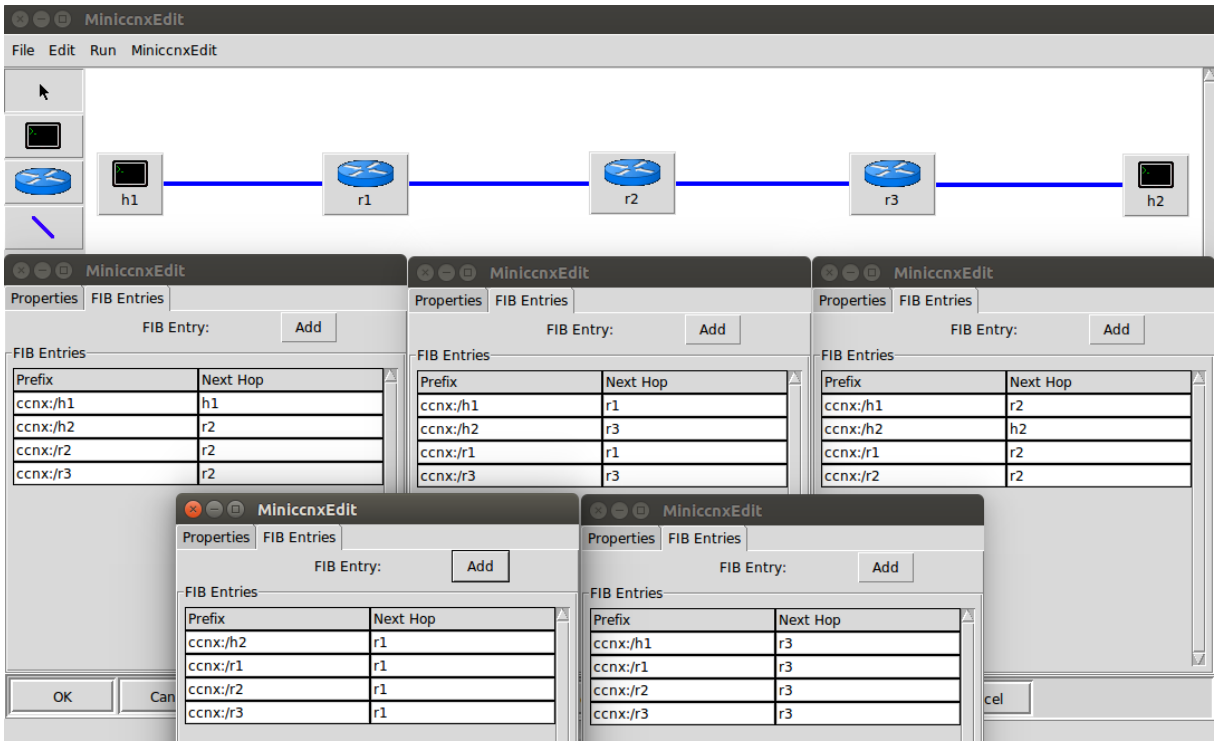
bastando apenas utilizar os recursos gráficos. A Figura 11 e a Figura 12 mostram um experimento com a configuração correspondente à do arquivo de configuração sendo realizada no miniccnxedit. Para executá-lo, basta utilizar o comando ***sudo miniccnxedit***.

Figura 11- Configuração dos recursos no miniccnxedit



Fonte: Elaborado pelo Autor

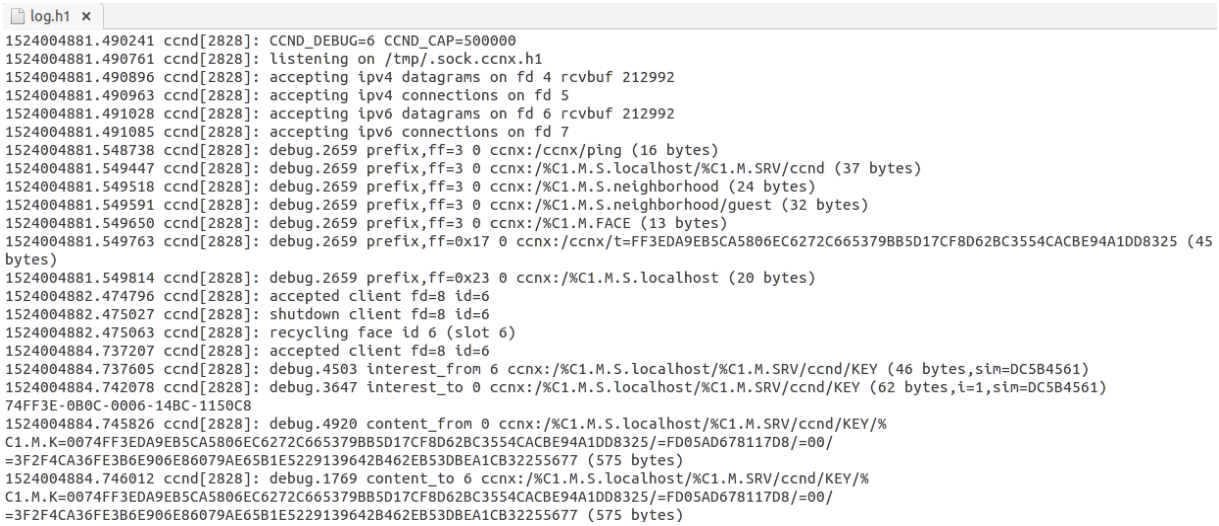
Figura 12 - Configuração das rotas no miniccnxedit



Fonte: Elaborada pelo Autor

Outro recurso interessante fornecido pelo Mini-CCNx é a possibilidade de realizar experimentos com roteamento dinâmico. Como explicado anteriormente, por utilizar emulação baseada em contêineres, protocolos e aplicações experimentados em uma máquina convencional também podem ser usadas no emulador. Dessa forma o protocolo de roteamento dinâmico OSPFN, se configurado, pode ser utilizado para descobrir as rotas baseadas em nomes do cenário emulado. O Mini-CCNx também fornece a possibilidade de visualizar o que ocorre na rede através de *logs* que registram todas as mensagens trocadas entre os nós da rede. Para cada nó será gerado um arquivo de log que possui essas informações. O formato desse arquivo pode ser observado na Figura 13.

Figura 13 - Registro de saída do Mini-CCNx



Fonte: Elaborado pelo Autor

8.3 ccnSim

Como o ccnSim é construído sobre a plataforma do Omnet++, primeiro é necessário instalar o Omnet++ e depois deve ser realizado o *patch* para que sejam feitas as adaptações e atualizações necessárias. Para a última versão do ccnSim, existem dois procedimentos descritos no guia de instalação: uma para as versões do Omnetpp-v4.x e outra para as versões do Omnetpp-v5.x, sendo que eles devem ser aplicados de acordo com a versão do Omnet++ instalado.

Para realizar um experimento no ccnSim é necessário primeiramente definir a topologia a partir de uma linguagem de descrição de rede fornecida pelo Omnet++ nomeada de **ned.** O quadro abaixo apresenta a topologia escrita nessa linguagem para o cenário ilustrado na Figura 8.

Algoritmo 3 - Definição de topologia linear no ccnSim

```
package networks;

network linear_network extends base_network{

   parameters:
      // Número de nós ccn
      n = 5;

      // Número de repositórios e pontos de ligação
      num_repos = 5;
      node_repos = "0,1,2,3,4";
```

```
        replicas = 1;

        // Número de clientes e ponto de ligação
        num_clients = 1;
        node_clients = "0";

connections allowunconnected:

node[0].face++ <--> { delay=10ms; datarate=1Mbps; } <-->node[1].face++;
node[1].face++ <--> { delay=10ms; datarate=1Mbps; } <-->node[2].face++;
node[2].face++ <--> { delay=10ms; datarate=1Mbps; } <-->node[3].face++;
node[3].face++ <--> { delay=10ms; datarate=1Mbps; } <-->node[4].face++;
}
```

Fonte: Adaptado pelo autor

O próximo passo consiste na geração de um arquivo com a extensão **.ini** dentro do diretório do ccnSim especificando os valores para os parâmetros da rede e da simulação do ccnSim. Esses parâmetros são apresentados na Tabela 4 acompanhados de uma breve descrição e valores de exemplos definidos com base no cenário de exemplo adotado neste tópico.

Tabela 4 - Parâmetros de configuração do ccnSim

Nº	Parâmetros	Descrição	Exemplos
1	T	Rede simulada (nome do arquivo ned)	*linear*
2	#Clients	Número de clientes	*1*
3	#Repos	Número de Repositórios	*5*
4	FS	Estratégia de encaminhamento	*nrr1*
5	MC	Política de decisão de cache (ou Meta-caching)	*lce*
6	RS	Estratégia de substituição	*lru*
7	Alpha	Expoente de Zipf	*1*
8	C	Tamanho do cache	*1e5*
9	NC	Nome do tamanho do cache (válido apenas para 2-LRU)	*1e4*
10	M	Cardinalidade do catálogo	*1e7*
11	R	Número total de solicitações simuladas	*1e7*
12	Lambda	Taxa de solicitação agregada para cada cliente	*100*
13	ClientType	Tipo de cliente	*IRM*
14	CDType	Tipo de distribuição de conteúdo	*IRM*
15	Toff	Tempo de desligamento (somente para simulações Shot Noise Model – SNM)	*1*
16	n	Execuções a serem simuladas (# runs = n + 1)	*0*
17	StartType	Tipo de Inicialização (Cold vs Hot start)	*cold*
18	FillType	Como os caches são preenchidos em caso de Hot Start	*naive*
19	Y	Parâmetro n parcial	*0.75*
20	Δ	Fator de downscaling (para simulações TTL)	*1*
21	TcFile	Arquivo de valores Tc para todos os nós	–
22	TcNameFile	Arquivo de valores Tc para NC de todos os nós	–

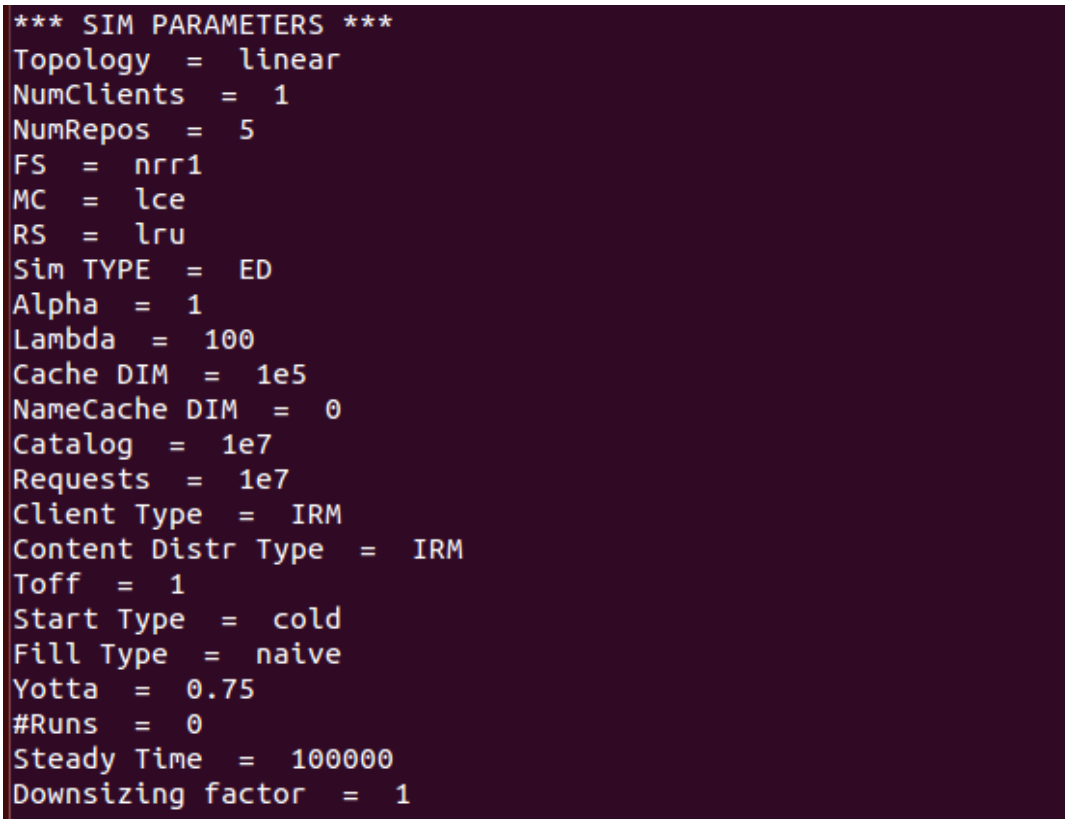
Fonte: Elaborada pelo Autor

Esse arquivo **.ini** pode ser gerado manualmente (o que é bem mais complicado) a partir de um outro arquivo de exemplo ou pode ser gerado por meio de um script apropriado para isso. Para executar uma simulação deve ser utilizado o comando ***./runsim_script_ED_TTL.sh {parameters}.*** O quadro abaixo mostra um exemplo de execução com base na topologia de exemplo definida para esta seção e os valores de exemplo definidos na Tabela 4.

./runsim_script_ED_TTL.sh linear 1 5 nrr1 lce lru 1 1e5 1e4 1e7 1e7 100 IRM IRM 1 0 cold naive 0.75 1

A Figura 14 mostra a saída que será visualizada ao executar o comando acima. Esse método foi acrescentado a partir do ccnsim-v0.4. O *script* ***./runsim_script_ED_TTL.sh*** cria automaticamente um novo arquivo **.ini**, a partir de um padrão, onde os valores ***{parameters}*** transmitidos da linha de comando estão associados a variáveis correspondentes no arquivo **.ini**.

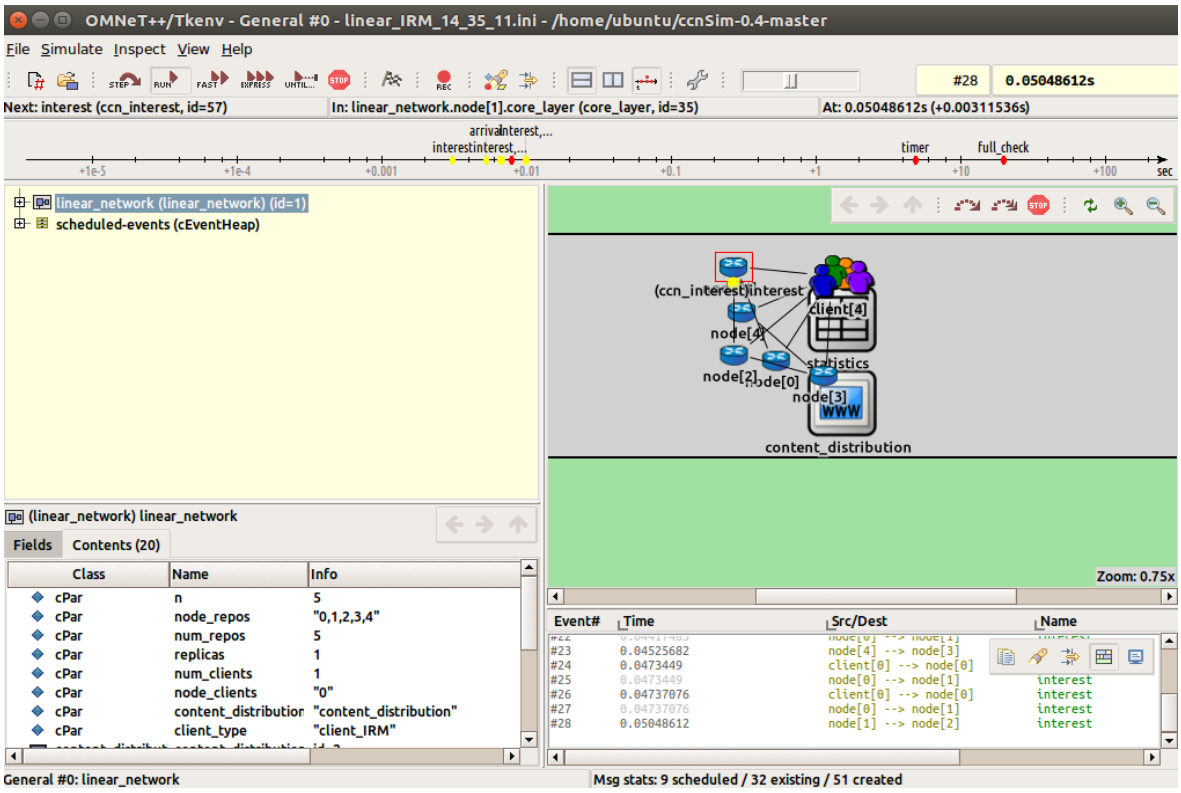
Figura 14 - ccnSim em execução em modo console



Fonte: Elaborada pelo Autor

Outra forma possível de executar uma simulação no ccnSim, ilustrada na Figura 15, é fazendo uso da janela gráfica do Omnet++ através do comando ***./ccnSim -f arquivo.ini.*** Esse método faz a animação gráfica da simulação permitindo a visualização dos nós do cenário, os pacotes trocados entre os nós, além de permitir iniciar e finalizar graficamente a simulação quando for desejado.

Figura 15 - ccnSim em execução em modo gráfico



Fonte: Elaborado pelo Autor

Por fim, uma última forma de executar simulações nessa ferramenta é através do comando ***./ccnSim -u Cmdenv -f arquivo.ini*** que executa a simulação em modo console permitindo a visualização das saídas geradas no *prompt* de comando do linux.

O ccnSim fornece uma variedade de estratégias de encaminhamento que permitem definir para onde e como serão encaminhados os pacotes de interesses que chegam em um nó da rede. Em relação ao seu cache, a ferramenta possibilita definir estratégias de decisão de armazenamento que são usadas para determinar se um conteúdo que chega no nó deve ser armazenado no cache local ou não. Outro recurso bastante útil é a possibilidade de configurar diferentes políticas de substituição de cache que permitem excluir os conteúdos quando é atingido o limite de armazenamento, considerando um determinado padrão ou conjunto de informações. A Tabela 5 exibe as opções disponíveis para cada um desses recursos fornecidos pelo simulador.

Tabela 5 - Tipos de estratégias no ccnSim

Estratégias de Encaminhamento (FS)	**Estratégias de Decisão (DS)**	**Estratégias de Substituição (RS)**
spr, random repository, nrr, nrr1	*lce, lcd, fixP, btw, two _lru, two_ttl, prob_cache, never costawareP, ideal_blind, ideal_costaware*	*lru_cache, ttl_cache, lfu_cache, random_cache, fifo_cache*

Fonte: Elaborado pelo Autor

As saídas são produzidas dentro da pasta ***/results***, na forma de arquivos ***.sca***, que mantêm informações sobre as configurações da simulação e dos elementos envolvidos nela. A evolução temporal do cenário simulado, juntamente com algumas métricas agregadas, é registrada no arquivo de descrição do cenário no diretório ***/logs***.

9. ANÁLISE QUALITATIVA

Apresentadas as ferramentas de prototipagem em redes centradas em conteúdo, seus principais recursos e exemplos práticos de utilização, elas são comparadas e analisadas em relação a alguns pontos estabelecidos que indicarão o quanto a ferramenta é útil ou não para um determinado experimento. Os pontos considerados nessa comparação são os parâmetros de configuração de rede, recursos comuns e recursos de configuração CCN.

É desejável que todas as ferramentas, sejam de simulação ou de emulação, apresentem um conjunto de parâmetros que permitam configurar a rede e assim reproduzir comportamentos de cenários de redes que se deseja imitar em um experimento. Dessa forma, as ferramentas apresentadas neste trabalho são comparadas em relação à apresentação ou não desses parâmetros. São eles: atraso, perda de pacotes e largura de banda. Na Tabela 6 é possível observar essa comparação entre as ferramentas. Percebe-se que todas possibilitam a configuração de todos os parâmetros estabelecidos, mostrando-se bastante eficazes na capacidade de configuração (*configurability*) dos parâmetros de rede.

Tabela 6 - Parâmetros de configuração de rede

Parâmetros	Mini-CCNx	ndnSIM	ccnSim
Atraso	Sim	Sim	Sim
Perda	Sim	Sim	Sim
Largura de Banda	Sim	Sim	Sim

Fonte: Elaborado pelo Autor

É interessante que alguns aspectos da simulação ou da emulação sejam configuráveis garantindo um certo nível de controle e flexibilidade aos experimentos. Para isso é necessário que as ferramentas apresentem mecanismos que possibilitem a capacidade de configuração do experimento de acordo com as necessidades dos usuários.

É desejável ainda que as ferramentas forneçam mecanismos que facilitem a experiência dos seus usuários quanto a utilização, deixando-os livre para se preocupar somente com detalhes da própria tecnologia que se deseja experimentar em vez de se preocupar com detalhes de configuração da ferramenta. Por fim, é interessante que o que acontece na rede durante o processo de experimentação seja devidamente registrado e armazenado em algum local para que possa ser visualizado posteriormente, permitindo a aplicação de um processo analítico em cima das saídas geradas, o que levará a uma conclusão acerca da tecnologia experimentada na ferramenta.

A Tabela 7 mostra alguns recursos presentes nas ferramentas, classificando os recursos em nível básico, intermediário ou avançado de acordo com as facilidades e possibilidades que eles proporcionam a experimentação.

Tabela 7 - Recursos comuns

Recursos Básicos	Mini-CCNx	ndnSIM	ccnSim
Capacidade de Configuração do experimento	Avançada	Avançada	Avançada
Interface gráfica	Avançada	Básica	Básica
Registros de saída	Básico	Avançado	Intermediário

Fonte: Elaborado pelo Autor

Inicialmente observa-se que todos apresentam um nível avançado de capacidade de configuração do experimento. No entanto as motivações que os levam a receber essa classificação são diferentes. Por ser um emulador talvez não seja tão perceptível a necessidade da existência de uma configuração de aspectos da emulação no Mini-CCNx. Mas ainda assim, essa necessidade pode vir a aparecer em algum momento. O parâmetro *appName* por exemplo, como já visto anteriormente, proporciona ao usuário a possibilidade de configurar elementos da emulação através de comandos e *scripts*, o que dá flexibilidade e um nível de controle ao experimento. No ndnSIM essa capacidade de configuração se manifesta através de classes e funções. Elementos como o tempo da simulação e o tamanho dos pacotes trocados, podem ser configurados nesse simulador. No ccnSim essa capacidade de configuração é algo identificável por meio de seus parâmetros, que dão a possibilidade de selecionar entre diferentes modos de simulação, além de parâmetros como configuração do tráfego.

O Mini-CCNx fornece interface para geração e configuração dos cenários enquanto o ndnSIM e ccnSim possibilitam somente a animação gráfica do cenário, característica que faz o emulador se destacar amplamente em relação aos dois simuladores. Isso indica o quanto o Mini-CCNx é desenvolvido em termos de usabilidade, tornando a experimentação facilitada através de uma interação transparente e direta entre o usuário, a rede e o protocolo CCN. Quanto à geração de saídas, todas as ferramentas fornecem esse recurso, no entanto, vale ressaltar que no ndnSIM existe uma variedade bem maior de *logs* que registram diversos aspectos da rede, dos nós e do próprio protocolo, destacando-se em relação aos concorrentes neste quesito.

Em um simulador de redes centradas em conteúdo, é interessante que haja mecanismos de configuração das estruturas apresentadas pelo protocolo. No caso da FIB é

indispensável a presença desses mecanismos já que será necessário estabelecer a comunicação baseada em nomes entre os nós. Além disso, a presença desse recurso contribui de forma significativa para a compreensão de como ocorre a comunicação na rede e de como a tecnologia experimentada funciona.

Outra forma de configuração possível da FIB é através de roteamento dinâmico fazendo uso de algoritmos de roteamento. Esta é uma opção interessante para casos em que se deseja realizar experimentos que reproduzam cenários que estabeleçam uma conectividade entre redes distintas, como em *intranets* e *extranets.* Dessa forma, pode ser interessante que haja algum recurso que torne possível esse tipo de configuração.

Para a PIT esse mecanismo de configuração é desnecessário, uma vez que essa estrutura é preenchida automaticamente baseada nos eventos da rede (troca de pacotes *Interests* e *Datas*). Também é interessante que o cache seja configurável principalmente quanto ao seu tamanho, tornando possível a reprodução de cenários onde existe uma variação de tamanhos do cache.

Estratégias de substituição do cache e estratégias de encaminhamento, apesar de não serem recursos essenciais, são opções interessantes que ampliam as possibilidades de experimentação em redes centradas em conteúdo permitindo a verificação do comportamento dos cenários com diferentes configurações.

Analisando a Tabela 8, percebe-se que a configuração da FIB é um recurso que está disponível para as três ferramentas. No Mini-CCNx isso pode ser feito pelo arquivo de configuração, pela ferramenta *miniccnxedit* ou por meio de um algoritmo de roteamento dinâmico como o OSPFN. No ccnSim a FIB é configurada por padrão pelo algoritmo de Dijkstra. Embora não existam implementações de algoritmos de roteamento fornecidos pelo simulador que possam ser selecionados para realizar o roteamento dinâmico no ccnSim, o algoritmo de Dijkstra se coloca como uma opção caso seja desejável realizar experimentos com esse recurso uma vez que ele executa essa ação de forma similar. Outro método possível de configurar a FIB no ccnSim é definir rotas entre os nós por meio uma matriz de roteamento definida em um arquivo, o que acaba sendo algo não trivial devido à inexistência de uma abordagem desse recurso em sua documentação por parte de seus desenvolvedores.

Tabela 8 - Recursos de configuração CCN

Capacidade de Configuração CCN	Mini-CCNx	ndnSIM	ccnSim
Configuração estática da FIB	Sim	Sim	Sim
Roteamento dinâmico	Sim	Não	Sim
Configuração do tamanho do Cache	Sim	Sim	Sim
Estratégias de substituição do Cache	Não	Sim	Sim
Estratégias de encaminhamento	Não	Sim	Sim

Fonte: Elaborada pelo Autor

No ndnSIM a configuração da FIB pode ser feita manualmente ou de forma automática por meio de classes e funções específicas para isso, mas não é possível realizar experimentos com roteamento dinâmico. Com exceção do Mini-CCNx, todas as outras ferramentas fornecem a possibilidade de experimentação com estratégias de substituição do cache e estratégias de encaminhamento, o que demonstra o quão ricas essas ferramentas são no que se refere ao fornecimento de recursos. Em relação à configuração do tamanho do cache todas as ferramentas possuem esses recursos embora as unidades de medidas para o tamanho sejam diferentes.

O Mini-CCNx é uma ferramenta completa para experimentação não deixando nada a desejar, uma vez que mantém todos os recursos fundamentais que uma ferramenta deve apresentar como foi possível verificar nas tabelas dos recursos apresentadas. Além disso, ela se destaca ao fornecer algumas peculiaridades em alguns recursos comuns a outras ferramentas como a possibilidade de geração e configuração de cenário pela interface gráfica.

O ndnSIM também não deixa nada a desejar sendo até mais completo do que o Mini-CCNx no que diz respeito ao fornecimento de recursos, mas ainda assim sua usabilidade é bastante inferior em relação ao Mini-CCNx e um pouco inferior ao ccnSim. Embora os métodos de construção do cenário e configuração sejam bastante intuitivos e a documentação contribua significativamente facilitando a compreensão, o fato de ter que construir e configurar os cenários através de código fonte ou mesmo por meio de um arquivo de texto pode representar um obstáculo no que se refere a usabilidade da ferramenta, principalmente na construção de cenários personalizados mais complexos. De qualquer forma esse detalhe não impacta de forma tão negativa, mesmo assim vale ressaltar que ela se coloca um pouco atrás em relação às demais nesse quesito já que existe a necessidade de compreender alguns elementos da sintaxe para configurá-los, o que exige um certo trabalho adicional em relação às outras ferramentas que fornecem a possibilidade de fazer isso de forma mais automática e interativa ou mesmo com menos trabalho manual.

O ccnSim se mostrou bastante completo ao fornecer os principais recursos levantados nas análises. Pode-se afirmar que ambos os simuladores, tanto o ndnSIM como o ccnSim, são bem ricos nesse quesito. Fazendo uma análise mais geral incluindo não somente as informações contidas nas tabelas acima, mas também outras informações apresentadas, pode-se perceber que ambas apresentam níveis parecidos quando se fala do fornecimento de recursos. Existem recursos comuns entre eles que podem ser usados com mais opções em somente um deles e também existem recursos peculiares que são fornecidos em somente um simulador, mas não no outro. Juntamente com o Mini-CCNx, ele está à frente do ndnSIM no requisito de usabilidade, apesar de que essa diferença é mínima. No entanto ele fica um pouco atrás na parte da aprendizagem do conceito uma vez que ao ser realizado um experimento no ccnSim a compreensão de como funciona a tecnologia pode ser um pouco prejudicada, pois a ferramenta tende a ser bastante abstrata nesse ponto.

No final, cada ferramenta apresenta alguma característica ou recurso que alguma outra não tem, tornando-as únicas. Estabelecendo um paralelo entre as três percebe-se que elas apresentam os principais recursos para experimentação em redes centradas em conteúdo, suas peculiaridades e as formas como estes recursos estão disponíveis e podem ser utilizados farão a diferença na hora de escolher uma delas.

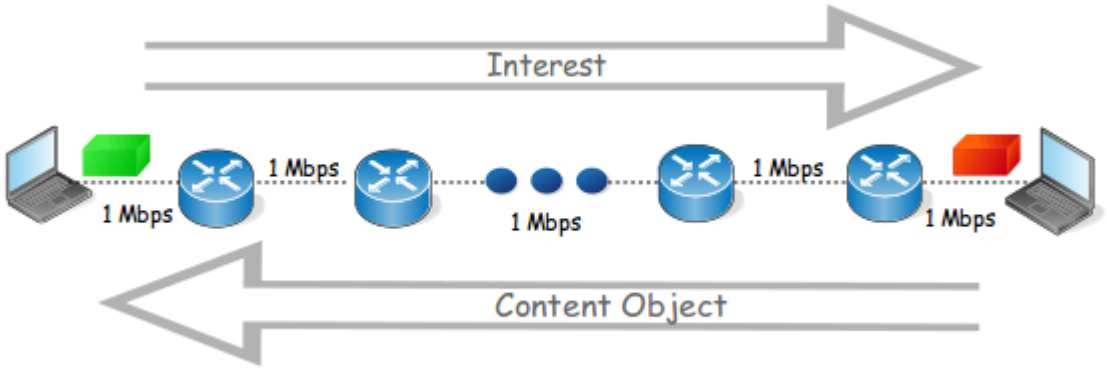
10. ANÁLISE QUANTITATIVA

Este capítulo as ferramentas selecionadas para a análise são experimentadas com a finalidade de avaliar o consumo de recursos totais e verificar como se comportam frente a um experimento tradicional. Portanto, esse trabalho adota uma abordagem um tanto diferente em relação a outros trabalhos de análise de ferramentas de prototipagem, uma vez que verificou o consumo dos recursos ao longo de um intervalo de tempo. A motivação para isso é proporcionar aos leitores uma ideia do comportamento das ferramentas no que se refere ao consumo de recursos ao ser realizado um experimento simples em um hardware com poucos recursos.

10.1 Condições e configurações do experimento

Os experimentos foram realizados em uma máquina de uso pessoal (processador AMD Dual Core com 4 GB de memória RAM) executando o Ubuntu 14.04. Para ambas as ferramentas, utilizou-se uma topologia do tipo linear na qual a quantidade de nós foi sendo variada em vinte, quarenta, sessenta, oitenta e cem nós. O tempo de execução foi de vinte minutos para cada ferramenta. A motivação para o uso de uma sequência aritmética na variação do número de nós é que ela permite identificar o padrão de crescimento do consumo dos recursos ao longo de um intervalo de tempo e assim estabelecer estimativas para a execução de experimento futuros.

Figura 16 - Cenário dos Experimentos



Fonte: Elaborada pelo Autor.

A Figura 16 ilustra o formato do experimento. Nele um nó consumidor em uma extremidade da rede envia pacotes *Interest* (solicitação de conteúdo) e na outra extremidade um nó produtor responde com pacotes do tipo *Object Content* (Conteúdo). Vale ressaltar que

para a comparação buscou-se deixá-las ao máximo em condições semelhantes em termos de capacidade de configuração e parâmetros, embora sejam conhecidas as diferenças arquiteturais e de implementação das mesmas. Nos casos em que não foi possível igualá-las utilizou-se valores arbitrários ou comuns a outros trabalhos com alguns ajustes para adaptar a pouca quantidade de recursos da máquina utilizado neste trabalho. A fim de evitar interferências por parte de processos de outras aplicações, somente a ferramenta envolvida na análise foi mantida em execução no momento da experimentação. Essas mesmas condições foram garantidas para as três ferramentas analisadas.

Embora tenham sido configurados nos cenários parâmetros de rede como largura de banda e atraso, informações relacionadas a rede não serão analisadas pois o foco é especificamente no consumo de recursos das ferramentas e não da tecnologia em si. O intuito de configurá-las foi somente para imitar um processo de experimentação em que normalmente esses parâmetros são configurados. Em relação às métricas analisadas neste trabalho, foram analisados o consumo de CPU, o consumo de memória e o tempo de inicialização para cada experimento de simulação ou emulação. Para a medição do consumo foi utilizada a ferramenta top, uma vez que tal ferramenta de medição nativa do linux possibilita a contabilidade dos recursos a serem analisados nesse trabalho seguindo as condições nele estipuladas para medição, conforme é possível identificar em Jack (2013).

No caso dos simuladores, quando executados surge um único processo associado a execução do mesmo. Assim foi medido o consumo desse processo que representa a ferramenta em execução. No caso do emulador, por se tratar de vários elementos envolvidos para representar um ambiente de emulação, existem vários processos e por isso foram somados os consumos de recursos desses processos por minuto ao longo do intervalo analisado.

A análise dessas informações permitirá aos leitores deste trabalho compreender a demanda de recursos ao ser realizado um experimento simples nas ferramentas abordadas aqui.

11. RESULTADOS

As informações referentes às saídas dos experimentos são resumidas nas tabelas apresentadas a seguir. A Tabela 9 mostra as informações sobre o consumo de CPU para as três ferramentas analisadas. Os dados são apresentados em porcentagem. Observando as informações relativas aos simuladores ndnSIM e ccnSim, é possível verificar que os dados não variam muito conforme aumenta o número de nós indicando que esse fator não influencia no consumo de CPU nessas ferramentas. Porém, ao observar o consumo do emulador Mini-CCNx, observa-se que o número de nós exerce influência no consumo sofrendo um aumento significativo conforme é aumentado o número de nós.

Na tabela, as colunas limite inferior e superior são apresentados com um nível de significância de 5% (intervalo de confiança para um nível de confiança de 95%). A proximidade existente entre esses dados confirma o que é possível observar as demais medidas apresentadas, ou seja, o grau de variabilidade é baixo e o conjunto das informações é bastante homogêneo, principalmente para os simuladores.

Tabela 9 - Consumo de CPU

Ferra-mentas	Número de nós	Média	Mediana	Desvio Padrão	Mínimo	Máximo	Tama-nho	Limite Inferior	Limite Superior
ndn-Sim	20	99,58%	99,60%	0,12%	99,40%	100,00%	20	99,53%	99,63%
	40	99,59%	99,60%	0,11%	99,40%	100,00%	20	99,54%	99,63%
	60	99,50%	99,60%	0,30%	98,20%	99,60%	20	99,36%	99,63%
	80	99,60%	99,60%	0,10%	99,50%	100,00%	20	99,55%	99,64%
	100	99,53%	99,60%	0,18%	99,00%	100,00%	20	99,45%	99,61%
ccnSim	20	99,40%	99,70%	1,20%	94,20%	99,80%	20	98,88%	99,92%
	40	99,70%	99,70%	0,10%	99,50%	100,00%	20	99,65%	99,74%
	60	99,64%	99,70%	0,17%	99,20%	100,00%	20	99,56%	99,71%
	80	99,67%	99,70%	0,11%	99,50%	100,00%	20	99,62%	99,72%
	100	99,71%	99,70%	0,09%	99,50%	100,00%	20	99,67%	99,75%
mini-CCNx	20	19,14%	19,10%	0,39%	18,50%	20,00%	20	18,97%	19,31%
	40	32,82%	32,75%	0,32%	32,20%	33,50%	20	32,68%	32,96%
	60	48,74%	48,65%	0,73%	47,50%	50,40%	20	48,42%	49,06%
	80	63,27%	63,50%	2,48%	55,40%	66,30%	20	62,18%	64,35%
	100	80,96%	80,70%	1,64%	76,20%	83,30%	20	80,24%	81,68%

Fonte: Elaborada pelo Autor

A Tabela 10 mostra as informações sobre o consumo de memória para as três ferramentas. Os valores são apresentados em Megabytes. Ao observar os dados, percebe-se que o nível de dispersão é bem maior se comparado aos dados presentes na tabela do consumo de CPU. Além disso, verifica-se que ocorre uma variação no consumo de memória conforme aumenta-se o número de nós.

Novamente as colunas limites inferior e superior são apresentadas com um nível de significância de 5% (intervalo de confiança para um nível de confiança de 95%).

Analisando o intervalo entre eles é possível confirmar o alto grau de variabilidade desses dados, caracterizando um conjunto bastante heterogêneo tanto para os simuladores como para o emulador.

Tabela 10 - Consumo de Memória

Ferra-mentas	Número de nós	Média	Mediana	Desvio Padrão	Mínimo	Máximo	Tama-nho	Limite Inferior	Limite Superior
ndn-Sim	20	369	370	147	126	612	20	305	434
	40	383	383	149	140	629	20	317	448
	60	389	390	144	150	629	20	326	452
	80	395	398	135	161	622	20	336	454
	100	395	397	128	157	594	20	339	451
ccnSim	20	438	472	127	171	605	20	383	494
	40	307	316	74	150	412	20	275	340
	60	395	428	136	91	570	20	336	455
	80	421	442	118	161	584	20	369	473
	100	221	224	43	129	283	20	202	240
mini-CCNx	20	314	322	62	192	391	20	287	341
	40	585	601	122	339	744	20	532	639
	60	856	884	188	472	1094	20	774	938
	80	1076	1164	243	605	1447	20	970	1183
	100	1356	1447	297	748	1796	20	1226	1486

Fonte: Elaborada pelo Autor

A Tabela 11 mostra a média do tempo de inicialização das ferramentas de acordo com a variação do número de nós. Como é possível observar, os tempos para o Mini-CCNx estão bem acima dos tempos de inicialização do ndnSIM e ccnSim, o que é bastante compreensível considerando as diferenças arquiteturais e de implementação entre elas. Nesse experimento foi necessário executar no Mini-CCNx um script de configuração por cada nó instanciado para executar uma série de configurações de forma que a aplicação de tráfego *ccntraffic* viesse a funcionar, o que contribui para o aumento desse tempo. Para o ndnSIM as variações no tempo ficam na casa dos milissegundos, sendo praticamente indiferente a variação no número de nós da topologia. Já o ccnSim, a maioria dos tempos estão na casa dos milissegundos podendo chegar a um ou mais minutos de acordo com o número de nós como pode ser observado na tabela em uma das variações.

Tabela 11 - Tempo de Inicialização

Número de Nós	Mini-CCNx	ndnSIM	ccnSim
20	15.33s	13.71s	26s
40	35s	13.57s	24.66s
60	57.66s	13.67s	1m37s
80	1m36s	13.71s	26.33s
100	2m13s	13.97s	26s

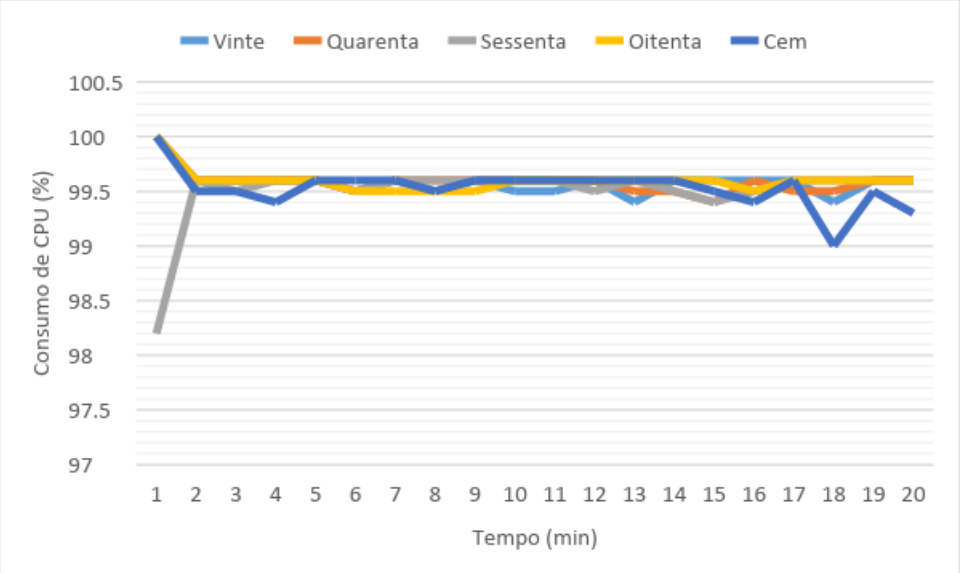
Fonte: Elaborada pelo Autor

12. ANÁLISE DOS RESULTADOS

Os gráficos apresentados a seguir mostram mais claramente o padrão de consumo dos elementos analisados. Para melhor identificar esse padrão, optou-se por utilizar gráficos de linhas que permitem identificar um comportamento de um conjunto de informações em uma sequência temporal.

O Gráfico 1 apresenta o consumo de CPU para o simulador ndnSIM. Analisando-o é possível observar que as linhas que representam as saídas para os diferentes números de nós estão bem próximas, sobrepondo-se em alguns momentos, indicando que as saídas se mantêm iguais ou semelhantes ao longo do intervalo de tempo independentemente do número de nós. Esse padrão é coerente com as informações encontradas na Tabela 9, que contém informações relacionadas ao consumo de CPU. Os simuladores atuam como uma aplicação que imitam o comportamento de uma tecnologia, por isso não importa muito a variação do número de nós, a demanda de consumo da aplicação permanecerá praticamente a mesma com pequenas oscilações. Isso significa que se fosse gerada uma topologia com mil nós, o padrão de consumo de CPU permaneceria o mesmo. Na verdade, para mudar esse padrão de consumo seria necessário modificar parâmetros que estimulem maior demanda de processamento como o número de pacotes circulantes na rede. Esse é o padrão de consumo de CPU que será encontrado ao ser realizado um experimento no ndnSIM.

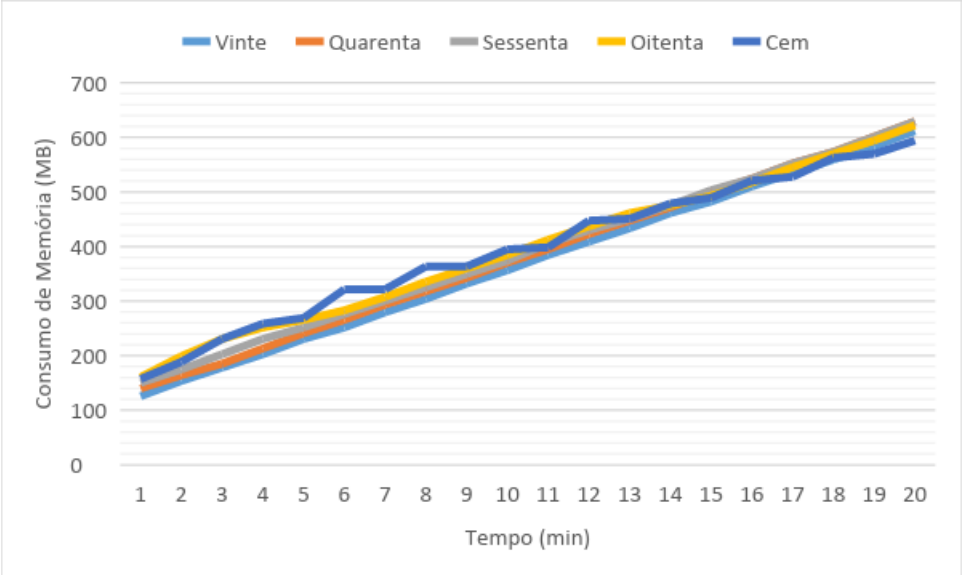
Gráfico 1 - Consumo de CPU no ndnSIM



Fonte: Elaborado pelo Autor

O Gráfico 2 apresenta o consumo de memória para o ndnSIM em Megabytes. Verifica-se que as linhas que representam a variação do número de nós encontram-se bem próximas, embora minimamente dispersas em alguns trechos de forma que é possível diferenciá-las e em outros estão levemente sobrepostas. De qualquer forma, essa pequena dispersão mostra que a variação do número de nós leva a um crescimento no consumo de memória. A inclinação das linhas de baixo para cima indicam que o consumo é crescente e mais espaço de memória é usado com o passar do tempo.

Gráfico 2 - Consumo de Memória no ndnSIM



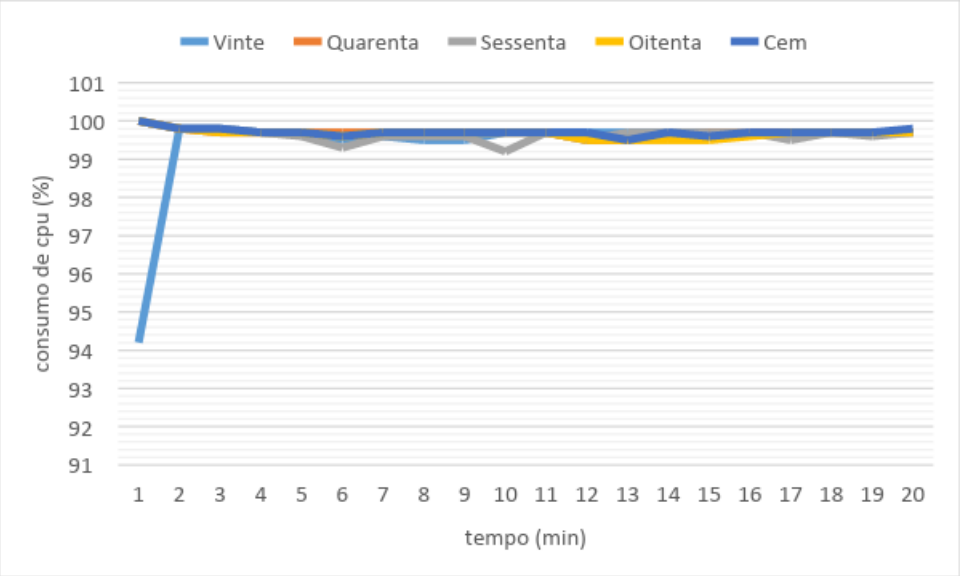
Fonte: Elaborado pelo Autor

O aumento provocado pela variação do número de nós ocorre porque cada nó criado na topologia nada mais é que uma instância de um objeto criado a partir de uma classe, de forma que um aumento no número de nós demanda cada vez mais espaço para armazenar mais objetos. A inclinação das linhas ocorre por causa da quantidade crescentes de pacotes que circulam na rede, pois ao tratar mais informações cada vez mais espaço será necessário para armazená-las até que chegue o momento ideal para serem processadas.

Portanto, diferentemente do processamento, a memória é o fator limitante para a experimentação no ndnSIM, uma vez que o aumento constante proporcionado pela variação do número de nós ou quantidade de pacotes na rede tende a alcançar o limite do recurso em algum momento causando a paralisação ou lentidão no sistema. Esse é o padrão de consumo de memória encontrado ao ser realizado um experimento no ndnSIM.

O Gráfico 3 apresenta o consumo de CPU para o simulador ccnSim. Ao observá-lo, percebe-se que suas linhas se encontram sobrepostas em alguns trechos, bem mais até quando comparadas com as linhas que representam o consumo de CPU no ndnSIM, impossibilitando a visualização da maioria delas nesses trechos. Esse padrão de consumo extremamente repetitivo coincide com as informações contidas na Tabela 9 que mantém as informações relacionadas ao consumo de CPU no ccnSim. Como já visto, esse padrão indica que a variação na quantidade de nós não é um fator determinante para a simulação. A mesma explicação adotada para esclarecer esse padrão de consumo de CPU no ndnSIM pode ser aplicada aqui, ou seja, por se tratar de uma ferramenta de simulação ela irá atuar como uma aplicação que imita a tecnologia experimentada e manterá um consumo de CPU praticamente inalterado ao longo da simulação, sofrendo algumas pequenas oscilações no consumo ao longo do intervalo de tempo.

Gráfico 3 - Consumo de CPU no ccnSim

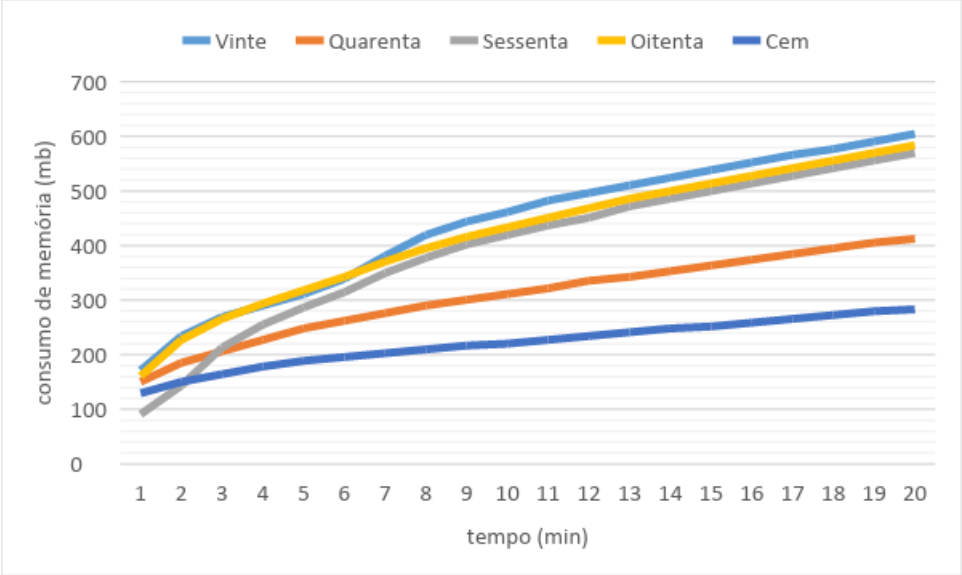


Fonte: Elaborado pelo Autor

O Gráfico 4 apresenta o consumo de memória no ccnSim em Megabytes. Suas linhas se encontram dispersas na maior parte do intervalo, de forma que é possível identificá-las claramente. Assim é possível concluir que a variação do número de nós impacta o consumo de memória. Também é possível observar que o padrão de consumo é bastante incoerente. Normalmente espera-se que o aumento do número de nós leve a um aumento no consumo de memória, o que não ocorre como é possível visualizar no gráfico. O que se percebe é que o cenário com maior quantidade de nós consome menos memória enquanto o demais mantém um consumo aproximado entre eles ou mesmo intermediário em relação ao

cenário com maior número de nós. Provavelmente esse comportamento está associado a algum algoritmo utilizado pela ferramenta fazendo com que ela apresente essa característica peculiar no consumo de memória.

Gráfico 4 - Consumo de Memória no ccnSim

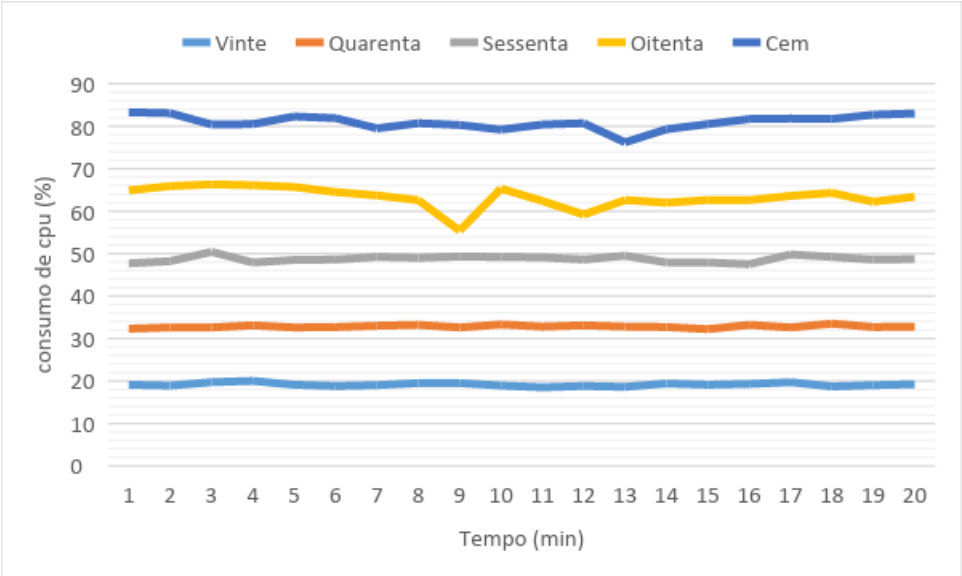


Fonte: Elaborado pelo Autor

O Gráfico 5 apresenta o padrão de consumo de CPU para o emulador Mini-CCNx. Ao observá-lo, verifica-se que suas linhas estão posicionadas paralelamente e com um distanciamento significativo uma em relação a outra, indicando que há um crescimento relevante no consumo de CPU quando é variado o número de nós. Esse padrão coincide com as informações contidas na Tabela 9 que contém as informações de consumo de CPU do Mini-CCNx.

Enquanto no ndnSIM todos os componentes são objetos instanciados a partir de uma classe em C++, no Mini-CCNx eles se resumem a um agrupamento de elementos presentes no sistema que acessam seus recursos para representar um ambiente de rede. Portanto, ao aumentar o número de componentes, consequentemente aumentará a demanda de processamento por parte dos elementos da rede, resultando no padrão de consumo visualizado no gráfico.

Gráfico 5 - Consumo de CPU no Mini-CCNx

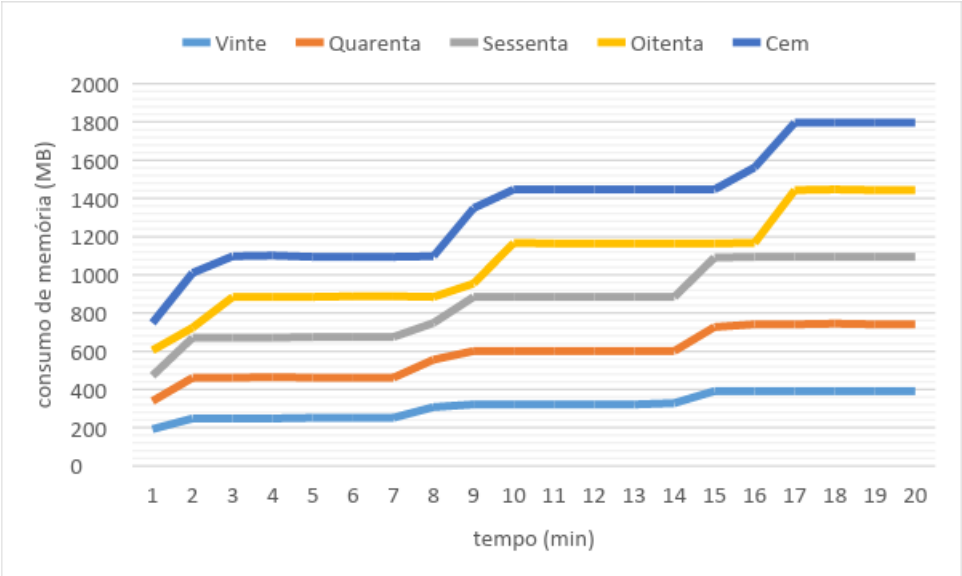


Fonte: Elaborado pelo Autor

Por fim, o Gráfico 6 mostra o padrão de consumo da memória para o Mini-CCNx. Suas linhas estão dispostas de uma forma que lembram o formato de degraus e existe um afastamento considerável entre elas. Para esse afastamento, a explicação dada anteriormente para o consumo de CPU no Mini-CCNx pode ser aplicada aqui, com a informação adicional de que quanto mais componentes gerados maior será a demanda de memória. Para o aumento gradual em cada linha interpreta-se que isso ocorre porque o *daemon* ccnd passa a ser mais requisitado com o passar do tempo para gerenciar as estruturas características da tecnologia como PIT, FIB e CS, em decorrência do acúmulo de informações da rede.

A quantidade de nós utilizada nestes experimentos pode ser encontrada na maioria dos experimentos a serem realizados, dos mais simples até os mais complexos. Ao observar o consumo de CPU das três ferramentas considerando o total disponível da CPU (200% para uma CPU de dois núcleos) pode-se concluir que elas apresentam um resultado satisfatório, uma vez que os valores registrados se mantiveram no máximo até metade desse total disponível. A expectativa para os simuladores é que eles mantenham esse mesmo padrão de consumo diante da variação do número de nós, porém o emulador obviamente terá um salto significativo caso isso ocorra, o que é compreensível pelas suas diferenças arquiteturais já citadas em seções anteriores.

Gráfico 6 - Consumo de Memória no Mini-CCNx



Fonte: Elaborado pelo Autor

Sobre o consumo de memória também pode-se concluir que é bastante satisfatório para as configurações da máquina (4 GB de memória RAM) utilizada nos experimentos. Percebe-se que os intervalos dos valores que representam o consumo de memória dos simuladores estão bem próximos, apesar dos seus padrões de consumo serem diferentes. Já o emulador apresenta valores mais altos no eixo relativo ao consumo de memória indicando maior consumo para uma mesma quantidade de nós nas outras ferramentas em um mesmo período, comportamento esse que já é esperado por apresentar características diferentes no que diz respeito a sua arquitetura e implementação.

13. CONSIDERAÇÕES FINAIS

Neste trabalho as ferramentas selecionadas a partir dos critérios preestabelecidos foram analisadas em função dos recursos fornecidos e dos recursos de hardware consumidos. Observou-se que todas as ferramentas selecionadas apresentam recursos suficientes para explorar as características fundamentais da tecnologia, embora deixem sempre algo a desejar em algum momento. Observou-se também que os simuladores se destacam ao fornecer uma maior quantidade de recursos e com uma diversidade bem maior que o emulador, o que é perfeitamente compreensível uma vez que se tratam de ferramentas implementadas como um *software* o que possibilita a criação de uma série de recursos personalizados de acordo com as necessidades e o comportamento que se deseja imitar. Por outro lado, se não existe uma variedade de recursos muito amplo para experimentação no Mini-CCNx ele compensa apresentando outros recursos como geração de cenário por meio da interface gráfica e roteamento dinâmico.

Analisando o consumo dos recursos de hardware, verifica-se que ao realizar experimentos comuns em uma máquina de baixo custo elas se saem muito bem não sobrecarregando o sistema, de forma que é possível trabalhar com alguns nós nos experimentos em máquina comuns.

Embora o uso da ferramenta de simulação ndnSIM seja relativamente fácil, existe uma série de detalhes sobre os quais o usuário leigo na tecnologia de redes centradas em conteúdo deverá estar atento, demandando um certo tempo e esforço no estudo dos principais componentes e APIs da ferramenta até que passe a entrar em contato com a tecnologia em si. Essa complexidade pode não ser aceitável para alguns usuários, pois uma forma de estudo ideal seria entrando em contato diretamente com um exemplo prático de funcionamento com poucos obstáculos permitindo partir quase que diretamente para a experimentação. Com o ccnSim, embora exista menos trabalho na definição do experimento essa questão também está um pouco presente no seu processo de experimentação. Nesse contexto a ferramenta de emulação Mini-CCNx se apresenta como adequada para essa situação devido a sua capacidade de abstração no fornecimento dos recursos e mecanismos de experimentação sem comprometer a compreensão do conceito nem requerer esforço adicional de configuração.

Diferentes formas de utilização podem ser adotadas de acordo com o que se pretende. Caso o objetivo seja compreender em um período curto de tempo a tecnologia estudada a ferramenta sugerida é o Mini-CCNx uma vez que ela fornece um ambiente de rede centrada em conteúdo altamente abstrato ao usuário com a possibilidade de configurar os principais parâmetros de uma rede do modelo, atendendo assim a um dos requisitos de

implementação estabelecidos. Caso o objetivo seja propor e testar soluções algorítmicas novas ou já existentes, identificar diferentes padrões de tráfego para esse modelo de rede ou haja a necessidade de uma ferramenta que tenha como característica a extensibilidade, o ndnSIM é o sugerido de acordo com a avaliação realizada neste trabalho, pois fornece uma série de recursos e APIs para tais atividades facilitando o processo de implementação e análise. Se o objetivo for realizar experimentos com diferentes abordagens da tecnologia com um alto nível de capacidade de configuração e combinação de recursos, o ccnSim será o mais adequado para isso.

Para trabalhos futuros, sugere-se que sejam realizados experimentos em máquinas mais potentes testando diferentes cenários e métricas de rede e que os recursos apresentados sejam explorados para elaboração de novas propostas. É sugerido ainda que sejam acrescentadas melhorias nas ferramentas principalmente em alguns pontos abordados nesse trabalho a fim de contribuir para a constante evolução das mesmas.

No Mini-CCNx por exemplo poderiam ser acrescentados recursos que possibilitem a visualização de uma maior diversidade de informações do experimento em seus registros de saída. Outra melhoria possível, sendo inclusive sugerida pelo seu desenvolvedor, seria a incrementação da ferramenta gráfica miniccnxedit para suportar o ambiente completo de execução do Mini-CCNx, uma vez que algumas operações relacionadas a execução do experimento ainda necessitam ser efetuadas através de linha de comando. No caso do ccnSim poderiam ser incluídos recursos que permitissem a padronização/automatização de uma série de parâmetros de configuração da simulação de forma a tornar a experimentação mais abstrata a seus usuários. Além disso sua adaptação para geração de cenários a partir do ambiente gráfico do Omnet++, exerceria um enorme impacto positivo sobre sua usabilidade. Essa mesma sugestão vale para o ndnSim, o que o tornaria uma ferramenta ainda mais completa.

14. REFERÊNCIAS

AFANASYEV, A. et al. NDN Technical Report: NFD Developer's Guide-NDN, Technical Report NDN-0021, **Revision 5**. 2015.

BARBOSA, J. J. A. **Estudo e Análise das Redes Centradas em Conteúdos**. 2014. 110 f. Dissertação de Mestrado – Instituto Superior de Engenharia do Porto, Porto. 2014.

CABRAL, C. M. S. **Mini-CCNx:** uma plataforma de prototipagem rápida para Redes Orientadas a Conteúdo. Dissertação (Mestrado) — Universidade Estadual de Campinas (UNICAMP), Faculdade de Engenharia Elétrica e de Computação (FEEC), 2013.

CABRAL, Carlos. **Mini-CCNx**. Disponível em: https://github.com/chesteve/mn-ccnx/wiki. Acesso em: 07 dez. 2015.

CCNSIM. **CcnSim-v0.4 User Manual**. Disponível em: https://github.com/TeamRossi/ccnSim-0.4/blob/master/ccnSim-v0.4-Manual.pdf. Acesso em: 07 fev. 2018.

CCNx. **Content-Centric Networking CCNx Reference Implementation**. 2017. Disponível em: https://github.com/ProjectCCNx/ccnx. Acesso em: 07 dez. 2015.

CHIOCCHETTI, Raffaele; ROSSI, Dario; ROSSINI, Giuseppe. ccnsim: An highly scalable ccn simulator. In: **Communications (ICC), 2013 IEEE International Conference on**. IEEE, 2013. p. 2309-2314.

DE BRITO, G. M. **Uma Análise de Desempenho de Redes Orientadas a Conteúdo Sem-Fio**. 2014. 69 f. Dissertação de Mestrado – Universidade Federal Fluminense, Niterói. 2014.

DE BRITO, Gabriel M.; VELLOSO, Pedro B.; MORAES, Igor M. Redes orientadas a conteúdo: Um novo paradigma para a Internet. **Minicursos do Simpósio Brasileiro de Redes de Computadores-SBRC**, v. 2012, p. 211-264, 2012.

JACK. **Como funciona a contabilidade do consumo de CPU no Linux?** Disponível em: https://jack.eti.br/como-funciona-a-contabilidade-do-consumo-de-cpu-no-linux/. Acesso em: 13 jun. 2017.

JACOBSON, Van et al. Networking named content. In: **Proceedings of the 5th international conference on Emerging networking experiments and technologies**. ACM, 2009. p. 1-12.

JAIN, Raj. **The art of computer systems performance analisys:** technices for experimental design, measurement, simulation, and modeling. Jonh Wiley & Sons. 1991.

MASTORAKIS, Spyridon et al. ndnSIM 2.0: A new version of the NDN simulator for NS-3. **NDN, Technical Report NDN-0028**, 2015.

MOSKO, Marc. Ccnx 1.0 protocol introduction. **Palo Alto Research Center, Inc**, v. 10, 2014.

NDN. **Named Data Networking**. Disponível em: https://named-data.net/. Acesso em: 03 jul. de 2016.

NDNSIM. **Getting Started**. Disponível em: http://ndnsim.net/current/getting-started.html. Acesso em: 23 out. 2016.