UNIVERSIDADE FEDERAL DO PAMPA

RAFAEL DUARTE BELTRAN

ICN STAGE: UM AMBIENTE PARA AVALIAÇÃO EXPERIMENTAL DE ICN

Alegrete
2022

RAFAEL DUARTE BELTRAN

ICN STAGE: UM AMBIENTE PARA AVALIAÇÃO EXPERIMENTAL DE ICN

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Software como requisito parcial para a obtenção do tlo de Mestre em Engenharia.

Orientador: Rodrigo Brandão Mansilha
Co-orientador: Diego Luis Kreutz

Alegrete
2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

B453i Beltran, Rafael
ICN STAGE: UM AMBIENTE PARA AVALIAÇÃO EXPERIMENTAL DE ICN / Rafael Beltran.
51 p.

Dissertação(Mestrado)-- Universidade Federal do Pampa, MESTRADO EM ENGENHARIA DE SOFTWARE, 2022.
"Orientação: Rodrigo Mansilha".

1. Sistemas distribuídos. 2. Tolerância a falhas. 3. Experimentação. 4. ICN. I. Título.

RAFAEL DUARTE BELTRAN

ICN Stage: Um ambiente para avaliação experimental de ICN

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Mestre em Engenharia de Software.

Dissertação defendida e aprovada em: 18 de outubro de 2022.

Banca examinadora:

Prof. Dr. Rodrigo Brandão mansilha - Orientador
UNIPAMPA

Prof. Dr. Diego Kreutz - Co-Orientador
UNIPAMPA

Prof. Dr. Claudio Claudio Schepke
UNIPAMPA

Prof. Dr. Bruno Lopes Dalmazo
FURG

Assinado eletronicamente por **RODRIGO BRANDAO MANSILHA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 26/10/2022, às 10:11, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Bruno Lopes Dalmazo**, **Usuário Externo**, em 26/10/2022, às 10:14, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **CLAUDIO SCHEPKE**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 26/10/2022, às 10:19, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **DIEGO LUIS KREUTZ**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 26/10/2022, às 10:27, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0969389** e o código CRC **CADB03CC**.

Dedico este trabalho aos meus pais, Paulo e Ana, por sempre acreditar, amar os filhos e incentivar a terminar tudo aquilo que começamos.
Ao meu irmão, Paulo Roberto, pela preocupação e apoio.
A minha noiva Sheila, por toda a compreensão, incentivo e amor.
Aos meus sobrinhos, Joaquim e Valentina pelos momentos de descontração e renovação.

AGRADECIMENTO

Ao Prof. Dr. Rodrigo Brandão Mansilha por todo apoio, incentivo e momentos de reflexão. Sei que vários momentos não atenderam as expectativas, mas, bastava uma conversa com o Sr. para eu poder encontrar novamente o caminho e seguir com o trabalho. Enfim, muito obrigado por acreditar em mim. Tenho certeza que não chegaria nesse ponto sem sua orientação.

Ao Prof. Mrs. Diego Luis Kreutz que aceitou ser meu coorientador e contribuir com o desenvolvimento e avaliações da minha dissertação.

Ao Prof. Dr. Cláudio Schpeke e ao Prof. Dr. Bruno Lopes Dalmazo que gentilmente aceitaram fazer parte da minha banca e colaborar com esta dissertação.

Ao meu pai deixo um agradecimento especial por todos ensinamentos que me deu durante sua vida, nunca vou esquecer todos momentos felizes que tivemos. A minha mãe por ser uma pessoa forte e dedicada a tudo o que faz, sempre compreensiva e carinhosa. Ambos sempre proporcionaram um ambiente de ensinamentos, amizade e amor para a mim e ao meu irmão.

Aos grandes amigos que fiz durante a graduação e mestrado, Gustavo e Daniel pelas noites de trabalho e risadas fazendo trabalhos juntos e contribuindo um com o outro.

Aos amigos que conquistei durante a vida, Bruno, Elionai, Eduardo, Mario e Mateus por nossos churrascos, momentos de diversão e Dota.

Por fim, a todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação, o meu muito obrigado.

“Sucesso é a soma de pequenos esforços, dia a dia.”
— Robert Collier

RESUMO

Após anos de pesquisa e desenvolvimento baseados principalmente em metodologias analíticas e em simulações, a comunidade da área de Information-Centric Networking (ICN) tem investido esforços em avaliações experimentais em direção à implantação da tecnologia. Porém, essa metodologia envolve desafios relacionados ao ambiente de testes, como heterogeneidade e falhas de nós e enlaces. Nesta dissertação propomos um arcabouço denominado ICN-STAGE para facilitar a execução de experimentos ICN distribuídos reprodutíveis e tolerantes a falhas. No ICN-STAGE, os experimentos são realizados como uma peça teatral com processos (papéis) interpretados por nós (atores) em um testbed (palco). Um controlador (diretor) instrumenta os atores, garante a coordenação global, detecta as falhas dos atores e os substitui conforme necessário. Resultados obtidos em Minikube e em FIBRE evidenciam a viabilidade e potencialidades do ICN-STAGE.

Palavras-chave: Sistemas distribuídos. Tolerância a falhas. Experimentação. ICN.

ABSTRACT

Keywords: After years of research and development based mainly on analytical and simulational methodologies, the ICN community has invested efforts in experimental evaluations towards the implantation of the technology. However, this methodology involves challenges related to the testbed, such as heterogeneity and faults of nodes and links. To overcome these challenges, we present ICN-STAGE, a framework for reproducible and fault-tolerant distributed ICN experiments. In ICN-STAGE, experiments are plays with processes (roles) enacted by nodes (actors) in a testbed (stage). A controller (director) instruments actors, ensures global coordination, detects actor failures, and replaces them as needed. Results from Mininet and FIBRE evidence ICN-STAGE's feasibility and potentialities .

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CCN	Content Centric Networking
HICN	Hybrid Information-Centric Networking
ICN	Information-Centric Networking
IP	Internet Protocol
NDO	Named Data Objects
NDN	Named Data Networking
NTP	Network Time Protocol
PTP	Precision Time Protocol
RTP	Real-Time Protocol
UTC	Universal Coordinated Time

SUMÁRIO

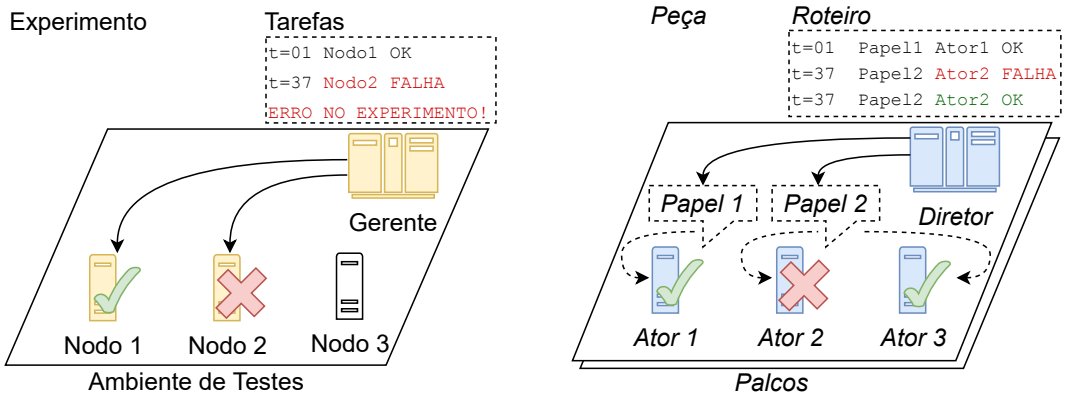
1 INTRODUÇÃO

As Redes Centradas em Informação (*Information Centric Networks* - ICN) são uma proposta para a Internet do Futuro com ênfase em disseminação eficiente e segura de conteúdos. A comunidade científica interessada em ICN tem caminhado em direção a realização de avaliações baseadas em metodologia experimental em ambiente real (RAHMAN et al., 2020; NICHOLS, 2019; Lim et al., 2018). Porém, alcançar resultados reproduzíveis obtidos por experimentos realizados em ambientes de testes distribuídos ainda está longe de ser trivial por muitos fatores, incluindo reprodução de configurações e ocorrência de falhas durante a execução (BAJPAI et al., 2019). A literatura contém exemplos de esforços para orquestrar experimentos ICN (HYUNWOO et al., 2015a; De Benedetto; ARUMAITHURAI; FU, 2017), mas pouco ainda foi feito em direção ao provimento de tolerância a falhas.

Falhas podem ser classificadas como intermitente (*i.e.*, quando sistema oscila entre modo falho e modo correto) ou permanente (*i.e.*, uma vez no modo falho, o sistema não retorna para modo correto) e podem ocorrer em nós ou enlaces. Essas falhas podem gerar erros durante a realização do experimento e, consequentemente, defeito nas análises dos resultados. Na Figura 1(a) é apresentado um diagrama onde temos alguns *Nodos* realizando tarefas distribuídas e ocorre uma falha irrecuperável. Uma maneira custosa e pouco produtiva de evitar o defeito na etapa de análise é repetir experimentos até obter resultados sem erros. Porém, repetir experimentos pode ser demasiadamente custoso ou inócuo na medida que novas falhas podem ocorrer, especialmente em experimentos de larga escala ou de longa duração. Uma segunda maneira de evitar defeito é corrigir conjuntos de dados defeituosos usando, por exemplo, técnicas de inteligência artificial (PAIM et al., 2021; PAIM et al., 2022).

Uma terceira maneira de evitar defeito é prevenir falhas. Alguns trabalhos (SANTOS; FERNANDES; KAMIENSKI, 2014; GARRETT; BONA; JR, 2017) propõem métodos para selecionar os nós mais confiáveis do ambiente de testes com base no histórico de desempenho. Uma quarta maneira de evitar defeito é tolerar falha em tempo de execução. Nesse sentido, alguns esforços de pesquisa avançaram para detectar falhas, mas carecem de mecanismos de recuperação (RUIZ et al., 2013; IMBERT et al., 2013).

Em (JUNIOR; CORDEIRO; GASPARY, 2018), foi proposto o EasyExp, um arcabouço tolerante a falhas para avaliação experimental de sistemas distribuídos. Nele, o usuário expressa experimentos sistematicamente, como uma sequência de ações crono-



(a) Coordenação de peças

(b) Coordenação de peças

Figura 1 – Diagramas do modelo convencional (a) de execução de experimentos distribuídos e (b) na versão ICN-STAGE

metradas. Um coordenador global garante que os nós executem suas ações conforme o esperado e acompanhe eventuais falhas e recupere falhas para evitar erros nos resultados.

Nesta dissertação, apresento o ICN-STAGE: um arcabouço para o provimento de tolerância a falhas e reprodutibilidade de experimentos ICN em sistemas distribuídos. O ICN-STAGE orquestra um experimento (peça) com um conjunto de processos (papéis) executados por nós virtuais ou físicos (atores) em um ambiente de testes (palco). Um controlador distribuído (diretor) garante a coordenação global da peça, detecta falhas de atores e os substitui conforme necessário como apresentado na Figura 1(b). Para implementar o ICN-STAGE, estendemos e especializamos o EasyExp para prover tolerância a falhas de diretor e suportar experimentos ICN, respectivamente. Também são apresentados os testes realizados no ICN-STAGE demonstrando sua capacidade de tolerância a falhas e reprodutibilidade de experimentos. As demonstrações utilizaram dois palcos específicos: ambiente local usando Minikube[1] e em ambiente distribuído usando FIBRE[2]. Os experimentos realizados foram desenvolvidos utilizando a arquitetura de ICN denomidada *Named Data Networking* (NDN)[3] e *Hybrid Information-Centric Networking* (hICN)[4].

O restante deste trabalho está estruturado da seguinte forma. No Capítulo 2 são explicados alguns conceitos básicos para contextualizar o trabalho. No Capítulo 3 comentamos alguns trabalhos próximos a nossa proposta. No Capítulo 4 é explicado o funcionamento do ICN-STAGE. No Capítulo 5 são apresentados a metodologia e os resultados da avaliação. Por fim, no Capítulo 6 são tecidas as conclusões.

[1] <https://minikube.sigs.k8s.io/>
[2] <https://portal.fibre.org.br/>
[3] <https://github.com/named-data>
[4] <https://github.com/FDio/hicn>

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos principais envolvidos neste trabalho. Na Seção 2.1 são apresentados alguns dos conceitos fundamentais de sistemas distribuídos. Na Seção 2.2 é explicada a arquitetura ICN que é o conceito principal abordado no trabalho.

2.1 Sistemas distribuídos

Sistemas distribuídos são elementos computacionais independentes e autônomos. Esses componentes são computadores também conhecidos como nós, e precisam se coordenar entre si. A conexão entre nós pode ocorrer tanto em *hardware* quanto *software*. A comunicação entre os componentes é feita através do envio de mensagens entre eles.

Para um usuário, há uma camada de abstração nas aplicações. Isso torna o sistema distribuído transparente, ou seja, o usuário não sabe com qual nó da aplicação ele está interagindo. Se houver alguma indisponibilidade a nível de *software* ou *hardware* em algum nó, isso não afetará o usuário.

Na Figura 2 podemos ver um sistema distribuído implementado em uma camada de software conhecido como *middleware*. Os computadores 2 e 3 estão executando a mesma aplicação B de forma distribuída. Os computadores 1 e 4 estão rodando aplicações diferentes, porém, através do *middleware* as aplicações A, B e C podem se comunicar.

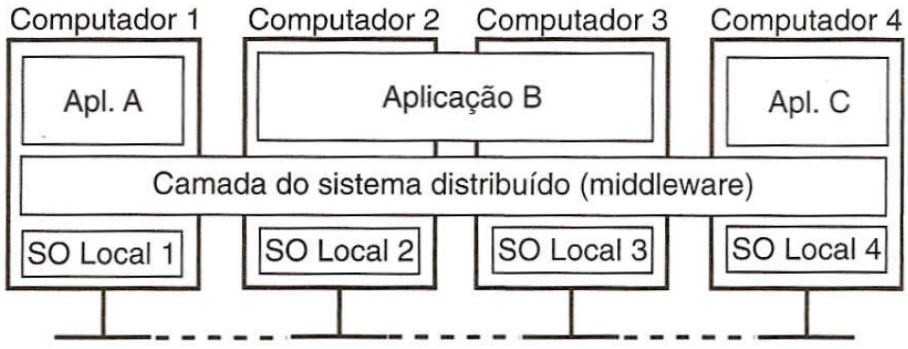


Figura 2 – Arquitetura de um sistema distribuído (STEEN; TANENBAUM, 2007)

Na Subseção 2.1.1 é introduzido o funcionamento da sincronização em sistemas distribuídos. Na Subseção 2.1.2 serão abordadas as categorias de falhas que podem ocorrer em sistemas distribuídos.

2.1.1 Sincronização

A sincronização é um dos conceitos mais importantes para o funcionamento de sistemas distribuídos. Com ela é possível garantir a execução de eventos em sistemas computacionais no momento correto. Há um conceito conhecido como sincronização intra-processos que garante que vários processos sincronizem suas ações. Isso pode ser realizado de duas formas. A primeira é por memória compartilhada e a outra por meio de mensagens. A sincronização pode ser realizada por relógios presentes no *hardware* dos nós. Há variações físicas que podem tornar relógios diferentes minimamente dessincronizados. Para mitigar esse problema foi inventado um padrão conhecido como *Universal Coordinated Time* (**UTC**). O UTC é composto por um conjunto de 40 estações de rádio ao redor do mundo que disponibiliza a informação do horário para diversos sistemas digitais sincronizados.

Existem alguns métodos conhecidos para sincronizar um sistema com o UTC. O primeiro é utilizando o algoritmo de Berkeley, onde há um conjunto de nós, um deles será o principal e o restante secundários. Os nós secundários realizam consultas no UTC. O nó principal irá consultar os nós secundários e realizar uma média do tempo obtido para obter uma maior precisão com o UTC. Outro método conhecido é utilizando o *Network Time Protocol* (**NTP**). O NTP é constituído por uma árvore de servidores. No topo da árvore há um servidor principal que transmite o UTC para os servidores aninhados (STEEN; TANENBAUM, 2017).

2.1.2 Tolerância a Falhas

Uma das características mais importantes dos sistemas distribuídos é ser capaz de identificar quando um nó está apresentando falhas e, ainda assim, continuar exercendo a função do sistema sem desencadear uma indisponibilidade ou uma perda de desempenho. A tolerância a falhas é um grande desafio, pois, está ligado diretamente a desafios como a sincronização e coordenação de elementos computacionais. Há quatro características principais para determinar a tolerância a falhas em um sistema distribuído: **disponibilidade**, **confiabilidade**, **segurança**, **manutenibilidade**.

A disponibilidade é a capacidade do sistema estar pronto para ser usado. Caso ocorra uma falha na função que estava sendo executada, quase que imediatamente deverá haver um nó novo para executar a função. A confiabilidade de um sistema se refere ao

tempo que ele permanecerá disponível sem ocorrer falhas. A confiabilidade é definida como o intervalo de tempo sem falhas.

Na citação abaixo de (STEEN; TANENBAUM, 2017) podemos ver um caso real da diferença entre disponibilidade e confiabilidade.

> Se um sistema cair em média por aparentemente um aleatório milissegundo a cada hora, ele terá uma disponibilidade de mais de 99,9999 por cento, mas ainda não é confiável. Da mesma forma, um sistema que nunca trava, mas é desligado por duas semanas específicas todo mês de agosto tem alta confiabilidade, mas apenas 96% disponibilidade.

A segurança se refere a momentos em que o sistema tem falhas que inviabilizam sua administração ou auto-gerenciamento. Porém, não ocorrerão catástrofes até que o sistema se recupere totalmente. Já a manutenibilidade se refere a facilidade de realizar manutenções no sistema em sistemas que apresentarem falhas.

Quando falamos de falhas em sistemas distribuídos o problema nem sempre pode estar associado a um nó, mas pode ser resultado de um efeito cascata de outras falhas. (COULOURIS et al., 2015) explica que podemos ter um entendimento melhor dos tipos de falhas e seus efeitos classificando-as da seguinte forma:

- Falhas por omissão: ocorrem quando um processo ou canal de envio de mensagens deixa de executar alguma tarefa.
- Falhas por omissão de processo: esse caso ocorre quando há falha em alguma tarefa e impede que as as tarefas subsequentes sejam realizadas.
- Falhas por omissão na comunicação: esse tipo de falha ocorre na situação de haver um canal de comunicação entre dois processos. A falha por omissão ocorre quando a mensagem não é enviada. Isso pode ocorrer por algumas razões (p.e., perda de mensagem por falta de espaço em *buffer*).
- Falhas arbitrárias: as falhas arbitrárias ocorrem quando uma determinada tarefa é executada porém, ela tem um retorno errado. Esse retorno errado será utilizado pelas demais tarefas, desencadeando outros resultados errôneos.
- Falhas de temporização: as falhas de temporização ocorrem quando temos um problema na sincronia das tarefas em execução ou na troca de mensagens entre processos.

A tolerância a falhas de um sistema é uma abstração para mudar o estado que apresenta erros para um estado funcional. Há duas formas principal de realizar isso. A primeira forma é usando a **recuperação para trás**. Essa forma consiste em determinado intervalo de tempo para realizar o salvamento do estado da tarefa. Quando for detectado algum problema, é carregado o último estado funcional para prosseguir a tarefa. A outra

forma de recuperação é a **recuperação direta**. Nessa forma, quando ocorre um erro na tarefa em execução, é feita a tentativa de colocar a tarefa em um novo estado funcional.

A **recuperação para trás** é mais utilizada pois para realizar a **recuperação direta** é necessário ter os problemas que podem ocorrer já mapeados para o sistema ser colocado no próximo estado funcional. Porém, a **recuperação para trás** é computacionalmente mais custosa, pois, é necessário armazenar os *checkpoints*.

2.1.3 Apache Zookeeper

O Apache Zookeeper é um arcabouço para a entrega e coordenação de entrega de mensagens para aplicações distribuídas. O Apache Zookeeper utiliza um sistema de arquivo hierárquico como dos sistemas UNIX como apresentado na Figura 3. Cada node da estrutura Zookeeper é conhecido como **znode**. A partir da raiz, há dois znodes principais. O primeiro é o znode de configurações, onde ficam centralizadas informações dos znodes. O segundo znode especial é responsável pela nomenclatura dos *workers*. O acesso às informações adicionadas ao Apache Zookeeper é realizado por bibliotecas de API, onde podem ser realizadas as operações básicas de criar, ler, editar e deletar informações. O serviço tem mecanismos rigorosos de *lock* para preservar uma informação em uso por outro recurso (HUNT et al., 2010).

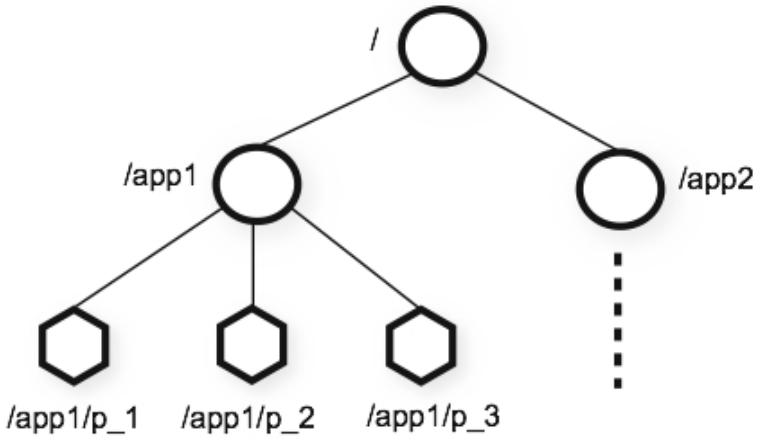


Figura 3 – Estrutura de *Namespace* em Apache Zookeeper (HUNT et al., 2010)

A Figura 4 ilustra a organização topológica em Zookeeper. Os nós em Zookeeper são denominados **znodes** e podem ser classificados entre *client* e *server*. Em geral, os clientes são nodos menos confiáveis para os quais se quer prover serviços de sincronização e tolerância a falhas. Os clientes acessam os serviços fazendo requisições para algum servidor. É necessário, portanto, pelo menos um servidor para prover os serviços para

os clientes. Para prover tolerância a falhas do serviço, opcionalmente é possível replicação de servidores. Quando a replicação está ativa, a coordenação é feita por processo de centralização em algum servidor escolhido como líder. O líder sincroniza a replicação de dados com os outros servidor. Quando ocorre a perda do líder no zookeeper, os nós mudam seu estado para o estado denominado de **LOOKING**. Então todos os nós irão iniciar uma troca de mensagens entre eles para eleger um líder novo. Quando houver um nó em comum para a escolha do líder, esse nó se tornará o novo líder. A partir desse momento, o nó escolhido passará para o estado de **LEADING** e os outros nós para o estado de **FOLLOWING**. O líder entrará para o estado **NEW_LEADER**. Quando todos os seguidores reconhecerem o novo líder, ele começará a exercer as atividades correspondentes a sua função.

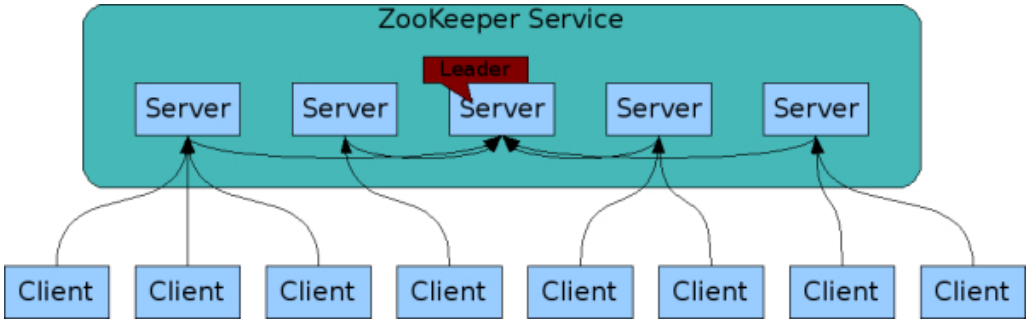


Figura 4 – Replicação em Apache Zookeeper (HUNT et al., 2010)

2.2 Information-centric networking

Information Centric-Networking (ICN) é um paradigma para a Internet do futuro como foco na distribuição eficiente e segura de conteúdo. No paradigma ICN, os dados transmitidos são denominados como *Named Data Objects* (NDOs). A abstração NDO abrange diversos conteúdos do modelo convencional como páginas na web, áudios, vídeos e documentos. Os conteúdos na rede ICN são publicados por algum **publicador** e podem ser requisitados pelos **consumidores** (JIANG et al., 2015).

ICN é implementado por uma rede de *cache*. Dessa forma, os conteúdos publicados não ficam centralizados apenas em seus publicadores. Por exemplo, quando um conteúdo é consumido por um *host* A, esse mesmo conteúdo ou fragmentos ficarão armazenados no roteador mais próximo. Supondo que um *host* B próximo ao roteador do *host* A vá consumir o mesmo conteúdo, a rede de *cache* vai receber um interesse para esse conteúdo e enviar o conteúdo para o *host* B do local que a distribuição for mais eficiente,

seja do publicador original ou do roteador próximo ao *host* A.

O paradigma ICN realiza uma verificação de origem nos conteúdos independentes. Isso garante que o conteúdo mesmo vindo da rede de *cache* mais próxima pertence ao publicador correto (AHLGREN et al., 2012). Nas subseções seguintes são apresentadas algumas das arquiteturas principais que implementam o paradigma ICN.

2.2.1 Named Data Networking

O projeto *Named Data Networking* (NDN) é um arquitetura ICN. A arquitetura NDN tem mecanismos para auxiliar no controle de concorrência, encaminhamento e armazenamento à medida que ela evolui. Não há uma estrutura obrigatória na criação de conteúdos na rede NDN. Um produtor de conteúdo pode publicar um conteúdo com a nomenclatura que desejar, porém, ele deve seguir uma determinada estrutura como um URL. O roteamento em NDN é baseado em nomes, tornando o *range* de conteúdos ilimitado. Quando há um conteúdo novo na rede, os roteadores NDN propagam pela rede essa informação. Os conteúdos são armazenados nos roteadores NDN para poderem ser reutilizados em nos encaminhamentos (ZHANG et al., 2014).

2.2.2 hICN

A arquitetura hICN é outro exemplo de implementação do paradigma ICN. Ela foi desenvolvida visando implantação prática. Por exemplo, a hICN procura instanciar o paradigma ICN como uma camada de sobreposição sobre redes IP existentes. O objetivo principal em hICN é que seja necessário o mínimo possível de mudança nos sistemas existentes. Os roteadores hICN são capazes de se comunicar com roteadores comuns e transportar objetos nomeados hICN e pacotes IP. A nomeação de dados em hICN é composta por duas partes. O nome do conteúdo que será utilizado pelos roteadores para realizar operações de encaminhamento. A segunda parte é o sufixo que será utilizado exclusivamente para o transporte dos dados. O encaminhamento de dados em hICN difere-se em ICN no aspecto que pode-se ser reutilizados estruturas de IP já existentes (CAROFIGLIO et al., 2019a).

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados à pesquisa realizada. Os trabalhos analisados demonstram diferentes experimentos realizados em diferentes ambientes. Também são apresentadas diferentes arquiteturas ICN, como NDN e CCN.

A Tabela 1 resume uma comparação entre os trabalhos relacionados e esta dissertação. Em suma, os trabalhos relacionados apresentam contribuições para a comunidade ICN. Contudo, não identificamos sistemas funcionais disponíveis para facilitar a reprodutibilidade e a tolerância a falhas. A seguir, cada um dos trabalhos relacionados é discutido em maior profundidade para melhor explicar essa lacuna. A coluna **Disponível** é referente ao fornecimento de informações ou códigos para reproduzir os mesmos resultados apresentados nos trabalhos. A coluna **Reprodutibilidade** se refere a capacidade do trabalho apresentar algum mecanismo de reprodutibilidade de experimentos. E a coluna **Tolerância a Falhas** identifica se o trabalho provê algum mecanismo para a correção de falhas. No próximo capítulo apresentamos uma proposta para preencher essa lacuna.

Tabela 1 – Comparação de trabalhos relacionados

#	Ref.	Título	Disponível	Reprodutibilidade	Tolerância a Falhas
1	(QUEREILHAC et al., 2014)	Demonstrating a unified ICN development and evaluation framework	✗	✗	✗
2	(BENEDETTO; ARUMAITHURAI; FU, 2017)	Icn personalized global-scale testbed using gts	✗	✗	✗
3	(HYUNWOO et al., 2015b)	A control, management framework for informationcentric network testbed	✗	✗	✗
4	(LAILARI et al., 2015)	Experiments with the emulated ndn testbed in onl	✓	✗	✗
5	(RAINER et al., 2016)	A low-cost ndn testbed on banana pi routers	✓	✗	✗
6	(LIU et al., 2014)	Experimental Evaluation of Consumer Mobility on Named Data Networking	✗	✗	✗
7	(MARCHAL; CHOLEZ; FESTOR, 2016)	Server-side performance evaluation of NDN	✗	✗	✗
8	(BELTRAN et al., 2022a)	Orquestrando Avaliações Experimentais com ICN-Stage	✓	✓	✓

No artigo curto (QUEREILHAC et al., 2014), os autores propõem um *framework* para implantar e emular a tecnologias ICN. Para isso, foi produzida uma extensão do *framework* NEPI. O objetivo é que o *framework* seja capaz de realizar experimentações com pouca interferência do usuário. A ferramenta proposta foi dividida em cinco módulos: descrição, implantação, monitoramento, coleta e processamento. A ferramenta pode ser executada em mais de um ambiente. No artigo ela foi avaliada no NS-3 e PlanetLab. Embora promissor, o projeto não teve continuidade.

Em (BENEDETTO; ARUMAITHURAI; FU, 2017), os autores demonstram a utilização do Geant Testbed Service (GTS) como ambiente para implantação personalizado

de topologias ICN. Foi criada uma rede utilizando o *framework* NDN com três nós, um produzindo conteúdo e outros nós consumindo. Não são apresentados muitos detalhes sobre os experimentos em si, pois o foco é a personalização de ambientes para ICN. Seria interessante se houvesse experimentos que direcionam algum parâmetro específico de ICN como *cache* ou roteamento. Infelizmente o Geant Testbed Service foi descontinuado em setembro de 2021.

No artigo (HYUNWOO et al., 2015b), os autores ressaltam a necessidade de um ambiente de *testbed* para CCN e implementam uma ferramenta para controle e gerenciamento de experimentos CCN. O *framework* utilizado para as experimentações foi o CCNx. O ambiente utilizado para os testes foram máquinas virtuais que podem ser escaladas. Contudo, os autores não apresentam detalhes sobre o gerenciamento da aplicação desenvolvida e nem resultados de experimentos com o CCNx.

No artigo (LAILARI et al., 2015) é feita uma experimentação em ambiente real do *framework* NDN. São utilizados três servidores distribuídos na América do Norte, Europa e Ásia. Cada um dos servidores tem um conjunto de máquinas virtuais. O experimento realizado consiste na publicação de conteúdo NDN e interesse desse conteúdo. É realizado o monitoramento da rede sendo coletados dados como o *delay*. A partir dos dados coletados é possível avaliar a performance da rede NDN. O trabalho não apresenta um gráfico com os resultados obtidos para facilitar o entendimento, mas disponibiliza o repositório com as configurações utilizadas. Não é citada qualquer ferramenta para *deploy*, ou seja, o ambiente foi criado de forma manual *adhoc*.

No artigo (RAINER et al., 2016) é apresentado um *testbed* construído com o microcontrolador **Banana Pi**. O objetivo é construir um ambiente para experimentação real e de baixo custo. Em cada microcontrolador é executada a ferramenta do projeto NDN conhecida como *Networking Forwarding Daemon* (NFD). O experimento realizado é para comparar estratégias de roteamento em NDN. Ao final do trabalho é feita uma comparação com o ndnSIM, que se mostrou mais escalável. Porém, o ndnSIM não consegue simular com perfeição as condições do hardware. Com a experimentação com microcontroladores é possível obter métricas como o consumo de energia. O trabalho apresenta contribuições, porém, mesmo sendo um *hardware* de baixo custo, não é uma tarefa fácil reproduzir o experimento. Em ambientes simulacionais é possível escalar o experimento em uma topologia maior e simular algumas condições de *hardware*.

No artigo (LIU et al., 2014) é realizada uma investigação sobre o desempenho do *framework* NDN com consumidores móveis. Os autores buscam simular o consumo de

clientes móveis devido ao crescente tráfego desses dispositivos na Internet. A ferramenta utilizada para a simulação foi o ndnSIM. Foi possível constatar que ainda há longo percurso no desenvolvimento de NDN. Na simulação não foi utilizado o recurso de *cache* em NDN. Isso gerou outro problema que foi a reemissão de interesses. Poderiam ter sido realizados testes de falha nos roteadores NDN para validar o comportamento dos dispositivos móveis.

No artigo (MARCHAL; CHOLEZ; FESTOR, 2016), os autores buscam demonstrar uma avaliação de desempenho no lado do servidor ao utilizar o *framework* NDN. Para atingir esse objetivo, foi desenvolvida a ferramenta NDNperf. Com o NDNperf é possível monitorar informações dos pacotes NDN como taxa de transferência, latência e tempo de processamento. O objetivo do trabalho é demonstrar como os mecanismos de segurança dos pacotes pode prejudicar a desempenho da geração de conteúdo do lado dos servidores. A experimentação foi realizada em dois computadores físicos. Isso dificulta a reprodutibilidade do experimento.

A maioria dos trabalhos relacionados foram descontinuados ou não fornecem materiais para reproduzir seus resultados. Foi realizada uma busca por mais informações pela descontinuidade desses trabalhos. Porém, não foi encontrada uma justificativa específica para o fim dos trabalhos.

4 ICN-STAGE

O sistema proposto nesta dissertação é voltado para um *usuário* que desenha experimentos ICN de longa duração e grão grosso de tempo e os executa através de algum ambiente de testes composto por nós distribuídos e suscetíveis a falhas. O objetivo geral do ICN-STAGE é facilitar a reprodução de experimentos tolerantes a falhas. O ICN-STAGE é capaz detectar e corrigir falhas (intermitentes ou permanentes) de nós do ambiente de testes (*e.g.*, máquina física executando um publicador NDN, ou *Name Data Networking*) subjacente para evitar que elas gerem erros de experimentos (*e.g.*, publicador ausente) e, consequentemente, defeitos de análise de resultados (*e.g.*, taxa média de respostas dos publicadores). Assumimos que os relógios são fracamente (ou podem facilmente ser) sincronizados através de protocolos como o PTP (padrão IEEE-1588).

4.1 Arquitetura

A Figura 5 apresenta a arquitetura do ICN-STAGE. Ela possui dois componentes principais: *diretor* e *ator*. O diretor tem uma *interface de usuário*, que permite o gerenciamento do próprio diretor e também peças e atores. A *interface de comunicação do diretor* gerencia canais de comunicação com os atores para enviar arquivos relacionados a um papel da peça, por exemplo.

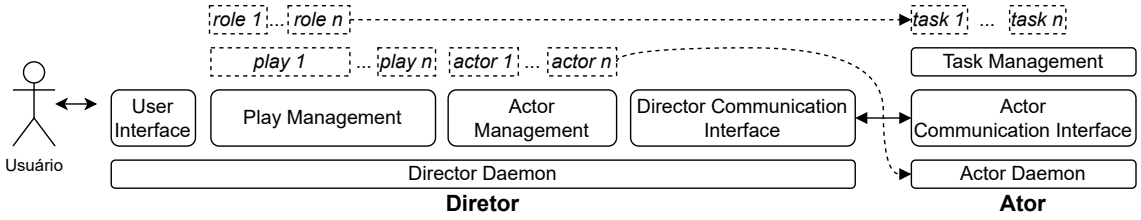


Figura 5 – Arquitetura do ICN-STAGE

O *Play Management* orquestra (*i.e.*, adiciona, remove e monitora) as peças. Uma peça pode conter vários papéis (por exemplo, publicador e solicitante de conteúdo), e cada papel pode ser instanciado diversas vezes (por exemplo, um publicador e dois solicitantes), que chamamos de *tarefas*. O *Gerenciamento de atores* atribui tarefas aos atores. Além disso, ele monitora passivamente pulsações (*i.e.*, *heartbeats*) dos atores, que são substituídos em caso de falha, e impedidos de retornarem à peça em caso de recuperação (de uma falha intermitente). O diretor pode ser executado como um processo único ou como um conjunto de réplicas para prover tolerância a falhas de diretor.

O Ator possui três subcomponentes principais. A *Communication Interface* permite a comunicação com o diretor para enviar pulsações periódicas e resultados, por exemplo. O *Task Management* controla tarefas (ou seja, instâncias de papéis), que podem ser uma nova ou uma recuperação de uma tarefa anterior com falha. Por fim, *Actor Daemon* executa o processo principal do ator.

4.2 Implementação

A implementação do ICN-STAGE está disponível publicamente no Github (BELTRAN et al., 2022a). O ICN-STAGE foi desenvolvido em Python3[1] (ROSSUM; DRAKE, 2009). ICN-STAGE atualmente suporta a arquitetura NDN (JACOBSON et al., 2012) e hICN (CAROFIGLIO et al., 2019b). O ICN-STAGE é compatível com dois palcos: (*i*) Minikube para execução em ambiente local, e (*ii*) FIBRE para execução em ambiente distribuído. O referido repositório também contém informações necessárias para reprodução dos experimentos apresentados na próxima seção de maneira simplificada[2].

Originalmente o projeto foi desenvolvido para que os nós (diretores e atores) fossem executados em máquinas virtuais localmente ou em serviços como *Planetlab*. Com o objetivo de melhorar a escalabilidade e aumentar o grau de automatização do ICN-STAGE, o projeto foi convertido para contêiner. O ICN-STAGE orquestra o *deployment* dos atores e diretores utilizando a ferramenta *Kubernetes*. Na Figura 6 é apresentada arquitetura do ICN-STAGE. No momento do *deploy* do palco é possível definir um valor *N* de diretores e um valor *M* de atores.

Na Figura 7 é apresentado o fluxo de estados de um ator. Supondo que haja um cenário onde o diretor já está em funcionamento e é inciado o estado de **adicionar atores**. Nesse estado é realizada uma verificação na lista de *containers* que serão atores e criada uma tarefa para inicializá-los. Após isso, é inicializada a tarefa de **adicionar papel**. Nessa etapa é realizada uma eleição de um ator para executar um papel designado pelo diretor. Por fim, é inicializada a tarefa de **executar papel**. Nessa última etapa é executado o papel definido pelo diretor na eleição. Todas as etapas citadas, possuem um grau de tolerância a falhas. Se houver um problema na etapa de **adicionar atores** ou **adicionar papel**, a tarefa de **recuperar ator** é inicializada. Se ocorrer um problema na etapa de **executar papel** é iniciada a tarefa de **eleger ator** para que um novo ator seja eleito para continuar o papel.

[1] <https://www.python.org/>
[2] Configurar o ambiente e executar: python3 play_ndn.py

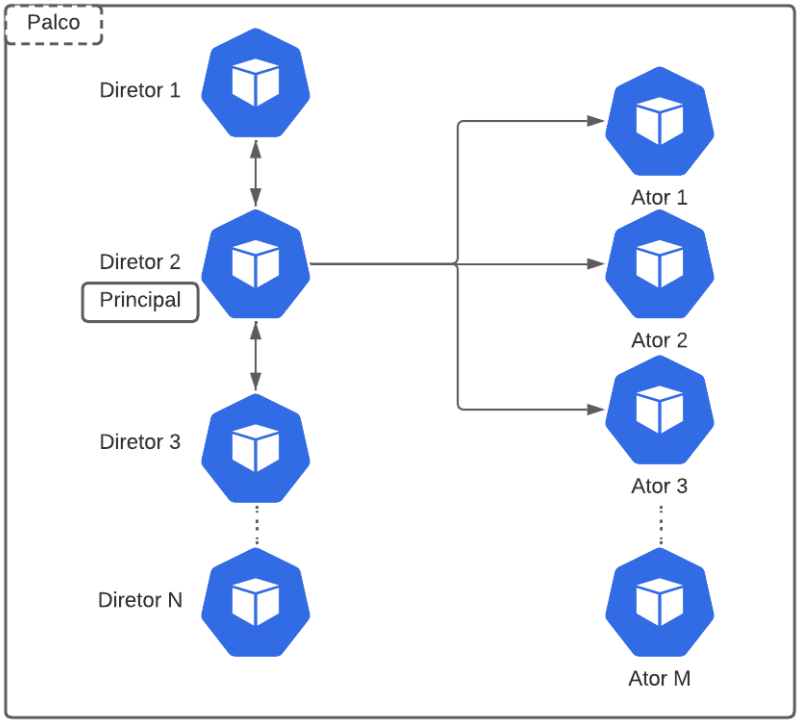


Figura 6 – Arquitetura do ICN-STAGE como *container*

Para a execução de um experimento, o processo é semelhante ao de adicionar atores. É criada uma tarefa para executar determinado experimento. Os atores já possuem os experimentos, então é enviado um arquivo com as informações para a execução do experimento.

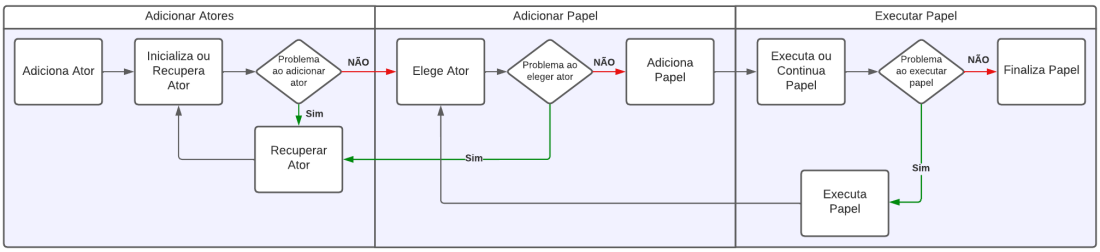


Figura 7 – Estados do diretor do ICN-STAGE

O ICN-STAGE utiliza o ZooKeeper[3] como *middleware* de comunicação distribuída. As interfaces de comunicação do Diretor e do Ator utilizam o ZooKeeper para persistir o modelo do sistema (peças, atores, pulsação, etc.). Além disso, o mecanismo de tolerância a falhas do Diretor é baseado em réplicas com escolha de líder. Em suma, o algoritmo de eleição do ICN-STAGE segue a decisão da eleição do ZooKeeper.

[3] <https://www.zookeeper.apache.org>

5 AVALIAÇÃO

Neste capítulo são apresentados os resultados obtidos durante os experimentos. Na Seção 5.1 são apresentadas as especificações dos experimentos realizados e como foram construídos. Foram definidos 5 cenários específicos de experimentos. Na Seção 5.2 são apresentados os resultados obtidos nos 5 cenários criados.

5.1 Metodologia

Foi executada uma peça simples no ICN-STAGE para demonstrar suas capacidades. A peça é composta por dois papéis: um publicador de conteúdo e um solicitante de conteúdo. Cada papel representa a execução do respectivo componente do NDN Traffic Generator[1], que permite gerar e atender fluxos de requisições NDN. Cada instância é conectada a um Named Data Networking Forwarding Daemon[2] (*i.e.*, uma espécie de roteador da arquitetura NDN), que também é responsabilidade do papel. Cada unidade de cache foi configurada com capacidade máxima de 1 conteúdo. Desse modo, a maioria dos interesses enviados pelo solicitante devem alcançar o publicador, pois cada solicitação tende a resultar em *cache miss* considerando a relação entre tamanho de cache e tamanho do catálogo. Definimos um catálogo de 100 conteúdos de 8KiB cada. O solicitante gera a demanda de um conteúdo, que é escolhido seguindo probabilidade uniforme, a cada 100 milissegundos durante 540 segundos. Para aferir a execução, nós monitoramos as solicitações recebidas pelo publicador (e conferimos com as requisições enviadas pelos solicitantes). Para demonstrar a flexibilidade do ambiente, executamos uma peça semelhante a anterior, porém baseada em uma arquitetura diferente. Para esse propósito escolhemos a arquitetura hICN[3]. A implementação atual do hICN disponibiliza algumas ferramentas para testar a infraestrutura. São elas o hicn-ping-server utilizado para publicar conteúdos e o hicn-ping-client que é utilizado para requisitar o conteúdo publicado pelo publicador.

Ambas as peças foram executadas em 5 cenários distintos de falha e redundância de ator (solicitante) e diretor conforme a Tabela 2. Para prover redundância de atores (cenários 3-5), escalamos um terceiro ator. Para prover redundância de diretores (cenários

[1] <https://github.com/named-data/ndn-traffic-generator>
[2] <https://github.com/named-data/NFD>
[3] <https://github.com/FDio/hicn>

4 e 5), instanciamos 3 diretores para haver quórum durante a eleição de líder. Para gerar as falhas de atores e diretores (cenários 2-5), utilizamos uma função do Kubernetes de remoção de nó.

Tabela 2 – Cenários da avaliação.

#	Qtd. Atores	Qtd. Diretores	Falha Ator?	Falha Diretor?	Descrição
C1	2	1	✗	✗	sem falha de ator e de diretor
C2	2	1	✓	✗	com falha e sem redundância de ator e sem falha de diretor
C3	3	1	✓	✗	com falha e redundância de ator e sem falha de diretor
C4	3	3	✓	✓	com falha e redundância de ator e com falha e sem redundância de diretor
C5	3	3	✓	✓	com falha e redundância de ator e com falha e redundância de diretor

As implementações dos *frameworks* NDN e hICN foram construídas em imagens de contêineres separadas. Cada imagem é destinada a uma determinada peça e estão hospedadas no Docker Hub[4]

5.2 Resultados

A Figura 8 apresenta as requisições recebidas pelo publicador durante cada um dos cinco cenários no FIBRE. Vale ressaltar que resultados similares foram obtidos em ambiente local.

O primeiro cenário (C1) não contém falha e serve como base de comparação. Observe-se uma média de 10 requisições por segundo durante 540 segundos, conforme especificado. O segundo cenário indica que a falha do ator encerra requisições e que, portanto, o mecanismo de injeção de falha é efetivo. O terceiro cenário mostra que o ICN-STAGE é capaz de tolerar falhas de ator. O quarto cenário sugere que a falha do diretor, a exemplo do segundo caso, também é efetivo. Por fim, o quinto cenário demonstra que o ICN-STAGE é capaz de tolerar falhas de atores e diretores combinadas.

Na Figura 9 são apresentados os resultados da execução no Minikube. Os cenários executados são os mesmos da execução em FIBRE. Os resultados em ambos ambientes são semelhantes. Isso demonstra que independente do palco o ICN-STAGE pode auxiliar na reprodutibilidade de peças.

Destacamos duas observações sobre os resultados gerais dos cenários com NDN. Primeiro, pode-se observar que os inícios e términos das execuções da peça NDN em diferentes cenários coincidem entre si, assim como as falhas e a execução da etapa restante, ilustrando a capacidade de reprodutibilidade do ICN-STAGE. Segundo, o tempo

[4]<https://hub.docker.com/>

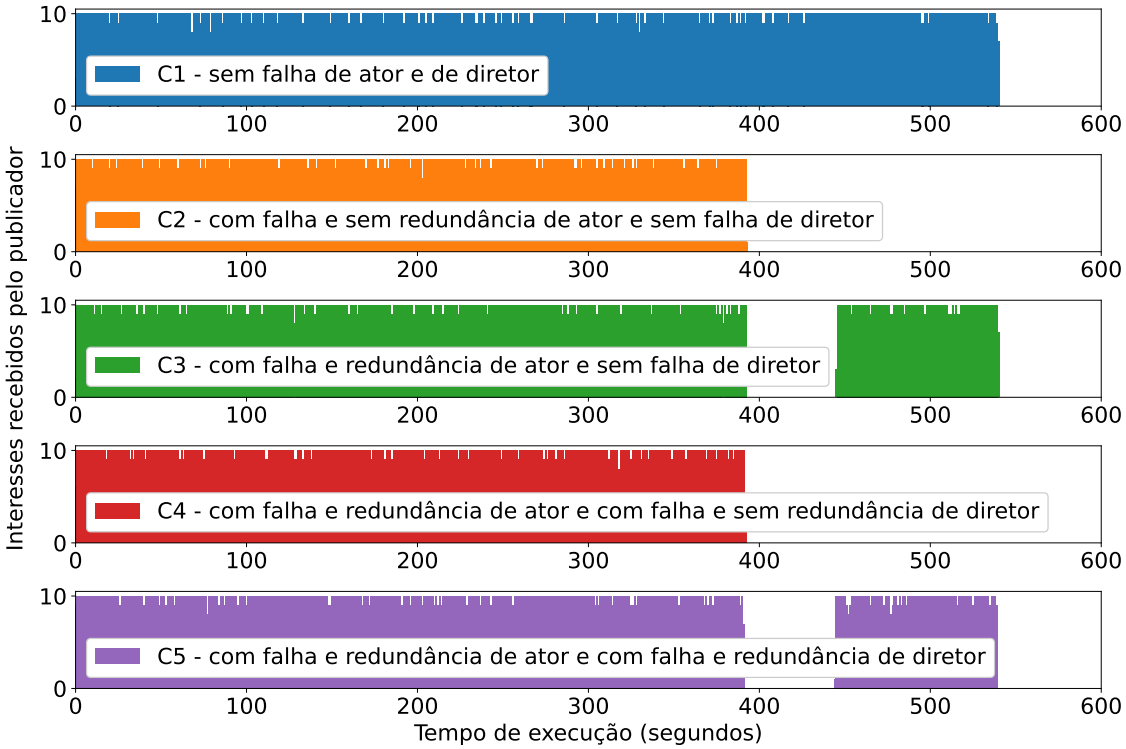


Figura 8 – Resultados da peça NDN em 5 cenários distintos executados no FIBRE (BELTRAN et al., 2022a)

necessário para detecção e correção de falha de ator foi, consistentemente, 53 segundos aproximadamente. Observamos que esse valor pode ser considerado baixo para experimentos de longa duração (ordem de horas ou dias), nos quais tolerância a falha é mais relevante, e que, caso necessário, esse tempo pode ser minimizado com futuras otimizações de código.

A Figura 10 apresenta a execução da peça hICN nos cinco cenários definidos na Tabela 2. Essa peça foi executada no palco Minikube. Cada um dos cenários foi executado com aproximadamente 800 segundos.

A peça hICN teve um comportamento similar a peça NDN. Foram introduzidos os mesmos cenários em uma peça de duração maior. O ICN-STAGE foi capaz de identificar as falhas e dar prosseguimento a peça com exito. No caso da peça NDN programamos para que a falha ocorresse aproximadamente aos 400 segundos. Já no caso da peça hICN o tempo da experimentação foi de aproximadamente 800 segundos. A falha foi programada para ocorrer aos 600 segundos. Levou cerca de 1 minuto e 45 segundo para o prosseguimento da peça hICN. A diferença de tempo entre o prosseguimento da peça NDN e hICN não está ligada diretamente ao ICN-STAGE, mas sim ao tempo para o inicialização das aplicações de cada de cada implementação ICN.

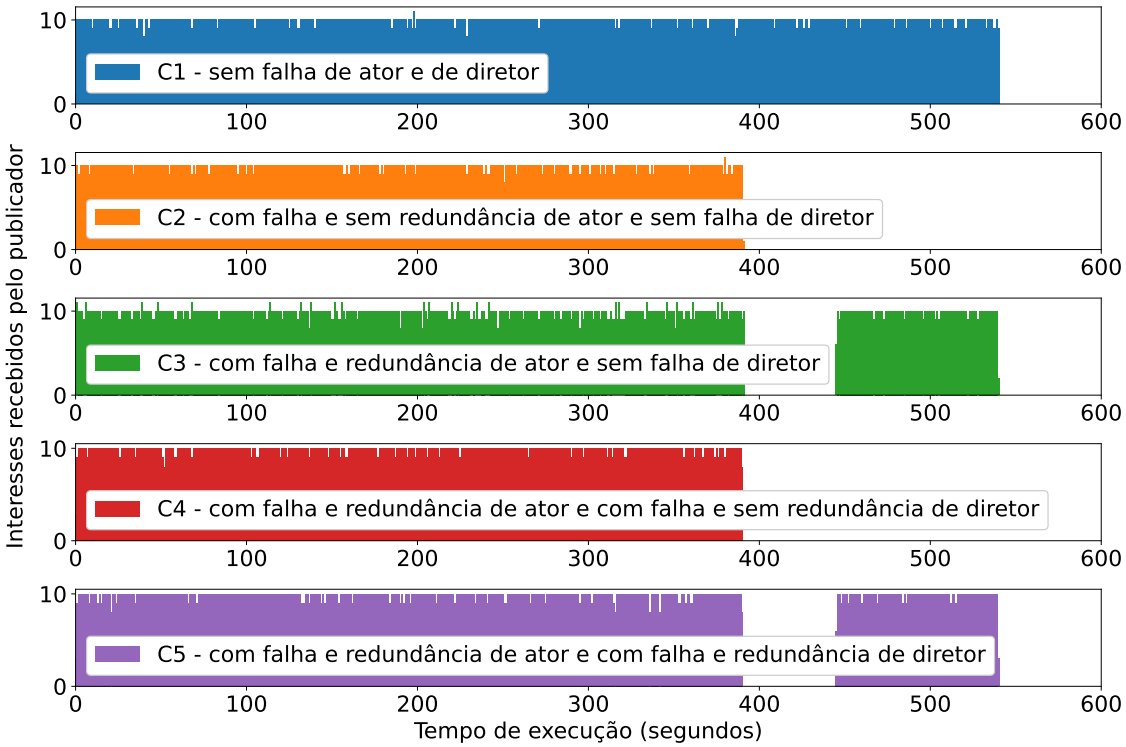


Figura 9 – Resultados da peça NDN em 5 cenários distintos executados no Minikube (BELTRAN et al., 2022a)

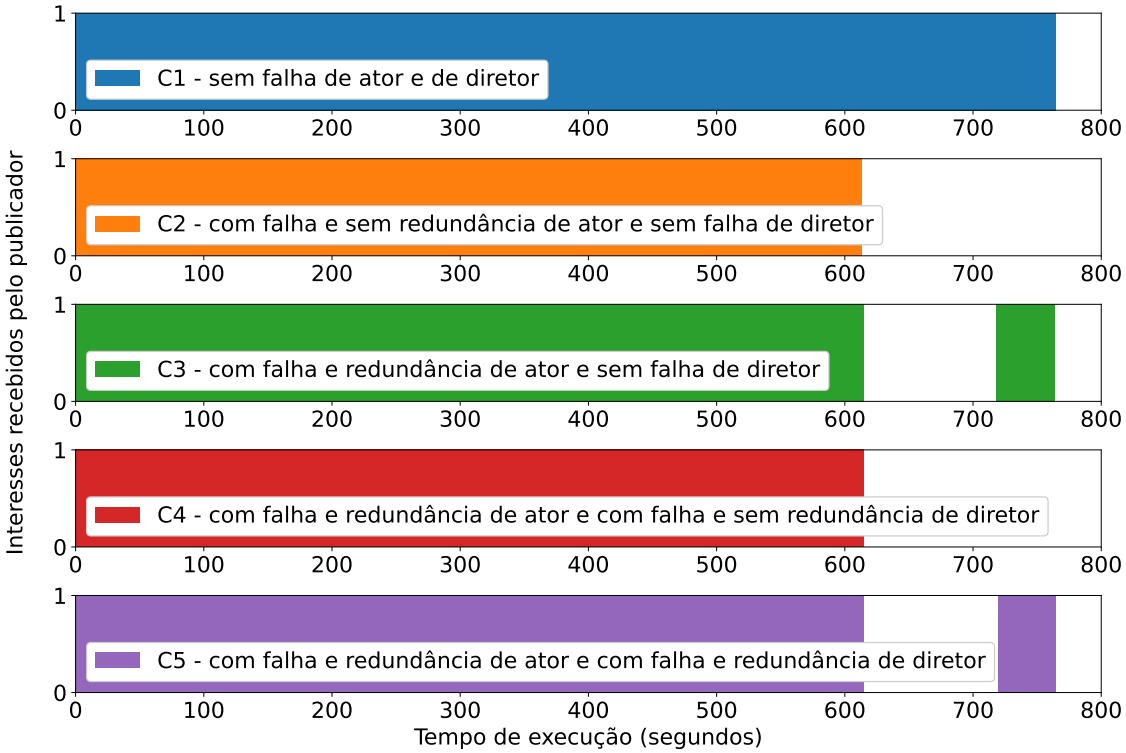


Figura 10 – Resultados da peça HICN em 5 cenários distintos executados no Minikube (BELTRAN et al., 2022b)

6 CONCLUSÃO

Nesta dissertação apresentamos o ICN-STAGE– um arcabouço para execução reprodutível de experimentos ICN tolerante a falhas. Demonstramos as funcionalidades do ICN-STAGE através de experimentos realizados em ambientes local e distribuído no FIBRE. Para a experimentação foi utilizado a arquitetura NDN e hICN. Os experimentos desenvolvidos foram uma simulação da publicação de conteúdos em um ator e outro ator enviando interesses para esses conteúdos.

Acreditamos que ICN-STAGE possa contribuir para condução de novos experimentos, bem como na reprodução de experimentos publicados anteriormente na literatura. Como trabalhos futuros, pretendemos criar peças que reproduzam experimentos publicados na literatura. Dessa forma, acreditamos que ICN-STAGE possa contribuir para a comparação cruzada de tecnologias e reprodutibilidade de resultados. Pretendemos aplicar ICN-STAGE na condução de atividades de pesquisa, desenvolvimento e inovação no âmbito de ICN. Por exemplo, vislumbramos oportunidades para implementar ICN para qualificar o processo de disseminação de conteúdo educacional na infraestrutura da RNP.

REFERÊNCIAS

AHLGREN, B. et al. A survey of information-centric networking. **IEEE Communications Magazine**, v. 50, n. 7, p. 26–36, 2012.

BAJPAI, V. et al. The Dagstuhl Beginners Guide to Reproducibility for Experimental Networking Research. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 49, n. 1, p. 24–30, feb. 2019. ISSN 0146-4833. Available from Internet: <https://doi.org/10.1145/3314212.3314217>.

BELTRAN, R. D. et al. **ICN-Stage repo**. 2022. Https://github.com/RafaelDBeltran/ICN-Stage$_W PEIF - 2022. Available from Internet : <>$.

BELTRAN, R. D. et al. **ICN-Stage repo**. 2022. Https://github.com/RafaelDBeltran/ICN-Stage$_P lay - Hicn. Available from Internet : <>$.

BENEDETTO, J. D.; ARUMAITHURAI, M.; FU, X. ICN personalized global-scale testbed using GTS. In: . [S.l.: s.n.], 2017. p. 208–209. ISBN 978-1-4503-5122-5.

CAROFIGLIO, G. et al. Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture. In: **Proceedings of the 6th ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2019. (ICN '19), p. 55–66. ISBN 9781450369701. Available from Internet: <https://doi.org/10.1145/3357150.3357394>.

CAROFIGLIO, G. et al. Enabling ICN in the Internet Protocol: Analysis and Evaluation of the Hybrid-ICN Architecture. In: **Proceedings of the 6th ACM Conference on ICN**. ACM, 2019. (ICN '19). ISBN 9781450369701. Available from Internet: <https://doi.org/10.1145/3357150.3357394>.

COULOURIS, G. et al. **Sistemas Distribuídos - 5ed: Conceitos e Projeto**. [S.l.]: Bookman Editora, 2015. ISBN 9788582600542.

De Benedetto, J.; ARUMAITHURAI, M.; FU, X. ICN Personalized Global-Scale Testbed Using GTS. In: **Proceedings of the 4th ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2017. (ICN '17), p. 208–209. ISBN 9781450351225. Available from Internet: <https://doi.org/10.1145/3125719.3132095>.

GARRETT, T.; BONA, L. C.; JR, E. P. D. Improving the Performance and Reproducibility of Experiments on Large-scale Testbeds with k-Cores. **Computer Communications**, Elsevier, 2017.

HUNT, P. et al. ZooKeeper: Wait-Free Coordination for Internet-Scale Systems. In: **Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference**. USA: USENIX Association, 2010. (USENIXATC'10), p. 11.

HYUNWOO, L. et al. ICN-OMF: A control, management framework for Information-Centric Network testbed. In: **2015 International Conference on Information Networking (ICOIN)**. [S.l.: s.n.], 2015. p. 416–417.

HYUNWOO, L. et al. ICN-OMF: A control, management framework for Information-Centric Network testbed. In: **2015 International Conference on Information Networking (ICOIN)**. [S.l.: s.n.], 2015. p. 416–417.

IMBERT, M. et al. Using the EXECO toolkit to perform automatic and reproducible cloud experiments. In: IEEE. **Int'l Conference on Cloud Computing Technology and Science (CloudCom 2013)**. [S.l.], 2013. v. 2, p. 158–163.

JACOBSON, V. et al. Networking Named Content. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 55, n. 1, p. 117–124, jan. 2012. ISSN 0001-0782. Available from Internet: <https://doi.org/10.1145/2063176.2063204>.

JIANG, X. et al. A survey on Information-centric Networking: Rationales, designs and debates. **China Communications**, v. 12, n. 7, p. 1–12, 2015.

JUNIOR, N. A. A.; CORDEIRO, W. L. da C.; GASPARY, L. P. Permitindo Maior Reprodutibilidade de Experimentos em Ambientes Distribuídos com Nodos de Baixa Confiabilidade. In: **Anais do XXXVI SBRC**. Porto Alegre, RS, Brasil: SBC, 2018. ISSN 2177-9384. Available from Internet: <https://sol.sbc.org.br/index.php/sbrc/article/view/2424>.

LAILARI, Z. et al. Experiments with the Emulated NDN Testbed in ONL. In: **Proceedings of the 2nd ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2015. (ACM-ICN '15), p. 219–220. ISBN 9781450338554. Available from Internet: <https://doi.org/10.1145/2810156.2812616>.

Lim, H. et al. NDN Construction for Big Science: Lessons Learned from Establishing a Testbed. **IEEE Network**, v. 32, n. 6, p. 124–136, 2018.

LIU, Z. et al. Experimental evaluation of consumer mobility on named data networking. In: **2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)**. [S.l.: s.n.], 2014. p. 472–176.

MARCHAL, X.; CHOLEZ, T.; FESTOR, O. Server-Side Performance Evaluation of

NDN. In: . New York, NY, USA: Association for Computing Machinery, 2016. (ACM-ICN '16), p. 148–153. ISBN 9781450344678. Available from Internet: <https://doi.org/10.1145/2984356.2984364>.

NICHOLS, K. Lessons Learned Building a Secure Network Measurement Framework Using Basic NDN. In: **Proceedings of the 6th ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2019. (ICN '19), p. 112–122. ISBN 9781450369701. Available from Internet: <https://doi.org/10.1145/3357150.3357397>.

PAIM, K. et al. Usando Redes Neurais para Reconstruir Traços de Sessões de Usuários de Sistemas de Larga Escala. In: **Anais do XXXIX SBRC**. Porto Alegre, RS, Brasil: SBC, 2021. p. 826–839. ISSN 2177-9384. Available from Internet: <https://sol.sbc.org.br/index.php/sbrc/article/view/16766>.

PAIM, K. O. et al. Fix me if you can: Using neural networks to regenerate networked systemsx2019; monitoring traces. In: **NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium**. [S.l.: s.n.], 2022. p. 1–9.

QUEREILHAC, A. et al. Demonstrating a Unified ICN Development and Evaluation Framework. In: **Proceedings of the 1st ACM Conference on Information-Centric Networking**. New York, NY, USA: Association for Computing Machinery, 2014. (ACM-ICN '14), p. 195–196. ISBN 9781450332064. Available from Internet: <https://doi.org/10.1145/2660129.2660132>.

RAHMAN, A. et al. **Deployment Considerations for Information-Centric Networking (ICN)**. RFC Editor, 2020. RFC 8763. (Request for Comments, 8763). Available from Internet: <https://rfc-editor.org/rfc/rfc8763.txt>.

RAINER, B. et al. A low-cost NDN testbed on banana pi routers. **IEEE Communications Magazine**, v. 54, n. 9, p. 105–111, 2016.

ROSSUM, G. V.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

RUIZ, C. C. et al. Managing large scale experiments in distributed testbeds. In: **Int'l Association of Science and Technology for Development (IASTED)**. [S.l.: s.n.], 2013. p. 628–636.

SANTOS, M.; FERNANDES, S.; KAMIENSKI, C. Conducting network research in large-scale platforms: Avoiding Pitfalls in PlanetLab. In: IEEE. **Advanced Information**

Networking and Applications (AINA). [S.l.], 2014. p. 525–532.

STEEN, M. V.; TANENBAUM, A. S. **Distributed systems**. [S.l.]: Maarten van Steen Leiden, The Netherlands, 2007.

STEEN, M. V.; TANENBAUM, A. S. **Distributed systems**. [S.l.]: Maarten van Steen Leiden, The Netherlands, 2017.

ZHANG, L. et al. Named Data Networking. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 3, p. 66–73, jul 2014. ISSN 0146-4833. Available from Internet: <https://doi.org/10.1145/2656877.2656887>.

ANEXO A — ARTIGO APRESENTADO NO XIII WORKSHOP DE PESQUISA EXPERIMENTAL DA INTERNET DO FUTURO (WPEIF 2022) NO ÂMBITO DO XL SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS (SBRC 2022)

- Título: Orquestrando Avaliações Experimentais com ICN-Stage
- Autores: Rafael Duarte Beltran, Kayuã Oleques Paim, Diego Kreutz, Rodrigo Brandão Mansilha, Weverton Cordeiro
- Site do evento: <http://sbrc2022.sbc.org.br/>
- Link da publicação: <https://doi.org/10.5753/wpeif.2022.223485>

ANEXO B — REPOSITÓRIO PARA REPRODUZIR OS RESULTADOS PUBLICADOS NO WPEIF'22

- Primeiro clonar o repositório do ICN-Stage, disponível em <https://github.com/RafaelDBeltran/ICN-Stage_WPEIF-2022>.
- É necessário instalar os *softwares* listados abaixo:
  - Minikube[1]
  - kubectl[2]
  - Python3[3]
- O próximo passo é rodar o comando para instalar as dependências:

```
pip install -r requirements.txt
```

- Para executar a peça é necessário ativar o Minikube com o comando abaixo.

```
minikube start
```

- Em seguida rodar os *script* usando o comando:

```
python3 play_ndn.py
```

[1] <https://minikube.sigs.k8s.io>
[2] <https://kubernetes.io/docs/tasks/tools/>
[3] <https://www.python.org>