Instituto Superior de Engenharia do Porto

ESTUDO E ANÁLISE DAS REDES CENTRADAS EM CONTEÚDOS

Jacinto António de Andrade Barbosa

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientador: Professor Nuno Pereira

Júri:

Presidente:

José António Reis Tavares, Doutor, ISEP

Vogais:

Maria João Monteiro Ferreira Viamonte, Doutor, ISEP

Nuno Alexandre Magalhães Pereira, Doutor, ISEP

Porto, outubro 2014

Dedicatória

Este trabalho é dedicado ao meu filho Tiago e à minha esposa Carla, por todos os momentos em que estivemos privados de um convívio familiar, em detrimento do tempo e dedicação despendidos com este meu objectivo académico.

Resumo

A Internet, conforme a conhecemos, foi projetada com base na pilha de protocolos TCP/IP, que foi desenvolvida nos anos 60 e 70 utilizando um paradigma centrado nos endereços individuais de cada máquina (denominado *host-centric*). Este paradigma foi extremamente bem-sucedido em interligar máquinas através de encaminhamento baseado no endereço IP.

Estudos recentes demonstram que, parte significativa do tráfego atual da Internet centra-se na transferência de conteúdos, em vez das tradicionais aplicações de rede, conforme foi originalmente concebido. Surgiram então novos modelos de comunicação, entre eles, protocolos de rede ponto-a-ponto, onde cada máquina da rede pode efetuar distribuição de conteúdo (denominadas de redes *peer-to-peer*), para melhorar a distribuição e a troca de conteúdos na Internet. Por conseguinte, nos últimos anos o paradigma *host-centric* começou a ser posto em causa e apareceu uma nova abordagem de Redes Centradas na Informação (ICN - *information-centric networking)*. Tendo em conta que a Internet, hoje em dia, basicamente é uma rede de transferência de conteúdos e informações, porque não centrar a sua evolução neste sentido, ao invés de comunicações *host-to-host*?

O paradigma de Rede Centrada no Conteúdo (CCN - *Content Centric Networking*) simplifica a solução de determinados problemas de segurança relacionados com a arquitetura TCP/IP e é uma das principais propostas da nova abordagem de Redes Centradas na Informação. Um dos principais problemas do modelo TCP/IP é a proteção do conteúdo. Atualmente, para garantirmos a autenticidade e a integridade dos dados partilhados na rede, é necessário garantir a segurança do repositório e do caminho que os dados devem percorrer até ao seu destino final. No entanto, a contínua ineficácia perante os ataques de negação de serviço praticados na Internet, sugere a necessidade de que seja a própria infraestrutura da rede a fornecer mecanismos para os mitigar. Um dos principais pilares do paradigma de comunicação da CCN é focalizar-se no próprio conteúdo e não na sua localização física. Desde o seu aparecimento em 2009 e como consequência da evolução e adaptação a sua designação mudou atualmente para Redes de Conteúdos com Nome (NNC – Named Network Content).

Nesta dissertação, efetuaremos um estudo de uma visão geral da arquitetura CCN, apresentando as suas principais características, quais os componentes que a compõem e como os seus mecanismos mitigam os tradicionais problemas de comunicação e de segurança. Serão efetuadas experiências com o CCNx, que é um protótipo composto por um conjunto de funcionalidades e ferramentas, que possibilitam a implementação deste paradigma. O objetivo é analisar criticamente algumas das propostas existentes, determinar oportunidades, desafios e perspectivas para investigação futura.

Palavras-chave: Content Centric, Information Centric, Named Content, CCN, CCNx, ICN, NNC

Abstract

The Internet, as we know it, was designed based on the stack TCP/IP protocol, which was developed in the 60s and 70s using a paradigm centered on the individual addresses of each machine (called host-centric). This paradigm has been extremely successful in interconnect machines via routing based on IP address.

Recent studies show that a significant proportion of the current Internet traffic focuses on the content transfer, instead of the traditional network applications, as originally conceived. Then came new communication models, among them point-to-point network protocols, where each machine on the network can perform content distribution (called peer-to-peer networks), to improve the distribution and exchange content in Internet. Therefore, in recent years the host-centric paradigm began to be challenged and there was a new approach in Information-Centric Networking (ICN). Considering that the Internet today, is basically a network of content transfer and information, why not focus its evolution in this sense, rather than communications host-to-host?

The paradigm of the Content Centric Networking (CCN) simplifies the solution of certain security issues related to TCP/IP architecture and is one of the main proposals of the new approach in Information-Centric Networks. One of the main problems of the TCP/IP model is the protection of content. Currently, to guarantee the authenticity and integrity of data shared on the network, it is necessary to ensure the safety of the repository and the way that data must travel to their final destination. However, the continued ineffectiveness against the denial of service performed on the Internet, suggests the need to be their own network infrastructure to provide mechanisms to mitigate it. One of the main pillars of the CCN communication paradigm is to focus on the content itself and not on its physical location. Since its onset in 2009 and as a result of its evolution and adaptation, its name has now changed to Named Network Content (NNC).

In this thesis, we make a study of an overview of the CCN architecture, featuring their main characteristics, which components comprise and how its mechanisms mitigate the traditional problems of communication and security. Experiments will be performed with the CCNx, that it is a prototype, composed of a set of features and tools, which enable the implementation of this paradigm. The aim is to critically analyze some of the existing proposals, determine opportunities, challenges and prospects for future research.

***Keywords**: Content Centric, Information Centric, Named Content, CCN, CCNx, ICN, NNC*

Agradecimentos

Ao meu filho Tiago, por toda a compreensão, que teve, durante estes últimos anos, em que esteve impedido de partilhar comigo as suas brincadeiras e aventuras, da forma que mais gostaríamos de o fazer e dos momentos em que sentiu falta do carinho paternal, em alturas que eventualmente mais precisou dele e eu estava ausente, sem nunca deixar de manifestar o seu apoio e dando me forças para prosseguir, nos pequenos bocadinhos que partilhamos.

À minha esposa Carla por todo o apoio incondicional e compreensão, que sempre me motivou, nesta minha vontade de alcançar este grau académico. Sem o seu apoio a minha motivação e força de vontade ter-se-iam perdido nos difíceis momentos profissionais que passei nos últimos tempos.

Aos meus Sogros pelo constante interesse em saber se os meus estudos estavam a correr bem.

A todos os meus amigos e colegas que de uma forma direta ou indireta contribuíram para que este meu objectivo académico se tornasse realidade.

Aos professores do ISEP, pela paciência, pela motivação e profissionalismo com que sempre me transmitiram os seus ensinamentos e me acompanharam na realização e orientação dos trabalhos práticos.

Ao Professor Nuno Pereira pela sua disponibilidade, profissionalismo e optimismo com que sempre me orientou.

A todos um grande bem-haja!

Índice

Lista de Figuras

Acrónimos e Símbolos

Lista de Acrónimos

Ack	Acknowledgement (data networks)
API	Application Programming Interface
AS	Autonomous System (Internet)
C	Linguagem de programação compilada e procedimental
CCN	Content Centric Network
CCNx	Content Centric Networking Project Software (Tools)
CRC32C	Cyclic Redundancy Check 32-Bit
CS	Content Store
DDoS	Distributed Denial of Service
DHT	Distributed hash table
DNS	Domain Name System
DDoS	Distributed Denial of Service
DoS	Denial of Service
DSCP	Differentiated Services Code Point
EIGRP	Enhanced Interior Gateway Routing Protocol
FIB	Forwarding Interest Base
FIFO	First In First Out
GUI	Graphic User Interface
HTTP	Hypertext Transmition Protocol
HTTPS	Hypertext Transmition Protocol Secure
ICN	Information Centric Networking
IGP	Interior Gateway Protocol
IP	Internet Protocol

ISEP	Instituto Superior de Engenharia do Porto
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
JAVA	Linguagem de programação interpretada e orientada a objetos
LFU	Least Frequently Used
LRU	Least Recently Used
LSA	Link-State Advertisement
MAC	Message Authentication Code
MAC	Media Access Control
MIC	Message Integrity Check
NACK	Negative Acknowledgement
NDN	Named Data Networking
NNC	Networking Named Content
OSPF	Open Shortest Path First
PARC	Palo Alto Research Center
PGP	Pretty Good Privacy
PIT	Pending Interest Table
PKI	Public Key Infrastructure
QoS	Quality of Service
RIP	Routing Information Protocol
RSA	One of the first practicable public-key cryptosystems
RTP	Real-time Transport Protocol
RTT	Round-Trip Time
SDSI	Simple Distributed Security Infrastructure
SHA-256	SHA-2 é uma família de funções de hash criptográficas
SIP	Session Initiation Protocol

SNMP	Simple Network Management Protocol
SPKI	Simple Public-Key Infrastructure
TCP	Transmission Control Protocol
TLV	Type-Length-Value
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoCCN	Voice over CCN
VOIP	Voice Over IP
X.509	Internet X.509 Public Key Infrastructure Certificate

1 Introdução

"*Content Centric Networking*" (CCN) é um novo paradigma de comunicações por computador centrado à volta da distribuição dos conteúdos, por oposição a uma comunicação centrada em máquinas origem/destino. Esta abordagem alternativa à arquitetura de rede, baseada no princípio em que a comunicação em rede deverá permitir ao utilizador focar-se nos dados que precisa por oposição a referenciar em específico, uma localização física de onde os dados têm que ser obtidos.

A CCN habilita o armazenamento de conteúdo, na rede, para reduzir o congestionamento e otimizar a velocidade de entrega, uma configuração mais simples de dispositivos de rede e constrói segurança na rede ao nível dos dados.

Este tipo de redes tem vindo a ser denominada de várias formas ao longo do tempo. Por exemplo, "*Networking Named Content*" (NNC) é uma evolução de nomenclatura que tem vindo a ser utilizada para se referir ao paradigma CCN e "*Information Centric Networking*" (ICN) para designar uma abordagem sobre redes informáticas, que assentam em paradigmas centrados no conteúdo, de entre as quais se destaca no objectivo deste estudo a CCN. No fundo trata-se de vários nomes para designarem uma ideia base.

1.1 Objectivo

Esta dissertação pretende levantar as principais propostas realizadas nesta área e apresentar um entendimento dos principais desafios que se colocam a esta tecnologia. Efetuar também uma análise de implementações existentes que implementam este paradigma em cenários experimentais.

1.2 Metodologia

A presente dissertação baseou-se em pesquisas na internet de bibliografia existente sobre o tema, como ponto de partida, por ser a proposta original, a partir da qual se começou a desenvolver esta temática, estudando a ideia principal dos seus autores [Van Jacobson et al. 2009] no *Palo Alto Research Center* (PARC).

Foi efetuado um exaustivo levantamento do estado da arte, que entretanto foi sendo atualizado ao longo do 1.º semestre de 2014, uma vez que a temática tem despertado interesse na comunidade científica, que se tem vindo a juntar e a deixar o seu contributo.

Foram também estudadas as vantagens e desvantagens que uma rede CCN pode introduzir na com a sua implementação.

Em termos experimentais foi utilizado o protótipo CCNx, que é um conjunto essencial de ferramentas e funcionalidades que nos permite implementar uma rede com uma arquitetura assente na CCN, fornecendo suporte a questões como o encaminhamento, comunicação, gestão de chaves, gestão da confiança, assinatura e encriptação de conteúdos, com o qual foram efetuadas algumas experiências em laboratório.

1.3 Estrutura

Seguidamente será efetuada uma breve descrição de cada capítulo que compõe a presente dissertação.

O capítulo 2 apresenta uma visão geral sobre os problemas atuais da internet e de que forma poderá a CCN colmatar essas problemáticas. Apresenta as principais ideias chave da CCN e o seu modelo de camadas, por comparação com o atual modelo TCP/IP que é utilizado na *Internet*.

O capítulo 3 descreve a arquitetura da CCN, apresentando a sua filosofia de funcionamento, os diferentes componentes e seus respectivos elementos, assim como as suas finalidades. A estruturação da hierarquia de nomes e a sequenciação das mensagens de comunicação. A fiabilidade do transporte e a mobilidade.

O capítulo 4 aborda a implementação da CCN em larga escala. Focalizando a sua integração com as atuais redes baseadas em TCP/IP e quais os seus pontos críticos. O encaminhamento intra-domínios, a conectividade local e o múltiplo anúncio de prefixo. Bem como também são abordadas as limitações apresentadas pelo estado da arte. Os problemas da escalabilidade, a avaliação de desempenho no que concerne à utilização da ligação e *Overhead*, a divulgação dos dados e o suporte multi-interface.

O capítulo 5 abordada as questões sobre a gestão e da segurança na CCN, relativamente à segurança do conteúdo, a gestão da confiança e o controlo de acesso a conteúdos. De que forma é efetuada a gestão de chaves e a segurança de rede.

No capítulo 6 apresenta-se o protótipo CCNx, a sua descrição, principais componentes e ferramentas que o compõem, a sua instalação e algumas experiências laboratoriais que foram efetuadas.

No capítulo 7 fazem-se as considerações finais, enumeram-se os principais problemas em aberto e quais serão as perspectivas futuras deste paradigma.

2 Visão Geral

Apesar de existirem várias propostas de arquiteturas baseadas no paradigma de orientação ao conteúdo, a CCN é a que tem impulsionado mais o interesse da comunidade científica pela investigação sobre este paradigma e, além disso, pela disponibilidade de uma implementação protótipo, denominada CCNx, que nos permite a implementação e a validação de novas propostas em redes experimentais.

A CCN simplifica a resolução vários problemas de segurança que apresenta a atual arquitetura de redes, sobre a qual assenta a internet, procurando-se desta forma garantir a integridade e autenticidade dos conteúdos que são transmitidos na rede, de forma a garantir a segurança do repositório, onde estão armazenados e do percurso que os dados percorrem até ao seu destino final. Conforme se pode perceber esta metodologia é contraintuitiva, pois quando existe a necessidade de proteger um determinado conteúdo é necessário para isso proteger todos os pontos intermédios de rede, utilizados no seu encaminhamento, desde a sua origem até ao seu destino. Além disso, podemos constatar a frequência com que são efetuados ataques de negação de serviço (DoS) o que leva a pensar que o ideal seria que os mecanismos de segurança necessários para mitigar estes problemas fossem fornecidos pela própria infraestrutura de rede.

Contrariamente ao modelo atual, o TCP/IP, o modelo de comunicação da CCN centra-se no conteúdo e não na sua localização física. Portanto, não há a necessidade de proteger os seus locais de armazenamento e os pontos intermédios da rede. Para tornar isto possível, a CCN obriga a que todos os dados, que são transferidos na rede, sejam assinados, garantindo desta forma, a sua integridade e autenticidade. Na CCN está também implementado um mecanismo de agregação de pacotes, o que limita a quantidade de tráfego na rede e define que todos os *routers* realizem *cache* de conteúdos. Estas características contribuem para a mitigação de ataques de negação de serviço, que existem atualmente na arquitetura de rede, sobre a qual assenta a *Internet*.

2.1 Cenário atual

Com a massificação da utilização da *Internet* e o aumento do fácil acesso a dispositivos de criação e modificação de conteúdos multimédia e informação, hoje em dia, as redes de comunicação passam por uma fase em que grande parte dos utilizadores está a deixar de ser um mero consumidor de conteúdos, para se tornar também ele num produtor. A maioria destes conteúdos, como fotos, vídeos, e *posts* nas redes sociais, são disponibilizados na

Internet. Por outras palavras, os utilizadores focam-se no conteúdo que a rede possui, enquanto a *Internet* atual focaliza-se na sua localização de armazenamento.

O grande problema é que na arquitetura atual, segundo [Jacobson et al. 2009], toda a preocupação está centrada nos endereços, não nos conteúdos produzidos e armazenados nas máquinas que estão nesses endereços. Toda a comunicação na *Internet* foi pensada em pacotes (TCP/IP) que transportam informação, que sabem de ondem veem e para onde vão e nada sabem sobre o seu conteúdo. Esta comunicação apenas se preocupa em encontrar o melhor caminho a percorrer para chegar ao seu destino, conforme se exemplifica na figura 1.

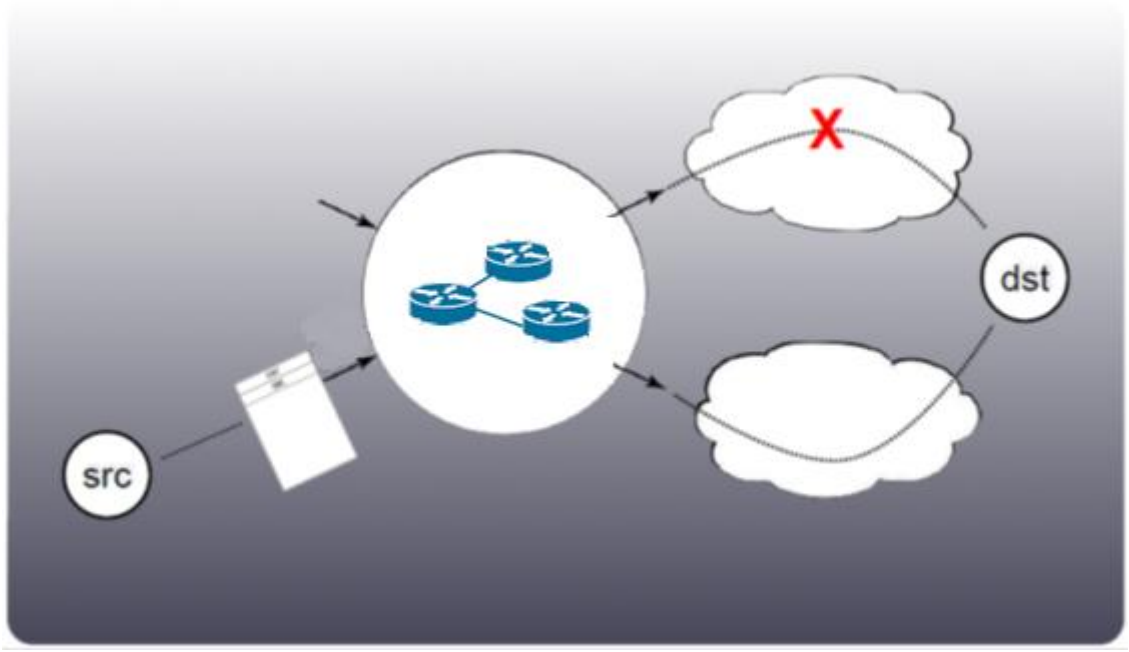


Figura 1 – Cenário atual de encaminhamento (retirado de [CCNx] e adaptada)

O problema resulta da filosofia em que foi pensada a comunicação de rede nos anos 60 e 70, em que existiam poucos computadores e a comunicação era centrada no "onde" estava a informação pretendida. Hoje em dia ninguém decora endereços, as pessoas procuram grandes quantidades de conteúdos e a internet tem dificuldade em dar resposta porque não foi pensada para responder a tal requisito. Se determinado conteúdo se torna inesperadamente muito popular, isso resulta em estrangulamento da comunicação entre o *host* que detêm esse conteúdo e todos os dispositivos que efetuaram essas solicitações. Tipicamente o fornecedor que está a alojar esse conteúdo, o que faz nestas situações ditas "virais", é inibir o acesso ao mesmo por forma a tentar controlar o congestionamento e a deterioração da qualidade de serviço (QoS). Quando colocamos um conteúdo na *Internet*, embora isso soe a um contrassenso, temos que esperar que esse conteúdo não se torne muito solicitado, senão corremos de ver o mesmo excluído (banido) da rede. E tudo isto por

culpa da arquitetura atual da Internet que não foi desenhada para responder a tais comportamentos.

Conforme podemos observar na Figura 2, se o mesmo conteúdo for solicitado por vários dispositivos, no atual modelo TCP/IP sobre o qual assenta a Internet, tem que forçosamente existir uma ligação independente para cada dispositivo, mesmo que ele esteja na mesma rede que outro que também solicitou esse conteúdo.

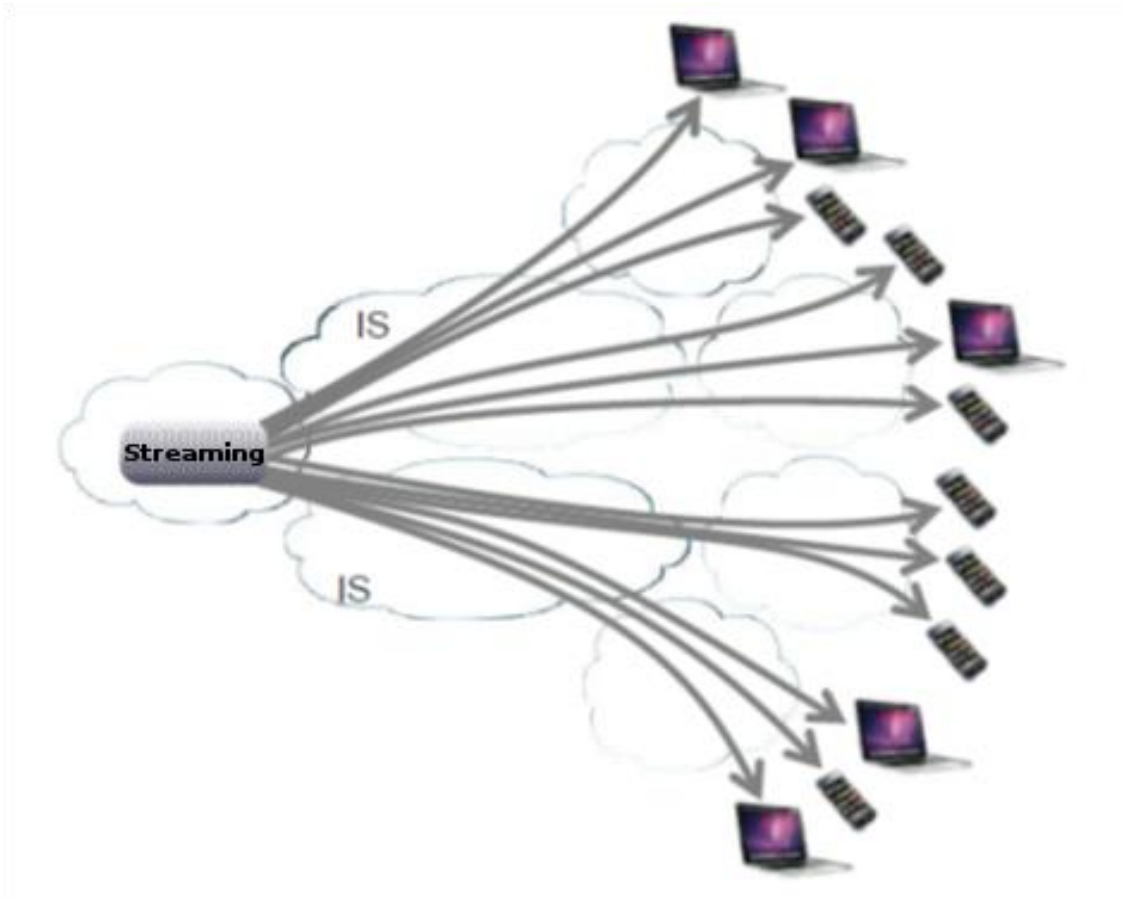


Figura 2 – Cenário atual na distribuição de conteúdos (retirado de [CCNx] e adaptada)

Desta incompatibilidade de modelos, resulta uma série de problemas relacionados, tais como a disponibilidade de conteúdo, segurança e dependência da sua localização física. Para Van Jacobson [Jacobson et al. 2009], a resolução destes problemas passa por substituir simplesmente o "*onde*" pelo "*o quê*" em relação ao conteúdo.

Uma vez que o conteúdo é o mais importante para os utilizadores, os serviços oferecidos pela rede devem ser orientados ao conteúdo em si e não à sua localização física. Assim, os utilizadores devem ser capazes de solicitar conteúdos pelo nome e compete à rede localizar esses conteúdos, onde quer que eles se encontrem. Isto é um dos principais pilares do

paradigma da comunicação baseada em conteúdos "*Content Centric Networking*". O aumento da eficiência na entrega e da disponibilidade do conteúdo são duas das suas principais vantagens, assim como uma implementação de mecanismos de segurança mais simplificada.

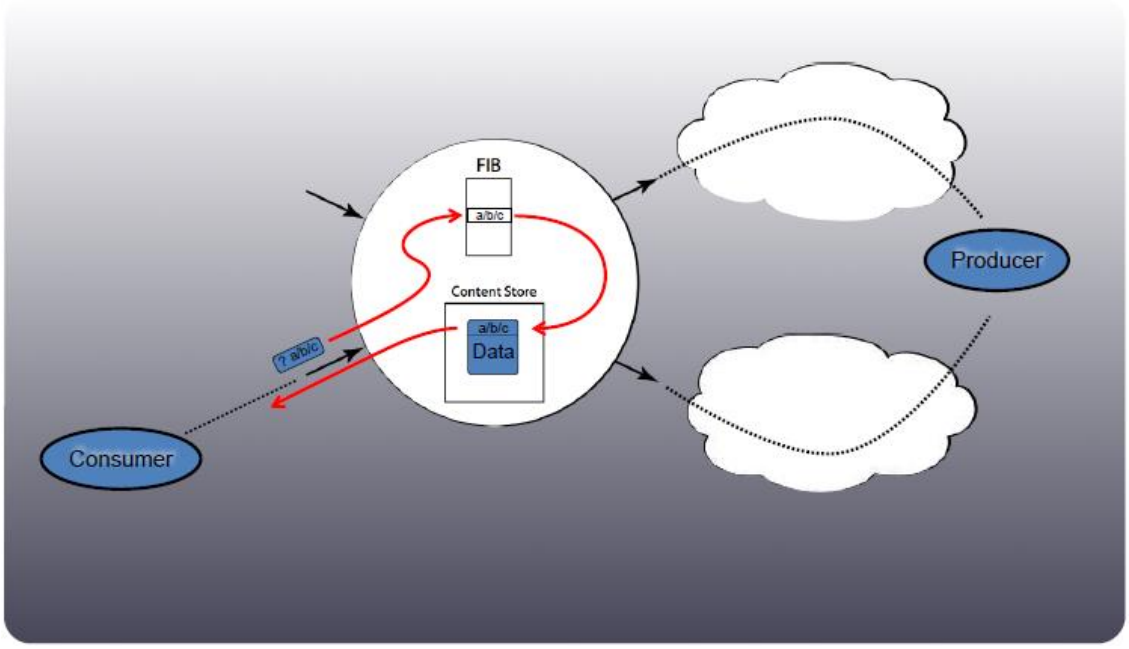


Figura 3 – Abordagem CCN (retirado de [CCNx])

A Figura 3 apresenta em traços gerais a abordagem CCN e os seus principais componentes. O "*Consumer*" representa os consumidores ou seja os utilizadores que solicitam conteúdos e que podem simultaneamente ser também "Producer" ou seja produtores de conteúdo. A FIB representa uma base de dados onde são guardadas as rotas para encaminhamento de Interesses à fonte do conteúdo. Várias interfaces de saída por destino, sem árvore de expansão. O "Content Store" guarda pedaços conteúdo que permanecem válidos após o encaminhamento. É utilizado como *buffer* de memória para economizar largura de banda, todos os nós fornecem *caching*. No capítulo 3 iremos apresentar de forma mais detalhada cada um destes componentes e qual a importância do seu papel no funcionamento do paradigma da CCN.

Desta forma poderemos substituir o atual cenário ilustrado na Figura 2, por um cenário muito mais eficiente, do ponto de vista da transferência de conteúdos, conforme o ilustrado na Figura 4, que representa o novo modelo de distribuição de conteúdos, onde os conteúdos são, ainda que de forma temporária, armazenados em *cache* para que, o nó mais próximo que possua determinado conteúdo solicitado, o possa fornecer sem a necessidade de este ter que ser solicitado à sua fonte inicial, conforme iremos detalhar no subcapítulo 2.2, onde serão apresentadas as principais ideias chave.

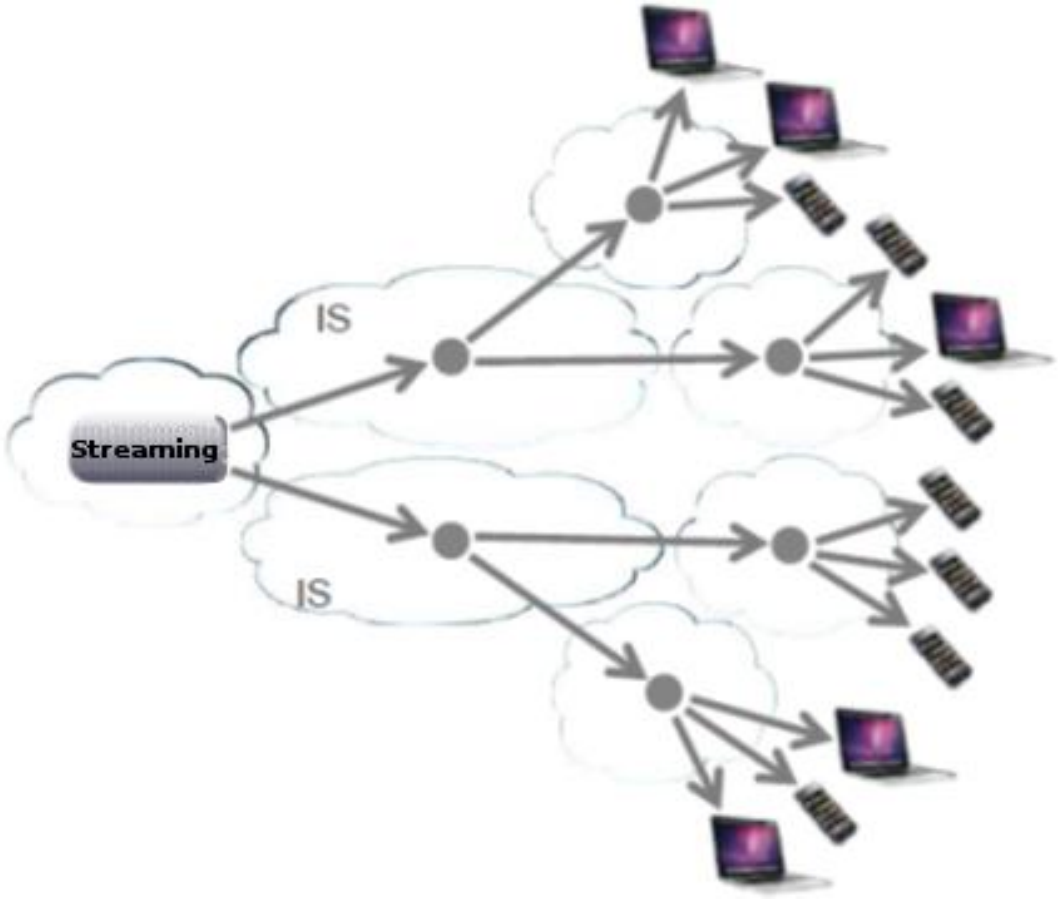


Figura 4 – Novo modelo de distribuição de conteúdos na CCN (retirado de [CCNx] e adaptada)

De acordo com [Martins and Alberti 2011], atualmente nas redes, o endereço IP está incumbido da dupla funcionalidade de identificação e localização dos *hosts*. Ou seja, a mobilidade na rede muitas vezes leva à troca do endereço IP, o que pode levar indiretamente à troca de identificação na rede. Na realidade é como se trocássemos de identificação ao nos movimentarmos, o que não faz qualquer sentido. Por este motivo é tão difícil identificar a origem de um ataque numa rede IP. A separação da identificação e localização que estão identificadas no endereço IP permite a melhoria do suporte à segurança, mobilidade e *multihoming*. Além disso, a agregação das informações na *Web* ao URL (*Uniform Resource Locator*), ao endereço do domínio e do *host* onde elas se encontram, exige o conhecimento prévio da localização da informação para que o acesso à mesma possa ser feito.

O modelo atual, segundo defende [Jacobson et al. 2009] apresenta graves problemas de segurança e um dos principais problemas do modelo TCP/IP é a proteção do conteúdo. No modelo TCP/IP, para garantirmos a autenticidade e a integridade dos dados transferidos na rede, é necessário garantir a segurança do repositório e do percurso que os dados devem percorrer até ao seu destino final, conforme representado na Figura 5.

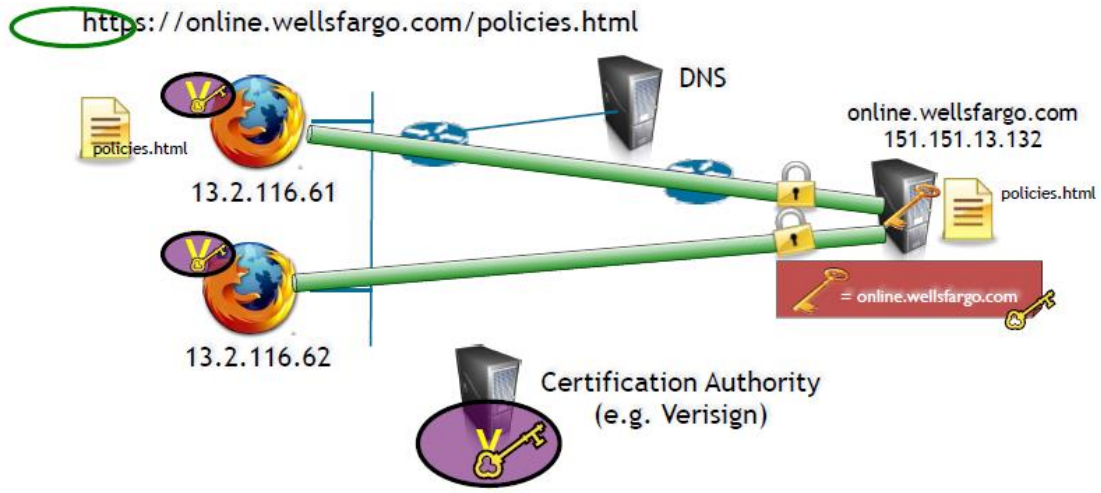


Figura 5 – Problemas de Segurança no modelo atual (retirado de [CCNx])

Nas redes centradas em informação o fluxo de mensagens é dirigido para os nós que manifestam o seu interesse através de identificadores de nomes da informação, em vez de nomes de interfaces de *hosts*. As redes centradas em informação possuem como funcionalidade principal a interligação dos produtores de conteúdo aos consumidores, independente das localizações dos *hosts* envolvidos na comunicação, conforme exemplificado na Figura 6.

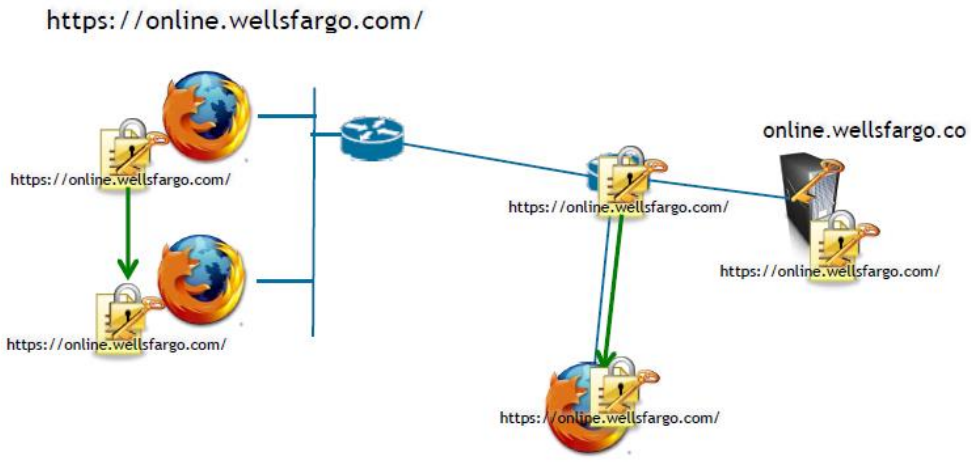


Figura 6 – Novo modelo de segurança baseada no conteúdo (retirado de [CCNx])

Em referência a [Ribeiro et al., 2012], desta forma, consegue-se um aumento da eficiência na disponibilidade e disseminação da informação, já que a obtenção dos conteúdos pode ser beneficiada ao utilizar cópias ao longo da rede. Assim, do ponto de vista do utilizador, toda informação terá um identificador único e esta informação poderá ser recuperada de forma

mais eficiente já que a informação pode ser encontrada num local mais próximo ou até mesmo no seu ambiente local, além do aumento do desempenho e facilidade no acesso a estas informações. Ou seja, o grande desafio destas redes é, portanto, estabelecer um padrão para hierarquia da informação de forma consistente e eficiente, além de fornecer soluções para encaminhamento das informações baseadas em informações da sua nomenclatura.

2.2 Principais ideias chave

A arquitetura do CCN – "*Content Centric Network*" é centrada nos dados, contrariamente ao modelo TCP/IP que não conhece o conteúdo dos dados que transporta e apenas sabe de onde vem e para onde vão

A arquitetura de uma rede CCN foi projetada para permitir que os utilizadores sejam capazes de obter conteúdos de uma forma mais eficiente e segura. Na arquitetura de rede, sobre a qual assenta a *Internet*, atualmente, quando um utilizador deseja obter um determinado conteúdo, tem que o pedir diretamente ao servidor onde esse conteúdo está guardado. A CCN apresenta um comportamento completamente oposto a este cenário, porque na CCN os dados têm um nome e não um local. Este novo modelo não necessita de DNS, porque os dados desejados são pedidos diretamente ao nível de rede e não a um endereço de uma máquina que têm esse conteúdo, desta forma não há desperdício de tempo a processar a resolução de nomes e encaminhamento, dado que uma cópia válida do conteúdo pode ser obtida do nó mais próximo que o tenha em cache. Este comportamento fornece mais segurança e maior rapidez. Atendendo a que a CCN introduz uma nova filosofia em que os dados têm um nome e não um local, isto permite que os utilizadores (consumidores) peçam conteúdos indicando o seu nome, sem terem que se preocupar com o local onde eles estão guardados, pois é da responsabilidade da rede encontrar e devolver o conteúdo que lhe foram pedidos. Conforme apresentado por [Jacobson et al. 2009], onde podem ser consultados mais detalhes.

Ainda de acordo com [Ribeiro et al., 2012], os *routers* utilizados nas implementações CCN, também designados por *routers* de conteúdo, efetuam um armazenamento de dados que recebem, também conhecido por cache, para desta forma proporcionarem uma maior disponibilidade de conteúdos. Tendo em conta que o armazenamento de dados, tipicamente em disco rígido, é sem sombra de dúvida mais económico que contratar uma maior largura de banda, podemos concluir que com a sua utilização conseguimos fazer com que pedidos posteriores do mesmo conteúdo sejam obtidos da cache, do nó mais próximo, reduzindo o tempo de resposta, bem como uma redução do consumo da largura de banda. A ideia é que qualquer nó que tenha uma cópia válida do conteúdo que está a ser pedido possa disponibilizá-lo.

Apesar da estrutura adotada pela CCN ser muito similar à estrutura das redes TCP/IP, existem algumas diferenças fundamentais. Uma dessas diferenças é que conforme abordado, no subcapítulo 2.3 o modelo de camadas, o principal ponto de controlo mais refinado da informação na CCN passa a ser o próprio conteúdo e não o protocolo IP (*Internet Protocol*), pois enquanto no modelo TCP/IP os conteúdos são colocados em pacotes e etiquetados com uma origem e um destino o que provoca algum *overhead*, neste novo modelo o conteúdo é colocado em pequenos pedacinhos apenas com uma etiqueta referente ao seu conteúdo, conforme iremos abordar com maior detalhe no capítulo 3. Esta característica salienta bem como a CCN é um paradigma de orientação ao conteúdo. Outra das diferenças é a criação de uma nova camada especificamente para a segurança. Como é sabido o protocolo TCP/IP não conhece os dados que transporta, portanto será fácil de concluir que não é possível existir segurança de dados nem uma transmissão eficiente de dados para vários pontos de rede. Dado que, na CCN, os conteúdos podem ser fornecidos por qualquer um dos nós da rede, sejam eles confiáveis ou não, é necessário fornecer garantias da sua integridade e autenticidade, sem ser preciso implementar segurança em toda a infraestrutura da rede, pois a integridade e confiança derivam dos dados, não do canal de onde proveem. Os dados são autenticados e seguros, e não as ligações que atravessam. As questões relacionadas com a camada de segurança serão discutidas no Capítulo 5. O modelo de camadas da CCN, comtempla também uma nova camada *"Strategy"* sobre a qual falaremos, mais adiante no ponto 3.7 do capítulo 3.

Os pacotes trocados na CCN são livres de *loops*. Esta propriedade possibilita que os *routers* os possam encaminhar utilizando múltiplas interfaces ao mesmo tempo e possam utilizar qualquer tecnologia disponível, tal como Ethernet, 3G, Bluetooth e IEEE 802.11. Para Van Jacobson [Jacobson, et al., 2009], tudo o que transporta bits no tempo ou espaço pode e será usado para comunicar. Os *routers* numa implementação CCN, conforme especificado na camada *"Strategy"*, tem a possibilidade de utilizar, para o encaminhamento dos pacotes, estratégias diferentes, em vez utilizarem todas as interfaces disponíveis para o encaminhamento, pois isso não seria um cenário muito eficiente. Por último [Ribeiro et al., 2012], uma propriedade fundamental é que a CCN pode ser utilizada sobre o protocolo IP, possibilitando assim uma implementação incremental e de forma controlada.

2.3 Modelo de camadas

O modelo de camadas adotado pela CCN é muito similar ao modelo de camadas do protocolo TCP/IP, mas existem algumas diferenças fundamentais ao nível da camada 2 que fazem com que suas propriedades sejam mais interessantes. Além disso para Van Jacobson [Jacobson, et al., 2009], a CCN pode ser implementada como uma camada sobre qualquer coisa incluindo sobre o TCP/IP.

De acordo com [Jacobson, et al., 2009] a CCN difere do TCP/IP de várias formas, sendo duas delas a camada "Strategy" e a camada "Security" que são duas novas camadas da sua pilha de protocolo, conforme Figura 7. A CCN consegue tirar partido máximo de várias ligações em simultâneo, tais como Ethernet, 3G, Bluetooth e IEEE 802.11 e tudo por causa da sua simples relação com a camada 2. A camada "Strategy" efetua um refinamento da optimização de escolhas dinâmicas para explorar da melhor forma várias ligações em que as condições se podem alterar a qualquer momento. Na CCN o conteúdo segura-se a si mesmo, em vez de criarmos ligações seguras por onde ele viaja, evitando assim muitas da vulnerabilidades *host-based* de que sofrem as redes IP.

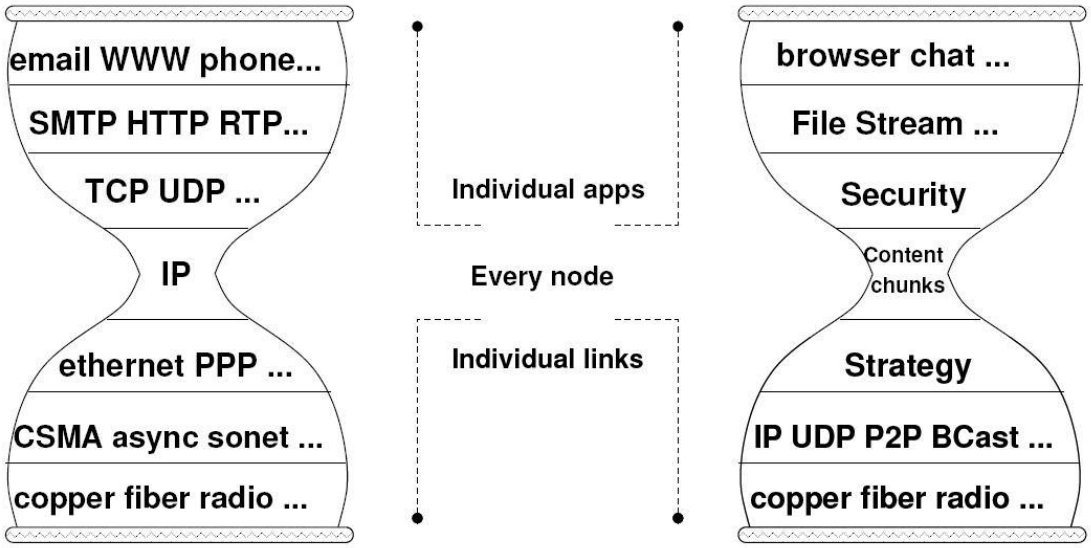


Figura 7 – Novo modelo de camadas CCN comparado com o modelo da pilha de protocolos TCP/IP (retirado de [Jacobson et al. 2009])

3 Arquitetura da CCN

Neste capítulo vamos descrever em que consiste a arquitetura da CCN, quais são os elementos que a compõem e qual o papel de cada um destes elementos.

A comunicação na CCN é efetuada utilizando apenas dois pacotes, sendo eles o pacote "*Interest*" e o pacote "*Data*". Essa informação transportada por estes pacotes é armazenada em nós, em que cada nó é composto por um armazenador de conteúdos "*Content Store*" (CS), uma tabela de interesses pendentes "*Pending Interest Table*" (PIT) e uma base de informações de encaminhamento "*Forwarding Information Base*" (FIB). Os nós são todos os dispositivos que compõem uma rede informática tais como *routers*, *switches*, computadores e dispositivos móveis. Em suma os nós são todos os ativos de rede capazes de armazenar conteúdos ainda que de forma temporária. Esta capacidade de armazenamento que estes dispositivos podem ter é designada por *cache*.

De seguida será apresentado de forma detalhada cada um dos elementos que compõem o paradigma da CCN, ou seja os tipos de pacotes utilizados e a composição interna de cada nó.

3.1 Tipos de Pacotes (*Interest, Data*)

A CCN, segundo [Jacobson, et al., 2009], utiliza apenas dois tipos de pacotes para efetuar toda a troca de informação. Esses dois tipos de pacotes são, o pacote "*Interest*" e o pacote "*Data*", ilustrados na Figura 8, onde cada pacote "*Interest*" é consumido por um pacote "*Data*".

O modelo é baseado em "*Pull-based*" com dois tipos de pacotes "*Interest*" e "Data", onde o consumidor impulsionado envia um pedido "*Interest broadcast*", espera pelo "*Data*". O pacote "*Data*" transmitido em resposta, satisfaz o pedido do pacote "*Interest*".

Conforme apresentado por [Ribeiro et al., 2012], para um utilizador solicitar conteúdos pelo seu nome, à rede, utiliza os pacotes "*Interest*". Este tipo de pacote, além do nome do conteúdo solicitado também transporta ainda as seguintes informações:

Resposta: Na CCN os conteúdos podem ser obtidos a partir da cache do nó mais próximo da rede ou da sua fonte. Há casos em que o utilizador sabe, *à priori*, nomeadamente na obtenção de conteúdos dinâmicos, que o conteúdo solicitado não estará armazenado em na cache dos nós da rede. Nessa condição, este campo, apesar de não estar contemplado na estrutura da Figura 8, permite ao utilizador indicar que o conteúdo deve ser devolvido diretamente da sua fonte e desta forma evita-se processamento desnecessário.

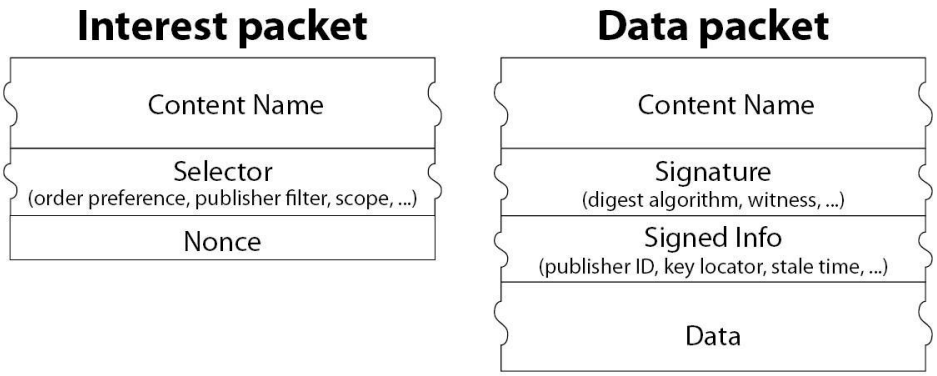


Figura 8 – Tipos de pacote (retirado de [Jacobson et al. 2009])

Filter: O conteúdo, na CCN, é pedido através do seu nome, que pode ser completo ou apenas parcialmente pelo seu prefixo. Assim que um pacote *"Interest"* é recebido por um *router* ou uma fonte de conteúdos, é efetuada uma comparação de maior prefixo entre o nome que consta no pacote *"Interest"* e os nomes dos conteúdos armazenados.

Se for encontrada uma correspondência, o conteúdo é enviado ao utilizador que o solicitou. Se existir mais do que uma correspondência para esse conteúdo, nesse casso será necessária fazer uma escolha de dentre as várias disponíveis, uma vez que apenas uma resposta pode ser enviada para cada pacote *"Interest"* enviado. Deste modo, o filtro de exclusão é utilizado com uma lista de parâmetros que não devem constar no nome do conteúdo a devolver. Os nomes são formados por um conjunto de componentes separados entre si pelo carácter "/ ", representando uma hierarquia [Jacobson et al. 2009], conforme será explicado no subcapítulo 3.3. Vamos supor, por exemplo, que temos armazenados em *cache* num *router*, os conteúdos `/isep.ipp.pt/entidade/apresentacao.mp4` e`/isep.ipp.pt/noticias/2014/reportagem.mp4`. Se esse *router* receber um pacote *"Interest"* para o nome de conteúdo `/isep.ipp.pt` e que o filtro de exclusão contém o parâmetro `/noticias`.

A comparação de maior prefixo, neste caso, iria ter como correspondência ambos os conteúdos presentes na cache. Mas, após ser analisada a lista do filtro de exclusão, o conteúdo `/isep.ipp.pt/noticias/2014/reportagem.mp4` seria eliminado da seleção, sendo devolvido apenas o conteúdo `/isep.ipp.pt/entidade/apresentacao.mp4` ao pedido.

Scope: A CCN trata os pedidos, por padrão, de forma recursiva. Na prática o que acontece é que, se um *router* não conseguir satisfazer determinado pedido *"Interest"*, o pacote será encaminhado para o próximo nó que for determinado pelo protocolo de encaminhamento. De forma a ser possível alterar este comportamento, utiliza-se o campo *scope* para definir qual o nível na hierarquia da rede, até onde o pacote *"Interest"* deve ser processado.

Se, por exemplo, o valor deste campo for definindo como ”2”, o pacote *“Interest”* pode alcançar no máximo o *router* do próximo salto.

Hash: O utilizador deve especificar neste campo quem deverá ser o emissor do conteúdo pedido, colocando aqui uma *hash* (SHA-256) da chave pública do emissor pretendido. Desta forma só será devolvido o conteúdo pedido se a *hash* da chave pública desse pacote *“Interest”* for igual à *hash* da chave pública que constar no pacote *“Data”*.

Nonce: O campo *nonce* é utilizado para impedir a existência de *loops* na rede e é preenchido com um número aleatório, possibilitando assim ao *router* a identificação de pacotes *“Interest”* duplicados.

Para transportar os conteúdos pedidos pelos pacotes *“Interest”*, a CCN utiliza os pacotes *“Data”*. Todos os pacotes *“Data”* são assinados pelo seu emissor de forma a garantir as questões de segurança. Para possibilitar a verificação desta assinatura, os pacotes *“Data”* também armazenam na sua estrutura o algoritmo de criptografia utilizado e um localizador que possibilita a obtenção da chave pública do seu emissor.

Um utilizador mal-intencionado, ou habitualmente designado por “Hacker” pode utilizar *Botnets*, para criar uma grande quantidade de tráfego em redes IP, através da utilização típica dos protocolos TCP, UDP ou ICMP e direcionar este tráfego para um alvo em específico, tendo como finalidade provocar o esgotamento seus recursos de rede e processamento de forma a que este fique indisponível.

Num pacote *“Interest”* não é possível indicar um destinatário, portanto podemos concluir que só é possível a um utilizador mal-intencionado tentar ataques ao emissor do conteúdo, uma vez que os pacotes *“Interest”* são os únicos que são encaminhados para ele. No entanto, após a primeira obtenção de conteúdo solicitado os próximos pedidos desse mesmo conteúdo, mesmo que sendo pedidos por outros, já serão satisfeitos pela sua existência em cache no nó de rede mais próximo, que tiver uma cópia válida desse conteúdo. Assim sendo, esta propriedade reduz eficazmente este tipo de ataque, pois nem todos os pacotes de *“Interest”* serão enviados até ao emissor do conteúdo. Pode-se desta forma concluir que, uma vez que na CCN os pedidos são baseados em encaminhamento, não há a possibilidade de ataques do tipo DoS. Ver [Ribeiro et al., 2012] para uma consulta mais detalhada sobre este assunto.

3.2 Estrutura do Nó da CCN

Cada nó da CCN é composto por três estruturas de dados para operações de encaminhamento de pacotes: um armazenador de conteúdo (*Content Store* - CS) para cache temporário de dados recebidos, uma tabela de interesses pendentes *(Pending Interest Table* - PIT) e uma base de informações de encaminhamento (*Forwarding Information Base* - FIB).

A prioridade de procura na tabela é CS > PIT > FIB

3.2.1 Content Store

A CS – "*Content Store*" guarda pedaços conteúdo que permanecem válidos após o encaminhamento. É utilizado como *buffer* de memória para economizar largura de banda, todos os nós fornecem *caching*. Existem várias políticas de *caching* possíveis (LRU, LFU).

A *"Content Store"*, conforme exposto em [Ribeiro et al., 2012], possibilita uma maior eficiência no acesso a conteúdos pelos consumidores, uma vez que utiliza a técnica de armazenamento de conteúdo em cache *routers*. Para que isto seja possível a CS utiliza uma tabela de índices que vai sendo atualizada com os nomes dos conteúdos que estão guardados na sua cache, possibilitando desta forma a verificação de uma correspondência do nome do pacote *"Interest"* ou pacote *"Data"* que possuam o nome dos conteúdos armazenados. A frequência de atualização desta tabela indexada depende da política de substituição e do período de validade que estiverem definidos para a utilização da *cache*.

3.2.2 Pending Interest Table

A PIT – "*Pending Interest Table*" mantém o caminho de encaminhamento de Interesse, a partir de diferentes interfaces. O pacote "Data" segue o pacote "*Interest*" de volta para o pedido, se não houver dados ocorre um "*Time Out*".

Resumidamente e de acordo com o exposto em [Ribeiro et al., 2012], a PIT é uma tabela indexada, que vai sendo atualizada, com nomes de conteúdo. Esta tabela guarda uma lista de interfaces de onde foram recebidos pedidos de pacotes *"Interest"* para um determinado conteúdo. Assim, quando um *router* recebe um pacote *"Data"*, verifica na tabela qual a lista de interfaces por onde foram recebidos pedidos com pacotes *"Interest"* e atualiza esta informação. Desta forma o pacote "Data" é encaminhado para todas as interfaces que constam dessa lista, o que nos leva a concluir que os conteúdos são enviados aos consumidores no sentido inverso ao que foi percorrido pelo seu correspondente pacote *"Interest"*.

É graças a este mecanismo, que a CCN consegue a agregação de pacotes *"Interest"*, e desta forma, consequentemente a redução da utilização da largura de banda. Dado que os pacotes *"Data"* efetuam sempre o caminho inverso ao percorrido pelo seu correspondente pacote de *"Interest"*, conclui-se então que os *routers* da CCN não necessitam de encaminhar mais do que um pacote de *"Interest"* para o mesmo conteúdo. Sempre que um novo pacote *"Interest"* for recebido, é apenas necessário adicionar a sua interface à lista de interfaces da PIT e deste modo, quando o pacote "Data" correspondente for recebido pelo *router*, todos os pacotes *"Interest"* serão satisfeitos de forma conjunta.

Atendendo a que um pacote *"Interest"* pode ser encaminhado através de múltiplas interfaces, diversas cópias de um pacote *"Interest"* podem seguir caminhos diferentes, mas passarem por alguns *routers* em comum. Então, de forma a evitar que os pacotes *"Interest"* efetuem *loops*, ou seja o mesmo pacote *"Interest"* seja recebido mais de uma vez pelo mesmo *router e seja entendido* como se tratasse de pacotes *"Interest"* distintos, a PIT também guarda os *nonces* de cada um destes pacotes. Assim, se for recebido pelo mesmo *router* mais do que um pacote *"Interest"* para um mesmo conteúdo, através da utilização de uma comparação do *nonce* do pacote *"Interest"* com aquele que se encontra armazenado na PIT, esta situação é detectada e esse pacote *"Interest"* é então descartado. Mesmo que os pacotes *"Interest"* não fiquem pendentes eternamente e caso o conteúdo solicitado não exista, é associado um temporizador a cada uma destas entradas da PIT e assim que estas expirarem a entrada correspondente é eliminada.

3.2.3 Forwarding Information Base

A FIB – "*Forwarding Information Base*" é por analogia a conhecida "*IP Routing Table*". É onde são guardadas as rotas para encaminhamento de Interesses à fonte do conteúdo. Várias interfaces de saída por destino, sem árvore de expansão.

A FIB de um *router* CCN, segundo [Ribeiro et al., 2012] é muito similar à de um *router* IP, contudo difere fundamentalmente em dois pontos. Pois genericamente uma FIB IP associa um prefixo de rede específico a uma única interface de saída e calcula o custo do melhor caminho através do protocolo de encaminhamento. Além disso uma FIB CCN pode conter uma lista com várias interfaces de saída, o que pode permitir que um pacote *"Interest"* seja encaminhado através de múltiplos caminhos. O segundo ponto fundamental é que ao contrário de uma entrada na tabela de encaminhamento IP, que apenas possui as informações do próximo salto, uma entrada FIB CCN possui informações que facultam todo o processo de encaminhamento dinâmico dos pacotes *"Interest"*.

A CCN possui uma estrutura que possibilita que os protocolos de encaminhamento atualmente utilizados na Internet possam ser adaptados para passar a trabalhar com prefixos de nome em vez e trabalharem com endereços IP. Estes protocolos de encaminhamento

fornecem à FIB qual é a interface de saída para um dado pacote *"Interest"*, com base no nome de conteúdo que este possui. Assim sendo, sempre que o protocolo de encaminhamento deixa de anunciar um determinado prefixo, este não é logo eliminado da FIB, porque alguns interesses para este prefixo podem ainda estar em circulação. As entradas da FIB só são eliminadas após expirar a validade que estiver definida. Para evitar que os pacotes *"Interest"* sejam descartados, derivado de alterações de rotas durante a fase de convergência do protocolo de encaminhamento, as entradas da FIB só expirarem de acordo com o tempo de duração que lhe estiver associado.

Se um nó receber um pacote de *"Interest"* e o conteúdo solicitado não estiver guardado na sua CS e simultaneamente não existir uma entrada que lhe corresponda, na PIT ou na FIB, então esse pacote *"Interest"* será descartado. Esta situação acontece porque o nó não têm o conteúdo solicitado e não sabe como encaminhar o pedido na direção de outo nó que esteja ao seu alcance e possua uma cópia válida deste conteúdo. Caso o pacote *"Interest"* esteja pendente na PIT de outros *routers* ao seu alcance, essas entradas irão eventualmente expirar e esperar que as entradas da PIT expirem pode provocar uma perda de tempo demasiadamente excessiva. Devemos também ter em conta que se um pacote *"Interest"* não conseguir encontrar o conteúdo solicitado, todos restantes pacotes *"Interest"* para esse mesmo conteúdo serão bloqueados, graças à agregação desses pacotes no *router*. Deste modo, [Yi et al. 2012] propõem que o nó envie, pela mesma interface que o mesmo foi recebido, após descartar o pacote *"Interest"*, uma confirmação negativa (NACK) onde consta o nome de conteúdo que está no pacote *"Interest"* descartado, e um código de erro, para ajudar na identificação do problema. Sempre que é recebido um NACK num *router* para determinado conteúdo, ele deverá tentar encaminhar novamente o pacote *"Interest"* através de um conjunto de interfaces alternativas. Caso não exista esse conjunto, então o *router* também não será capaz de atender ou encaminhar os pacotes *"Interest"* para este conteúdo. Desta forma deve ser encaminhado pelo *router,* um NACK, através das interfaces existentes na entrada da PIT correspondente, sendo este removido em seguida. A utilização desta técnica permita à CCN detectar de forma mais eficiente a existência de problemas na obtenção de conteúdos.

3.2.4 Como pedir Conteúdos

O consumidor envia um pacote *"Interest"* à rede que contém, no mínimo, o nome do mesmo. O *router* após receber o pacote *"Interest"* extrai o nome do conteúdo e efetua uma procura pelo maior prefixo no seu CS. Se for encontrada uma correspondência, então será gerado um pacote *"Data"* no *router* e envia-o para a interface por onde chegou o pacote *"Interest"*. Se não for encontrada uma correspondência o *router* verifica se já existe uma entrada na sua PIT para o conteúdo. Em caso afirmativo, o *router* verifica se o *nonce* do interesse recebido está contido na lista de *nonces*, que é armazenada na PIT. Sempre que é recebida uma cópia de um pacote *"Interest"*, este deve ser descartado. Se não for uma cópia, então trata-se de um pacote *"Interest"* novo que é guardado na PIT, assim como a sua interface de entrada e o seu "*nonce*" e o pacote *"Interest"* é descartado. Se não existir nenhuma na PIT, o *route*r efetua

uma procura de maior prefixo na sua FIB e tenta encontrar interfaces de saída para o pacote *"Interest"* ser encaminhado. Caso não consiga encontrar uma interface de saída para o conteúdo solicitado pelo pacote *"Interest"*, então o pacote é descartado e é enviado um NACK para a sua interface de entrada. Se encontrar uma interface de saída, nesse caso é criada uma entrada na PIT que contem a interface de entrada e o "*nonce*" do pacote *"Interest"*. O *router* a seguir decide quais as interfaces pelas quais vai encaminhar o pacote *"Interest"* através de uma consulta a sua camada *"Strategy"*.

Quando um *router* recebe um pacote "Data" efetua uma extração ao nome do conteúdo e efetua uma verificação na PIT, se existe alguma entrada para o mesmo. Caso não exista isso significa que o conteúdo não foi solicitado e é descartado o pacote *"Data"*. Caso exista, o conteúdo do pacote de *"Data"* dados é armazenado no CS desse *router* e posteriormente encaminhado por todas as interfaces que estejam na entrada da PIT. Se o *router* receber um NACK e não um pacote *"Data"*, nesse caso poderá tentar reencaminhar o pacote *"Interest"* utilizando interfaces alternativas ou pura e simplesmente descartar a entrada correspondente da PIT e encaminhar esse NACK para todas as interfaces que constem nela. Pode ser consultado [Ribeiro et al., 2012] para informação mais detalhada sobre esta temática.

Segundo defendem [Martins and Alberti, 2011], um dos princípios das redes centradas na informação é que os utilizadores podem beneficiar de cópias de conteúdos espalhados ao longo da rede, permitindo otimizar a disseminação global da informação. A cache de conteúdos é portanto parte crucial nas abordagens de redes centrada na informação.

Na CCN podemos encontrar o conteúdo utilizando um mecanismo de pesquisa local ou ao longo do caminho do pacote *"Interest"* até à potencial fonte. No entanto, podem-se colocar questões acerca da utilização de *caching*, no que concerne aos seus aspectos legais e contratuais.

Por exemplo:

Quem se responsabiliza pelos conteúdos em cache ao longo da rede, os desenvolvedores do conteúdo ou os proprietários da cache?

Quais são os direitos e obrigações dos utilizadores que fornecem a cache para operadoras de serviços?

Encaminhamento Dinâmico: Atualmente na redes IP, conforme [Martins and Alberti, 2011] o encaminhamento é efetuado pelos protocolos de encaminhamento, que têm como principal responsabilidade, encontrar uma rota com o menor custo entre dois nós e propagá-la para os *routers* vizinhos na rede. Desta forma o encaminhamento de pacotes é definido exclusivamente pelo caminho indicado pelo protocolo de encaminhamento. Tendo em conta que os pacotes da CCN não efetuam *loops*, é possível, graças a isso, serem encaminhados por múltiplas interfaces ao mesmo tempo. Neste cenário o protocolo de encaminhamento passa a

ter a função de identificar todas as rotas disponíveis para um determinado nome de conteúdo e proliferar esta informação na rede, para que os *routers* possam construir as suas FIB's. Sempre que um pacote *"Interest"* necessita de ser encaminhado, pode haver várias interfaces de saída disponíveis. Desta forma o *router* pode encaminhar o pacote *"Interest"* para todas as interfaces disponíveis ou escolher um apenas um subconjunto das mesmas, em função da decisão que tomar. O facto dos pacotes de *"Data"* percorrerem obrigatoriamente o caminho inverso ao do pacote *"Interest"* é outra característica importante da CCN. Tendo em conta que, cada *router* por onde passa um pacote *"Interest"* passará obrigatoriamente o respectivo pacote *"Data"*, tirando partido desta particularidade os *routers* recolhem algumas estatísticas que podem ser utilizadas no apoio ao processo de encaminhamento de pacotes *"Interest"* e desta forma poder escolher os melhores caminhos de forma dinâmica em função das condições atuais da rede.

Neste contexto, [Yi et al. 2012] propõem que algumas informações adicionais sejam armazenadas na FIB, de forma a permitir uma classificação das interfaces. No caso de um pacote *"Interest"* ser encaminhado através de uma interface e o respectivo conteúdo também é recebido pela mesma interface, o RTT para essa interface pode ser calculado pelo *router* e armazenado na entrada correspondente da FIB. O caminho com a menor latência pode ser utilizado, caso todos os *routers* escolham as interfaces que apresentam o menor valor de RTT, para um determinado nome de conteúdo.

Além da latência, as interfaces também podem ser classificadas em função da taxa de sucesso no acesso a conteúdos. Essa classificação pode ser efetuada utilizando uma classificação por cores, onde em cor amarela serão as interfaces que acabaram de ser ligadas, ou que existem em novas entradas na FIB, indicando que estão num estado neutro. Em cor verde, a interface de chegada sempre que um nó da rede recebe um pacote *"Data"*, dando assim a indicação que o seu resultado foi positivo no acesso ao conteúdo. A cor de uma interface passa de verde para amarelo quando expira o tempo de validade de um pacote *"Interest"* pendente ou após a interface ficar sem ser utilizada durante um determinado período de tempo. No caso serem detectados erros de enlace, em uma ou mais interfaces, ou caso recebam um NACK, então estas interfaces passam à cor vermelha.

Após serem classificadas e lhes ser atribuído um ranking, que vai pesar no encaminhamento do pacote *"Interest",* dependendo da política de encaminhamento definida pelos *routers*, para escolher as interfaces. As políticas de encaminhamento podem variar conforme as regras em que se baseiam e a sua implementação pode ser efetuada, independentemente, por diferentes organizações. Se por exemplo, a política de encaminhamento definir que apenas as interfaces com a menor latência podem ser utilizadas, neste caso este processo será baseado no RTT estimado que está associado às mesmas.

Outro exemplo seria, uma determinada política que poderia definir uma preferência por um determinado vizinho. Neste caso, uma grande percentagem dos pacotes *"Interest"* seria encaminhada pelas interfaces ligadas de forma direta ou indiretamente a este vizinho.

Por último, [Yi et al. 2012] definem uma possível estratégia de encaminhamento que utiliza as informações existentes na FIB e na PIT para a escolha das interfaces que farão parte do processo de encaminhamento. A tomada de decisões depende do pacote *"Data"* recebido ser um novo pacote *"Interest"*, um pacote *"Interest"* retransmitido ou um NACK de Interesse.

Pacote *"Interest"* Novo: Quando um *router* recebe um novo pacote *"Interest"*, efetua o procedimento normal, que foi descrito anteriormente. Se o pacote *"Interest"* precisar ser encaminhado, então a interface verde com melhor ranking deve ser a escolhida. Caso não existam interfaces com esta cor então a interface amarela com melhor ranking deve ser utilizada.

NACK de Interesse: após receber um NACK de Interesse, o que significa que o pacote *"Interest"* encaminhado por uma determinada interface não pode ser satisfeito, o *router* deverá tentar reencaminhar o pacote *"Interest"* através da interface com melhor ranking, mas diferente dessa interface. Isto significa que o *router* irá efetuar uma exploração às diferentes interfaces para tentar obter esse conteúdo. Este processo não deverá demorar muito tempo, uma vez que que o conteúdo solicitado pode efetivamente não estar disponível. Então, sempre que um pacote *"Interest"* é encaminhado pela primeira vez, é iniciado um temporizador no *router*, designado por temporizador de exploração e define o tempo de expiração, tendo por base uma estimativa média de amostras de RTT. Assim, a exploração de interfaces alternativas tem início após ser recebido um NACK de interesse e finaliza assim que o *router* é bem-sucedido ou quando o temporizador de exploração termina a sua contagem.

Pacote *"Interest"* Retransmitido: Um pacote *"Interest"* é considerado retransmitido se existir uma entrada correspondente na PIT e o seu *nonce* não consta na lista de "*nonces*" desta entrada. Sempre que um *router* recebe um pacote *"Interest"* retransmitido, antes do temporizador de exploração terminar a sua contagem, não é efetuado o seu encaminhamento. Caso o temporizador já tenha finalizado a contagem, o encaminhamento será efetuado através de uma interface verde ou amarela que seja diferente daquela utilizada pelo seu antecessor.

Apesar de serem utilizadas preferencialmente, as interfaces classificadas como verdes, no processo de encaminhamento de pacotes *"Interest"*, também é fundamental realizar sondagens periodicamente às interfaces classificadas como amarelas, tendo como principal objectivo desta forma descobrir outros caminhos com melhor performance. Por exemplo, um caminho pode ficar disponível após a recuperar de falhas de enlace ou mesmo um caminho que estava classificado, anteriormente, como amarelo pode ficar verde, após o aparecer um conteúdo em cache que seja mais próximo do consumidor. Assim sendo, um *router* CCN sonda de forma proactiva as interfaces classificadas como amarelas, encaminhando uma cópia de um pacote *"Interest"* para cada uma delas. Esta sondagem fornece informações sobre o

desempenho e disponibilidade de eventuais caminhos alternativos, contudo isto pode resultar na obtenção de dados duplicados. Para controlar este problema pode-se controlar limitando a frequência da sondagem.

O encaminhamento dinâmico de pacotes *"Interest"*, além de proporcionar uma melhor utilização de recursos da rede, fornece também um mecanismo natural de proteção contra alguns tipos de ataque de negação de serviço, conforme iremos abordar mais adiante. Para uma consulta mais aprofundada sobre o tema, deverá ser consultado [Ribeiro et al., 2012].

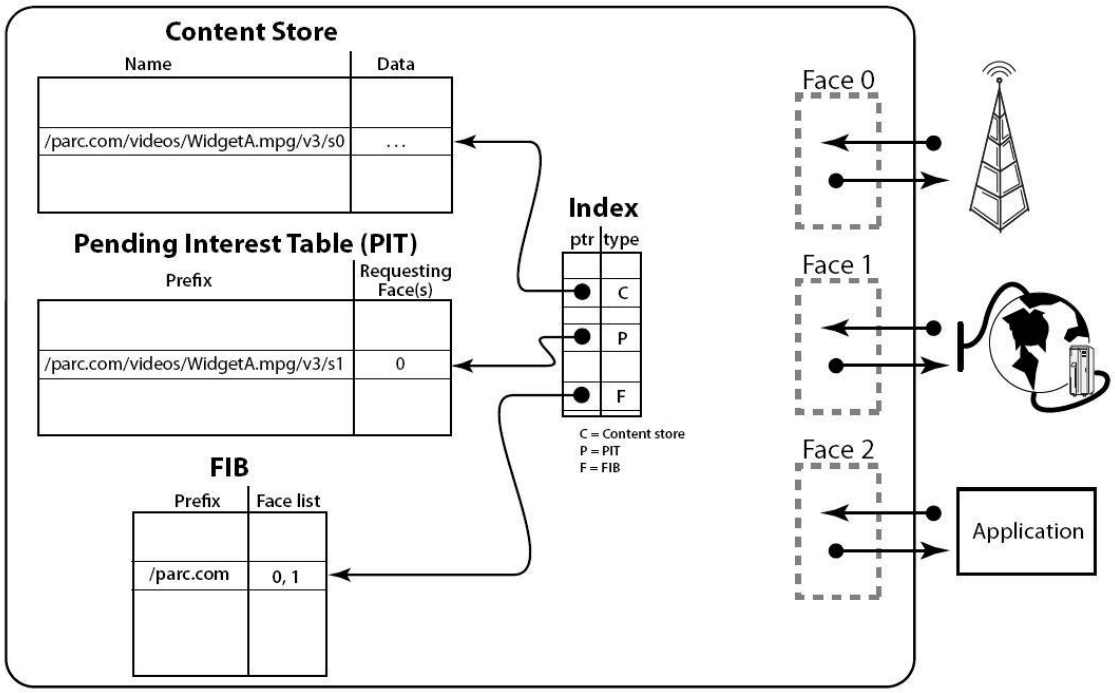


Figura 9 – Modelo de encaminhamento da CCN (retirado de [Jacobson et al. 2009])

3.3 Hierarquia de Nomes

O esquema de nomes escolhido pela CCN é muito similar ao que é utilizado para a nomenclatura de ficheiros e diretórios no sistema operativo Linux. Os nomes são formados por caracteres alfanuméricos, separados entre si pelo carácter "/ ", representando assim uma hierarquia [Jacobson et al. 2009]. Os *routers*, segundo [Ribeiro et al., 2012], não têm conhecimento da semântica dos nomes atribuídos aos conteúdos, porque estes são opacos à rede. Para a rede apenas a sua estrutura hierárquica é relevante. Deste modo, tendo em conta que os nomes têm esta estrutura, é dada total liberdade aos emissores de conteúdo

para adoptarem qualquer padrão de hierarquia de nomes que se adeqúe melhor as suas necessidades.

Por exemplo, supondo que pretendemos obter um vídeo de uma reportagem sobre o ISEP, o Instituto Superior de Engenharia do Porto, que foi publicado através do nome `/isep.ipp.pt/noticias/2014/reportagem.mp4`. Segundo, [Ribeiro et al., 2012] é importante salientar que na ausência de motores de pesquisa, os utilizadores precisarão efetuar solicitações para conteúdos informando os nomes dos mesmos. Desta forma, os nomes devem refletir o significado dos conteúdos. Isso significa que se o conteúdo é uma reportagem sobre o ISEP, então dar o nome de `aula_de_zumba.mp4` a esse conteúdo não faria sentido e confundiria os utilizadores.

A utilização deste esquema de hierarquia de nomes torna fácil o estabelecimento de relacionamentos entre pedaços de conteúdo [Ribeiro et al., 2012], como ilustrado na Figura 10.

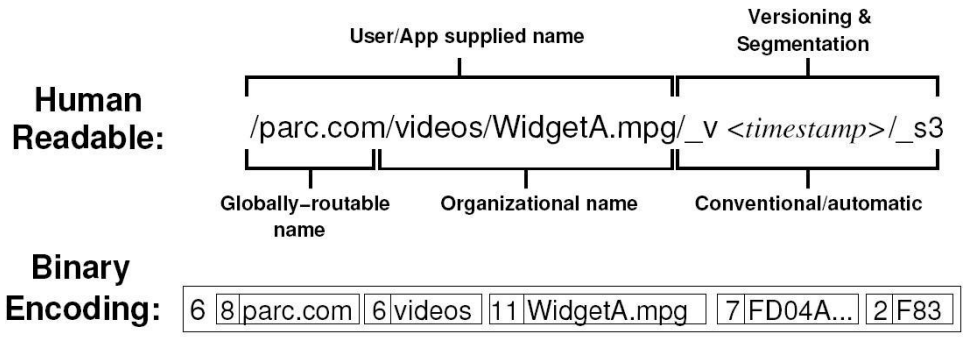


Figura 10 – Exemplo de nome de dados (retirado de [Jacobson et al. 2009])

Por exemplo, um pedaço de conteúdo do segundo segmento da primeira versão do vídeo `reportagem.mp4` poderia ser nomeado como `/isep.ipp.pt/noticias/2014/reportagem.mp4/V1/2`. Desta forma, uma requisição que utilize o nome `/isep.ipp.pt/noticias/2014/reportagem.mp4` , segundo [Jacobson et al. 2009], poderia remeter ao primeiro segmento deste conteúdo e utilizando informações existentes neste, juntamente com o conhecimento do padrão de hierarquia de nomes utilizado pelo emissor, o consumidor poderia então solicitar os segmentos seguintes. O número de possibilidades de estruturação dos nomes é realmente quase ilimitado, pois só depende da criatividade e organização dos emissores.

A utilização desta hierarquia de nomes utilizada na CCN, de acordo com [Ribeiro et al., 2012] permite outra característica muito importante que é a escalabilidade. Uma vez que o encaminhamento de solicitações é baseado nos nomes de conteúdo, a estrutura hierárquica

dos mesmos permite que os prefixos sejam agregados nos *routers*, reduzindo assim o tamanho das tabelas de encaminhamento.

A estrutura de nomes do conteúdo na CCN, segundo [Jacobson et al. 2009] é composta por nomes em texto simples, nomes intuitivos, sem utilização de criptografia, não havendo necessidade da utilização de um mecanismo indireto entre nomes e conteúdo (DNS, DHT).

Utilização de prefixos roteáveis e persistência dos nomes, proporcionam uma agregação hierárquica para o encaminhamento rápido e envio.

É utilizada uma navegação relativa começando pelo ramo mais à direita até chegar ao ramo mais à esquerda, conforme representado na Figura 11.

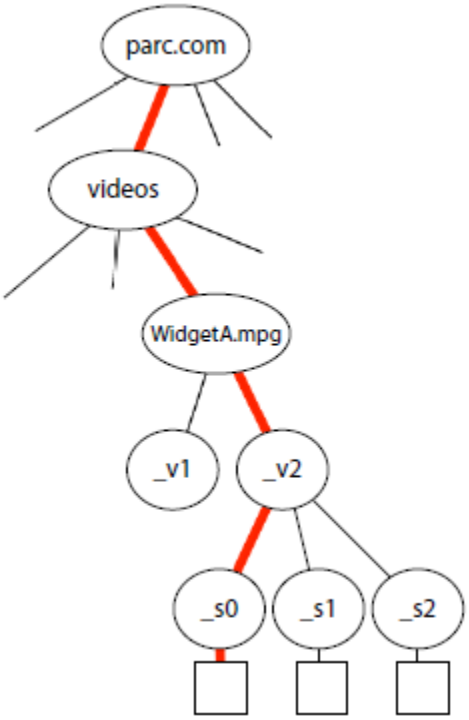


Figura 11 – Navegação de nome em árvore (retirado de [Jacobson et al. 2009])

Desta forma, defende Van Jacobson [Jacobson et al. 2009], a CCN permite uma geração dinâmica de conteúdo, com uma resolução de "*Context-aware*", utilizando uma codificação de estrutura onde o nome é dividido em componentes ou seja em octetos (bytes).

3.4 Sequência da Mensagem (pontos chave)

Numa comunicação TCP/IP entre *hosts*, os dados são identificados por uma simples numeração sequencial.

Exemplo de um pedido de uma página *Web*:

```
REQ http://nytimes.com/today ->
<- RESP http://nytimes.com/today
http://nytimes.com/20120406/index.html
<NameMAC 3fde... />
<DataMAC 07a2... />
<html>
...
<html/>
```

A CCN necessita de algo mais sofisticado, de acordo com [Jacobson et al. 2009], porque os consumidores da informação, pedem pequenos pedaços de uma grande coleção de dados e vários recipientes podem partilhar esses mesmos pedaços de dados.

A localização e partilha de dados são agilizadas através da utilização de uma hierarquia e agregação de nomes, que são pelo menos em parte legíveis para o ser humano, refletem alguma estrutura organizacional da sua origem e não apenas a sequência de uma conversação efémera.

Na figura 12, conforme exposto por [Cholez, 2012], procura-se representar o esquema minimalista de uma possível sequência da mensagem na comunicação assente no paradigma CCN. Desde o envio do pacote *"Interest"* ao retorno do pacote "Data".

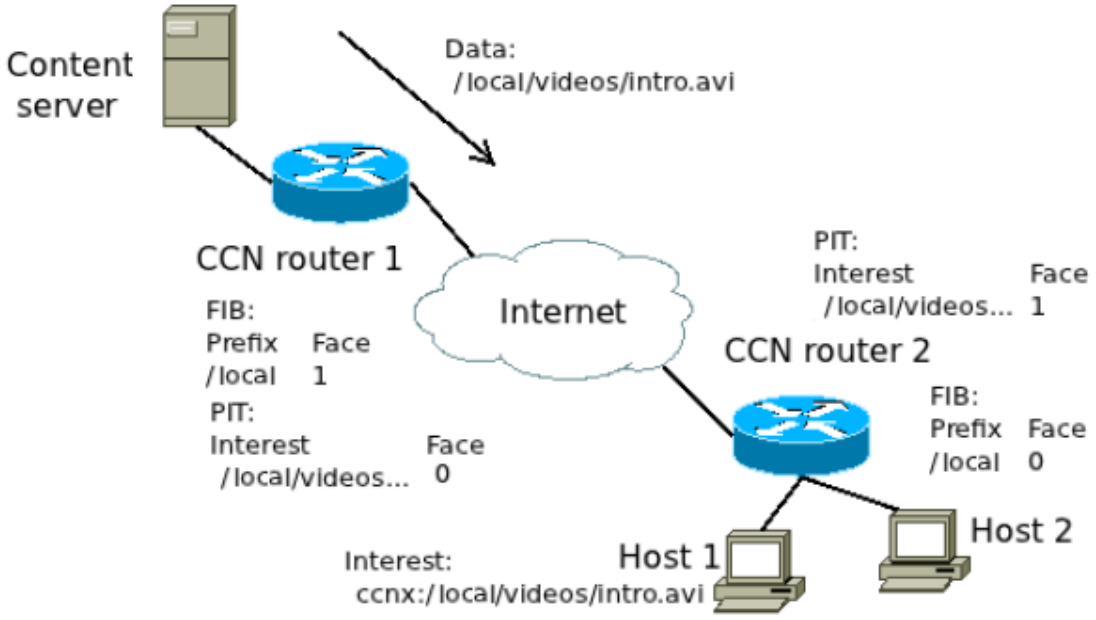


Figura 12 – Esquema minimalista de uma rede CCN (retirado de [Cholez, 2012] e adaptado)

Pontos-chave da comunicação na CCN:

- O conteúdo é assinado pelo seu fornecedor inicial.
- O conteúdo é associado ao seu nome.
- O conteúdo pode ser gerado dinamicamente.

3.5 Fiabilidade do transporte e controlo

Na CCN o trafego de rede é o transporte do conteúdo e não conversação como existe atualmente no TCP/IP, desta forma a disseminação do conteúdo é efetuada tornando o conteúdo popular e desta forma, tal como nos diz o senso comum, tudo o que é popular não provoca uma afluência muito acentuada, pois o conteúdo popular não provoca congestionamento.

Este modelo de pedido/resposta dá o controlo ao utilizador de QoS (congestionamento, prioridade), possibilidade de um controlo mais refinado sobre a qualidade (QoS) do trafego que entra.

Atualmente na internet, problemas que necessitem de QoS são especificamente localizados.

Maioritariamente metade dos problemas é causada por uma série de dependências criadas pelas "filas" (*Queues*). A outra metade é causada por falhas no controlo de receptor base, no estrangulamento das ligações.

Ao contrário do IP, CCN é local, não tem "filas" (*Queues*) e os receptores têm um controlo refinado. Mas isto faz agregação de tráfego e enfrenta controlos específicos de agregação.

O pacote "*Interest*" da CCN é o que controla o fluxo, assim como acontece por analogia com o "*Ack*" no protocolo TCP.

Um grupo de pacotes "*Interest*" pode ser emitido, tal como o equivalente a uma janela de anúncio do TCP.

De acordo com [Martins and Alberti, 2011] , o controlo de fluxo na CCN é baseado numa regra simples, na qual um pacote *"Interest"* pode obter no máximo um pacote "Data". Por outras palavras, pacotes "Data" e pacotes *"Interest"* utilizam uma razão de um para um, o que mantém o controlo de fluxo balanceado. É utilizado no TCP, um controlo de fluxo semelhante, entre o pacote "Data" e o pacote de reconhecimento (Ack - Acknowledge). Os pacotes *"Interest"* têm o mesmo papel do anúncio da janela deslizante no protocolo TCP, ou seja, o tamanho da janela do transmissor pode variar a partir da quantidade de pacotes *"Interest"* enviados [Jacobson et al. 2009]. Esta é a regra básica que garante que o controlo de fluxo seja mantido e torna possível a comunicação de forma mais eficiente entre vários *"hosts"* sobre várias redes heterogéneas.

A proteção e a confiança viajam com o próprio conteúdo na CCN, por isso conclui-se que segurança é nativa do conteúdo, conseguindo evitar deste modo, algumas das vulnerabilidades existentes nas redes IP atualmente. Os nomes na CCN apresentam uma estrutura tipicamente hierárquica. Os mecanismos para resolução de nomes de forma hierárquica, no contexto das redes centradas na informação, podem criar nomes legíveis para o ser humano, além de fornecerem pistas para a resolução de localizadores.

Segundo Van Jacobson [Jacobson, et al., 2009], a relação de confiança com o emissor, na CCN, é conseguida utilizando o prefixo do nome do conteúdo, uma vez que cada pacote enviado contém a assinatura do seu emissor, de forma explícita no seu nome. Conforme explicam [Martins and Alberti, 2011] a assinatura é efetuada em cada pacote através do seu respectivo nome e estas assinaturas são padrões de chaves públicas fornecidas por uma PKI (*Public Key Infrastructure*). A verificação do nome com o conteúdo é efetuada através da respetiva chave privada, pertencente ao consumidor do conteúdo. A confiança na chave de assinatura e a integridade dos dados devem ser fornecidas através de mecanismos adicionais. Por exemplo, através de um certificado baseado em alguma PKI. O paradigma publica/assina (*publish*/*subscribe*) na CCN acontece quando um nó anuncia o nome de determinado conteúdo ao *router*. É desta forma que a informação é publicada. A assinatura é efetuada quando um nó interessado no conteúdo envia um pacote *"Interest"* à procura de um potencial emissor. Os pacotes *"Interest"* são enviados por *broadcast* e a escolha de um dado emissor é

efetuada a partir da análise do maior prefixo correspondente ao nome especificado no pacote *"Interest"*. Os pacotes "Data" seguem o caminho inverso até alcançar o emissor do pacote *"Interest"*.

Numa rede de grandes dimensões, a natureza das conversações ponto-a-ponto do TCP, significa que existem vários pontos entre o emissor e o receptor, onde o congestionamento pode ocorrer da agregação da conversação mesmo que cada conversação possa está a decorrer num fluxo balanceado.

Na CCN o controlo de congestionamento é definido salto a salto, pois é possível definir ou delimitar a procura a determinado número de saltos caso o conteúdo não esteja localmente alojado.

CCN não utiliza filas FIFO entre *links*, em vez disso utiliza uma memória LRU ou seja cache.

Pacotes "*Interests*" que falharem são retransmitidos.

3.6 Mobilidade

As abordagens de redes centradas no conteúdo devem suportar a mobilidade tanto dos objetos de informação, assim como dos *hosts* e redes, conforme apresentado em [Martins and Alberti, 2011].

Dado que a localização do objeto de informação influência semanticamente nos nomes CCN, estas redes não desacoplam a identificação e a localização da informação num dado domínio. O desacoplamento não acontece também para entidades físicas, como *hosts* e redes. Segundo [Jacobson et al. 2009], a CCN endereça conteúdo e não *hosts*, não existindo a necessidade de mapear um endereço a partir de um identificador. Neste caso, quando uma ligação é interrompida devido a um evento de mobilidade, esta ligação pode ser restabelecida logo que exista uma conectividade disponível.

Se não se preocupar com quem está a falar, então não se preocupe se eles mudarem. Se apenas pode pedir algumas pequenas "peças" (bocadinhos) de cada vez, não importa muito se uma cair. Se puder usar qualquer uma e todos os seus *links* simultaneamente, é fácil para a pilha (*Stack*) executar experiências. Se toda a comunicação flui balanceada, saberá exatamente, o que está a trabalhar e se está bem. E tudo isto sem ligação necessária entre a camada 2 e CCN (*named content*). Não há mais problemas de endereçamento de nós, pois o suporte de interface múltipla torna a mobilidade mais fácil. A CCN fornece uma gestão unificada de plataforma de comunicações de dispositivos móveis.

As tecnologias baseadas em CCN, permitem a inscrição através de rede sem fios, partilha e acesso a conteúdo sensível, sem a necessidade da implementação de infraestrutura ou sistemas de suporte em "*Backend*" (bastidores). Utilizando os métodos criptográficos centrados em dados do CCN, os dados podem ser partilhados com segurança e com níveis de privacidade definidos pelo utilizador ou fornecedor.

Os procedimentos de inscrição fortemente utilizados, podem ser aplicados a múltiplos tipos de dispositivos.

O modelo de protocolo agnóstico de comunicação do CCN, disponibiliza a construção de uma plataforma de comunicação a baixo custo para uma variedade de dispositivos heterogéneos.

3.7 Estratégia

A camada "Strategy", segundo [Jacobson et al. 2009], efetua um refinamento da optimização de escolhas dinâmicas para explorar da melhor forma várias ligações em que as condições se podem alterar a qualquer momento.

Os dispositivos hoje em dia têm tipicamente várias interfaces de rede e estão a aumentar a sua mobilidade. Uma vez que o IP está restringido a efetuar encaminhamento em árvore, é difícil conseguir tirar partido das várias interfaces ou adaptar-se às mudanças que possam ocorrer na mobilidade rápida.

Os pacotes da CCN não fazem *loops*, pelo que assim consegue tirar total vantagem de múltiplas interfaces. As conversações na CCN são sobre dados e não conversações para nós, por tanto não precisa de obter ou vincular uma identidade da camada 3 ou seja um endereço IP a uma camada 2 como por exemplo um endereço MAC (Media Access Control). Mesmo quando a ligação muda rapidamente, a CCN consegue sempre trocar dados, assim que seja fisicamente possível de o fazer. Além disso, uma vez que os pacotes "Interst" e "Data" da CCN funcionam aos pares, cada nó é refinado, por prefixo, por performance de informação por interface, para escolher a melhor interface para encaminhar "Interest" através da correspondência parcial do prefixo.

Na CCN as várias interfaces são geridas pela *Strategy Layer* (Camada Estratégia), para cada entrada FIB:

- Ações: `sendToAll`, `sendToBest`, etc.
- *Triggers:* `interestSatisfied`, `interestTimedOut`, `faceDown`, etc.
- Atributos: `broadcastCapable`, `isContentRouter`, etc.

Estas ações, *triggers* e atributos, são colectivamente chamados de camada *"Strategy"* da CCN e o programa numa entrada FIB é a estratégia para obter dados associados com o prefixo da FIB.

É efetuada uma avaliação de desempenho precisa por prefixo e por interface

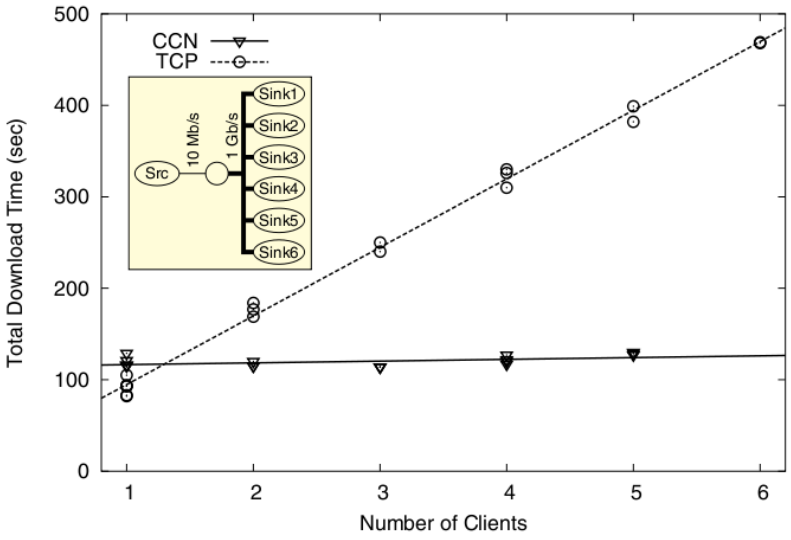


Figura 13 – Tempo total transferência vs. Número clientes (retirado de [Jacobson et al. 2009])

Da figura 13, acima, podemos concluir que enquanto na tradicional comunicação TCP/IP o tempo de total de transferência aumenta proporcionalmente em função do número de clientes de conteúdo, na CCN esse tempo total de transferência permanece praticamente inalterada independentemente do número de clientes que solicitaram esse mesmo conteúdo.

3.8 Transporte

Para Van Jacobson [Jacobson et al. 2009], normalmente um dos problemas com que nos debatemos diariamente é na obtenção de conteúdos de grandes dimensões, ou seja gigabytes de informação, então a questão que se coloca é:

Como poderemos obter coisas grandes?

- Esta questão é um fundamentalismo no transporte.

O transporte na CCN está desenhado para operar no topo de serviços de entrega de pacotes não confiáveis, incluindo ligações altamente dinâmicas da computação móvel e ubíqua. Assim, os pacotes "*Interest*", "Data" ou ambos podem-se perder ou corromper em trânsito, ou os dados solicitados podem estar temporariamente indisponíveis. Para fornecer confiança, entrega resiliente, os pacotes "*Interest*" que não forem satisfeitos durante um período de

tempo razoável devem ser retransmitidos. Ao contrário do TCP, os remetentes CCN são *stateless* e o consumidor final (o aplicativo que originou o interesse inicial) é responsável por reemitir pacotes "*Interest*" insatisfeitos caso ainda queira esses dados. A camada *"Strategy"* do receptor é responsável pela retransmissão numa interface em particular (desde que ele saiba o tempo limite para o nó a montante sobre a interface), bem como escolher qual e quantas das interfaces de comunicação disponíveis para usar para enviar pacotes "*Interest*", quantos pacotes "*Interest*" insatisfeitos devem ser autorizados, a prioridade relativa de diferentes pacotes "*Interest*", etc.

Pacotes de rede subjacentes podem duplicar pacotes e a distribuição multiponto CCN também pode causar a duplicação. Todos os pacotes "Data" duplicados são descartados pelos mecanismos básicos do nó descritos no subcapítulo 3.2. Embora os dados não façam *loop* no CCN, os pacotes "*Interest*" podem fazer um *loop* e fazer parecer com que pareça que existe um pacote "*Interest*" numa interface, quando não existe realmente nenhum pacote "*Interest*". Para detectar e evitar isso, os pacotes "*Interest*" contêm um valor *nonce* aleatório para que os duplicados recebidos de diferentes caminhos sejam descartados.

Os pacotes "*Interest*" da CCN efetuam o mesmo controlo de fluxo e sequenciamento função como pacotes ACK TCP. Uma vez que um nó está habilitado para ver quaisquer dados provenientes de seus interesses, tempo de resposta e a taxa pode ser medida diretamente e usada para determinar de forma dinâmica a melhor maneira de satisfazer os interesses de algum prefixo.

Na CCN o transporte é a árvore do caminho do "*Space Name*" do conteúdo.

4 Implementação em larga escala da CCN

A arquitetura totalmente distribuída da CCN, elimina pontos únicos de falha e estrangulamentos. Afastando-se assim da tradicional arquitetura Cliente-Servidor e focalizando-se no conteúdo como meio de comunicação. Utilizando armazenamento e capacidades de comunicação em dispositivos heterogéneos, proporciona proliferação escalável e propagação para uma plataforma.

Para [Martins and Alberti 2011], um novo projeto de arquitetura de rede que aspira tornar-se uma verdadeira rede mundial deve ser escalável ao extremo, permitindo que triliões de nós e terminais sejam ligados em rede. Além disso, os projetos devem ser capazes de transportar os *exabytes* de informação mensal da Internet atual e futura, e ainda ser economicamente viável. Os *routers* da CCN têm dois desafios ao lidarem com a escala: A gestão do número de prefixos de nomes e o armazenamento de informações de estado por pacote (*Per-Packet State*). As informações de estado são necessárias ao longo do caminho fim a fim, o que representa uma desvantagem do ponto de vista da escalabilidade. Integração com Redes IP.

4.1 Pontos críticos na adoção do CCN

Para podermos adoptar a CCN, segundo [Jacobson et al. 2009], devemos ter em consideração alguns pontos fundamentais, nomeadamente que tudo pode correr sobre CCN e CCN pode correr sobre qualquer coisa. A CCN pode correr como uma camada sobre IP e IP como uma camada sobre CCN. Devemos também ter em conta que existe uma similaridade com IP ou seja a agregação de nome hierárquico com maior procura de correspondência. Além disso os atuais esquemas de encaminhamento que funcionam para IP, também funcionam para CCN.

Apesar de a CCN ser compatível com a infraestrutura existente e possibilitar a implementação incremental, ainda assim torna-se necessário colocar a seguinte questão principal:

Sendo os prefixos do CCN e IP diferentes, serão alguns protocolos de encaminhamento IP compatíveis?

Para obtermos a resposta a esta pergunta vamos apresentar no subcapítulo 4.2 qual a proposta da CNN para solucionar esta questão.

4.2 Encaminhamento Intra-Domínios

Em [Jacobson et al. 2009], Van Jacobson defende que há muitas formas (emergentes) de fazer encaminhamento e em geral são mais fáceis de implementar e trabalhar para CCN do que para IP. Pois na CCN não há "*loops*" de dados, por conseguinte não existem problemas de

convergência. O encaminhamento por ser efetuado para multi-destinos, pois o estado pode ser aproximado, não existindo assim o problema dos falsos positivos.

O modelo de transporte do CCN, de acordo com [Jacobson et al. 2009], faz uma correspondência de encaminhamento e acrescenta segurança. Os protocolos de *Link-state* existentes podem ser utilizados, sem ser modificados, para construir a CCN FIB. Ao contrário do IP *Routing*, a CCN não precisa de topologia, porque o "*Content Model*" suprime os duplicados, portanto nada pode fazer "*loop*", logo a largura de banda utilizada é teoricamente próxima do mínimo. Sem topologia, os interessados em conteúdos encontrarão sempre de forma dinâmica o caminho mais curto para a fonte.

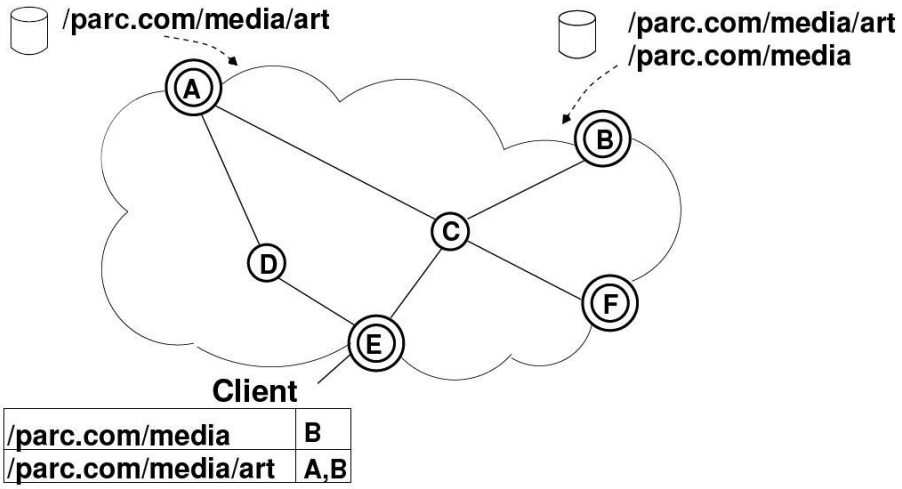


Figura 14 – Encaminhamento de interesses para um domínio com conteúdos multimédia (retirado de [Jacobson et al. 2009])

4.2.1 Ligação local

A comunicação na CCN, conforme exposto em [Jacobson et al. 2009], pode ser roteada utilizando protocolos clássicos: vetor de distância (RIP, EIGRP) e estado do *link* (OSPF, IS-IS) dentro do "*Autonomous Systems*" (AS). A capacidade de encaminhamento da opção TLV (*Type Length Value*) para IS-IS ou OSPF, descreve os recursos ligados. A distribuição de prefixos na CCN, processa-se da seguinte forma, o repositório de Média anuncia a disponibilidade de conteúdos com nome completo, em seguida o r*outer* com suporte CCN, cria um FIB associando o prefixo e a interface e anuncia (*broadcast*) com o prefixo IGP LSA

4.2.2 Múltiplo anúncio de prefixo

A forma como na CCN o múltiplo anúncio de prefixo é efetuado, segundo [Cholez, 2012] é diferente da utilizada na comunicação TCP/IP. As diferenças entre TCP/IP e CCN são que em IP "todos os *hosts* com este prefixo podem ser alcançados por mim" (*loopfree à priori*: única saída), enquanto que, na CCN "parte do conteúdo com este prefixo pode ser alcançado por mim" e além disso, todas as interfaces de uma entrada FIB podem ser contactadas (*loopfree à posteriori*: PIT).

A ligação entre ISPs também se processa de forma diferente do tradicional. Na CCN a implementação é "*Bottom-up*" tendo em conta os incentivos ao ISP, tais como a redução do custo do "*peering*" através de *caching e uma solução* ao nível de domínio prefixos de conteúdo do BGP que é igual ao IGP TLV no *Autonomous System level*.

No entanto há questões por resolver como por exemplo, o problema de ao ligar dois ISP's com suporte CCN através de uma abertura, de que forma poderemos anunciar o conteúdo do cliente ISP?

Esta questão é um dos assuntos que ainda requer investigação e a realização de experiências por forma a encontrar uma solução que resolva esta problemática.

4.3 Limitações (Estado da Arte)

Existem ainda uma série de questões em aberto e a resolução de novos problemas que resultaram da introdução da CCN, conforme a seguir iremos abordar neste capítulo

Existem ainda uma série de problemas e/ou questões em aberto, segundo [Cholez, 2012], tais como, continuam a ser necessárias comunicações ponto-a-ponto para por Ex. a utilização de VOIP. De que forma poderemos perante esta necessidade, enviar dados em CCN? Uma solução poderá passar por ambos os utilizadores estarem acessíveis sob um prefixo único (`/domain/user`) e enviar a informação enviada codificada no nome do pacote *Interest.*

No entanto a prova de conceito da PARC [Jacobson et al. 2009]: é a implementação do VoCCN, pois de acordo com a especificação não há necessidade de existir um *proxy* SIP. O Emissor codifica SIP convida e emite pacostes "*Interest"* para ligar ao Receptor

- `(/domain/bob/encoded-invite-msg)` = publicação a pedido (*on demand publishing*)

Receptor envia resposta SIP no pacote "Data" msg, pois o fluxo de dados RTP, em dados CCN, é uma criação *à priori* de nomes de "*Interest"*

- Nome de Interests: "`/domain/user/call-id/rtp/seq-number`"

No estado da arte atual levantam-se ainda as principais questões:

- Relativamente à arquitetura do CCN e da sua hierarquia de nomes, colocam-se questões sobre problemas a resolver, tais como, de que forma poderemos garantir a unicidade global ou seja como criar nome domínio único de forma global? E relativamente ao processamento de repositórios com poderemos fazer o encaminhamento de dados locais?

- Existe a necessidade de encontrar um algoritmo de encaminhamento eficiente para localizar dados e criar incentivos para fazer com que as pessoas distribuam conteúdos. Além de que será necessário encontrar uma solução eficiente para a gestão de conteúdos.
- Relativamente a encontrar a melhor estratégia de cache, a questão que se coloca é que estratégia de cache, deveremos adoptar tendo em conta a cooperação, o tamanho, a política de substituição, diferenciação de serviço?

- No que concerne à segurança na CCN, existem também algumas questões em aberto, tais como, qual será a forma de impor exclusividade de um nome de conteúdo?

- De que forma poderemos construir a validação entre a integridade e a ligação ao nome e qual o protocolo a utilizar para verificar a integridade dos dados e a ligação ao nome?

- E relativamente à revogação de conteúdo (corrompido), como poderemos tratar esta problemática?

A implementação da CCN nos *routers*, de acordo com [Cholez, 2012], também levanta várias questões, pois não é tão simples como em *routers* IP, o trabalho de verificação de dados a efetuar pelos *routers*. Passarão a existir mais estados na tabela de encaminhamento e monitorização e entradas mais complexas (os nomes vs endereços IP).

- Que tipos de dispositivos de armazenamento, poderemos ter associados aos *routers* para a cache de conteúdos?

- De que forma poderemos mitigar o aparecimento de potenciais novos ataques de DoS?

- Na implementação da CCN nos routers há ainda que ter em conta relativamente ao encaminhamento o aumento do custo. Poderá ser esta implementação considerada um retrocesso em comparação com *routers* IP?

Todas estas questões em aberto ainda continuam sem resposta, porque carecem de muita investigação por parte da comunidade científica. Não existe ainda matéria suficiente para se

tirarem conclusões ou conhecimentos sobre como contornar ou resolver estes problemas que resultam da introdução do paradigma da CCN.

4.4 Problemas de Escalabilidade

Segundo [Cholez, 2012], a arquitetura CCN também apresenta duas questões a resolver, sendo uma delas a problemática da escalabilidade de encaminhamento baseado em nomes e a outra a mudança do espaço de endereço de mil milhões de IP's para pelo menos um trilião de nomes conteúdo.

A escalabilidade irá obrigar a os r*outers* CCN tenham que ser mais potentes, pois terão que ser s*tateful para guardarem o* trilho dos pacotes *Interests*, terão que ser capazes de verificar assinaturas e de armazenar em cache o conteúdo.

4.5 Avaliação de Desempenho

Com a introdução deste paradigma, [Jacobson et al. 2009], torna-se necessário comparar quais as suas vantagens e desvantagens perante a atual arquitetura TCP/IP. A seguir apresentam-se algumas das constatações retiradas de experiências que se efetuaram, nomeadamente nos que diz respeito à utilização da ligação e *Overhead* provocado, o seu comportamento na divulgação de dados e no suporte multi-interface.

Baseado no protótipo do CCNx concluímos que a transferência local de ficheiros grandes, em TCP é de 90% utilização de ligação de banda, enquanto que o CCN apenas consegue utilizar 68%, pois 22% são para cabeçalho CCN + encapsulamento IP/UDP). Portanto conclui-se que a CCN é cerca de 3 vezes mais lenta para alcançar o rendimento máximo estável, mas ainda pode ser otimizado.

Relativamente ao *Download* de uma página Web, o CCN apresenta menos *overhead* do que o HTTP, pois o HTTP atinge valores na ordem dos 15%, enquanto que o CCN fica-se pelos 7% e também supera o HTTPS que atinge um *overhead* na ordem dos 25%, ao passo que a CCN ronda os 8%.

A distribuição multiclientes no CCN é muito mais eficiente, pois baseia-se no armazenamento em cache de conteúdos evitando desta forma as tradicionais ligações TCP cliente-servidor.

Desta forma se aumentarmos o número de clientes CCN, o tempo *download mantêm-se* constante, enquanto que no TCP tem um crescimento linear.

Na utilização de VOIP entre dois *Hosts*, cada um ligado a duas redes, como a camada *"Strategy"* verifica periodicamente as interfaces e usa a mais rápido a responder. Se uma interface estiver desligada, então acontecerá um *failover* automático.

5 Gestão / Segurança

A comunicação de dispositivos "*Plug and Play*" é um dos benefícios específicos da tecnologia CCN, através da inscrição básica e de mecanismos de dispositivos para um emparelhamento intuitivo, configuração e configuração de múltiplos dispositivos. Mas todas estas operações pressupõem uma confiança e segurança.

O modelo de segurança centrado nos dados, segundo [Jacobson et al. 2009], proporciona a partilha segura de conteúdos sensíveis. O modelo de segurança do CCN é centrado na segurança do próprio conteúdo, em oposição aos pontos de extremidade independentemente de onde viajam os pacotes através da rede, o conteúdo está protegido. Esta proteção fundamental não impede a utilização adicional de medidas de segurança. Por ex. "Transmissão cifrada entre dois pontos, para satisfazer requisitos regulamentares".

5.1 Segurança na CCN

Atualmente para se obter um conteúdo na Internet, segundo [Ribeiro et al., 2012], seja uma página *Web*, sejam dados multimédia, é necessário primeiro determinar o endereço IP do repositório de armazenamento deste conteúdo. Em geral, apenas o nome desse repositório é conhecido e, portanto, são necessários mecanismos de resolução de nomes, como o DNS (*Domain Name System*). Após isto, o cliente estabelece uma comunicação ponto-a-ponto com esse repositório e requisita o conteúdo através de algum protocolo de aplicação conhecido.

Caso o cliente deseje ter a certeza que o conteúdo recebido é realmente aquele que deseja obter, deve garantir que o servidor DNS e o repositório do conteúdo sejam confiáveis, que não tenham sido comprometidos e que o canal por onde o conteúdo foi transmitido é seguro. Desta forma, a tentativa de garantir a segurança de uma simples página *web*, por exemplo, implica na necessidade de garantir a segurança de diversos componentes da infraestrutura da rede. Essa complexidade é reflexo da falta de planeamento de segurança durante a fase de projeto da Internet. Na CCN esta problemática não se verifica, porque a CCN adota a segurança como um dos seus pilares e fornece mecanismos que simplificam a garantia dos requisitos de segurança como a autenticidade, integridade, confidencialidade e disponibilidade dos dados.

Apresentamos a seguir os mecanismos usados pela CCN para fornecer tais requisitos.

5.1.1 Segurança do Conteúdo

Tendo em conta que na CCN, um conteúdo pode estar guardado tanto na fonte como em cache num qualquer nó da rede e como esses repositórios podem estar sob o controlo de utilizadores mal-intencionados, de acordo com [Ribeiro et al., 2012] a segurança do conteúdo deve ser independente da sua localização física. Esta característica facilita a implementação de modelos de segurança, pois agora para garantir a segurança de um conteúdo obtido por um utilizador, já não é necessário confiar em toda a infraestrutura envolvida na sua obtenção. Para isso, deve-se fornecer ao utilizador condições para verificar se o conteúdo recebido é uma cópia fiel do conteúdo divulgado pelo emissor (integridade ou validade), quem é o emissor do conteúdo recebido (autenticidade ou proveniência) e se o conteúdo recebido condiz semanticamente com a requisição efetuada pelo utilizador (relevância).

A melhor forma de atribuir nomes aos conteúdos de modo a garantir sua integridade, proveniência e relevância são ainda temas fortemente debatidos. Apesar dos envolvidos diretamente no projeto da arquitetura CCN adotarem o modelo de nomenclatura hierárquico e afirmarem que este é o mais seguro [Smetters and Jacobson, 2009], outros, tais como [Ghodsi et al., 2011], argumentam a favor da utilização dos nomes auto certificados.

Relativamente à estratégia de segurança, de acordo com [Ribeiro et al., 2012] a CCN implementa segurança, é independente da infraestrutura, e fácil de gerir a comunicação entre dispositivos. Com a sua forte segurança e fácil associação de dispositivos e configuração, é a arquitetura ideal, uma vez que necessita de menos configurações de infraestrutura, porque não assenta no tradicional modelo Cliente-Servidor. Algumas das suas características são a utilização de uma assinatura digital no núcleo de segurança da CCN, criptografia para fornecer privacidade e confiar no conteúdo e não na forma como o obtemos. Como os pedidos são baseados no encaminhamento, não há ataques de DoS clássico. A autenticação do conteúdo implica a autenticação de ligação entre o nome e o conteúdo, que será embutido em cada pacote “Data” CCN com uma assinatura (nome, conteúdo, *SignInfo*) que inclui: criptográfico *digest* ou impressão digital do editor de chave, chave ou localização chave. Assim o conteúdo pode ser autenticado por cada nó através de assinaturas de chave pública.

Tendo em conta a segurança vs performance existe a necessidade de se encontrar um algoritmo de assinatura diferente disponível.

5.2 Gestão da confiança

Como foi referenciado anteriormente, segundo [Ribeiro et al., 2012] os nomes auto certificados verificados por chave utilizam a *hash* da chave pública do emissor na sua formação. Desta forma, implicitamente, ao especificar o nome do conteúdo na requisição, os utilizadores estão a especificar também qual deve ser o emissor desse conteúdo. No entanto, nada impede que emissores mal-intencionados utilizem a chave pública de outro emissor legítimo para assinar e nomear seus conteúdos. Para evitar a fraude, os conteúdos também são assinados utilizando a chave privada, par da chave pública usada na atribuição do nome. Assim, para conferir a autenticidade dos conteúdos é necessário seguir algumas etapas.

Em primeiro lugar temos que obter a chave pública do emissor, sendo que a esta pode estar presente no conteúdo como meta-dado. Depois vamos verificar se a chave pública obtida é realmente aquela que foi utilizada para formar o nome do conteúdo. Para isso, basta calcular a *hash* da chave pública e comparar o resultado com o nome do conteúdo. Após isto, utiliza-se a chave pública obtida para verificar a assinatura do conteúdo. Caso essa verificação seja concluída com sucesso, então podemos considerar que o conteúdo é íntegro e autêntico.

De forma a evitar que os utilizadores possam obter conteúdos inválidos, essa verificação poderia ser efetuada no momento da publicação do conteúdo, assim a rede poderia descartar essas publicações de conteúdos inválidos. Neste cenário, os utilizadores passariam a confiar em todos os conteúdos recebidos, já que eles confiariam que estes seriam sempre íntegros e seriam publicados pelo emissor pretendido.

Na hierarquia de nomes adotada pela CCN, os nomes também identificam o emissor do conteúdo, não através da chave pública mas sim pelo próprio nome do emissor. Por exemplo, seja "`/isep.ipp.pt/entidade/apresentacao.mp4`" um nome de conteúdo. Nesse caso podemos supor que, o emissor do conteúdo em questão tem o nome "`/isep.ipp.pt`". Quando o utilizador solicitar este conteúdo, ele irá esperar obter como resposta um conteúdo publicado pelo ISEP e não por outra entidade. Dado que esses nomes de conteúdo não possuem qualquer relação direta ou indireta com o próprio conteúdo, de acordo com [Ribeiro et al., 2012], a CCN utiliza a assinatura do mapeamento entre o conteúdo e seu nome para esse efeito. Quando o conteúdo é recebido, o receptor deve obter a chave pública do emissor e utilizá-la para verificar a assinatura presente naquele conteúdo. No entanto, caso a verificação seja positiva, tudo que o receptor saberá é que a chave privada que faz par com a chave pública utilizada na verificação foi realmente aquela que assinou o mapeamento entre o conteúdo e seu nome. Mas, como não existe relação entre o nome do conteúdo e a chave pública utilizada na verificação, o receptor não consegue garantir que esta chave pública pertence efetivamente ao emissor pretendido.

Através de um mapeamento seguro entre o nome do emissor e a sua chave pública, pode-se criar uma relação indireta entre o nome e a chave pública do emissor. Tipicamente esse mapeamento seguro é representado por um certificado digital. Consequentemente, a hierarquia de nomes da CCN implica na utilização de um sistema de infraestrutura de chaves

públicas. Dentre os possíveis sistemas, conforme exposto em [Zhang et al. 2010], tais como o PGP, X.509 e SDSI/SPKI, aparentemente o mais indicado seria o SDSI/SPKI, já que os mesmos utilizam o conceito de espaço de nomes local, utilizado para nomear chaves públicas.

Desta forma, os utilizadores da CCN poderiam solicitar a chave pública de um emissor através do seu nome, da mesma forma que qualquer outro tipo de conteúdo é requisitado.

Uma alternativa ao SDSI/SPKI é o mecanismo de estabelecimento de confiança em chaves públicas de emissores proposto por [Pournaghshband and Natarajan, 2011], de forma a possibilitar que aplicações possam determinar quais as chaves que são confiáveis. Neste mecanismo os consumidores devem solicitar recomendações sobre a chave pública de emissores a todos os membros da sua comunidade de confiança. A formação de uma comunidade de confiança é conseguida por pessoas que o utilizador conhece através de algum relacionamento no mundo real. Assim, uma política local, definida pelo próprio consumidor, é utilizada para avaliar a consistência das respostas e consequentemente, a confiança depositada na chave de um determinado emissor.

Atualmente, os mecanismos fornecidos pela CCN, de acordo com [Smetters and Jacobson, 2009], possibilitam que um utilizador possa verificar se um conteúdo recebido é válido e se o mesmo foi publicado pelo emissor pretendido. No entanto, para solicitar um conteúdo, o utilizador precisa confiar, *à priori*, no emissor cujo nome consta no nome do conteúdo. Por exemplo, é fácil confiar que os vídeos encontrados no domínio `/isep.ipp.pt` foram publicados pelo ISEP – Instituto Superior de Engenharia do Porto, do Instituto Politécnico do Porto, já que é sabido, através de meios externos, que tal domínio realmente pertence ao ISEP. Por conseguinte, para confiar num emissor completamente desconhecido a CCN propõe um mecanismo conhecido como segurança baseada em evidências, que visa auxiliar os utilizadores no processo de estabelecimento de confiança nos emissores.

De acordo com [Ribeiro et al., 2012], para se entender o conceito de segurança baseada em evidência, imaginemos uma situação em que um emissor A decide publicar uma página *Web* na rede e o emissor A também têm uma relação de confiança com um emissor B, que por sua vez, publica um determinado vídeo popular. Vamos supor ainda que um determinado utilizador confia no emissor A, mas que pretende receber o referido vídeo popular cujo emissor seja B. Este utilizador pode confiar no emissor B e por consequência confiar no vídeo publicado por ele, pois o emissor A disponibiliza um link na página *Web* apontando para o vídeo publicado pelo emissor B. Deste modo, como o utilizador em questão confia no emissor A, ele pode aceder a esse link e obter o vídeo publicado por B. Para garantir que realmente foi o emissor A quem criou o link para o vídeo, esse link necessita de estar assinado pelo emissor A. No contexto da CCN, um link seguro é um mapeamento N → (N, H (P)), onde N é o nome do link, N é o nome do conteúdo para o qual o link aponta e H (P) é a *hash* da chave pública do emissor do conteúdo N . Quando um emissor publica o *link* N, ele está a referir na realidade, que para ele, o conteúdo referente ao nome N é aquele que o emissor P nomeou como N. Se cada conteúdo obtido possuir *links* como os descritos, os utilizadores poderiam interpretá-los como evidências de que os conteúdos apontados são seguros.

Ainda segundo [Ribeiro et al., 2012], um outro exemplo de segurança baseada em evidências, supondo que um emissor honesto publicou um conteúdo popular com o nome N. Para enganar os utilizadores, um emissor mal-intencionado publicou um conteúdo diferente com o mesmo nome N. Além disso, se os utilizadores utilizarem as evidências, ou seja os *links,* existentes em conteúdos confiáveis obtidos anteriormente, poderão verificar que a maioria dos emissores confiáveis indica, através dos *links*, que para ser confiável, o conteúdo de nome N tem que ter sido publicado por um emissor honesto. Apesar de interessante, essa proposta cria relações de confiança estáticas, já que os *links* estão presentes nos conteúdos e esses, uma vez obtidos, não são mais alterados pelos seus emissores. Então, se um emissor confiável passa a ser malicioso, os utilizadores podem ser levados a confiar erradamente nos mesmos, já que as evidências presentes nos conteúdos obtidos por eles ainda apontam nesse sentido. Atendendo que é importante um mecanismo de revogação de certificados digitais para os sistemas PKI, o processo de revogação de um *link* também é importante para o modelo de segurança baseada em evidências. Contudo, tal modelo não prevê nenhum mecanismo para esse efeito.

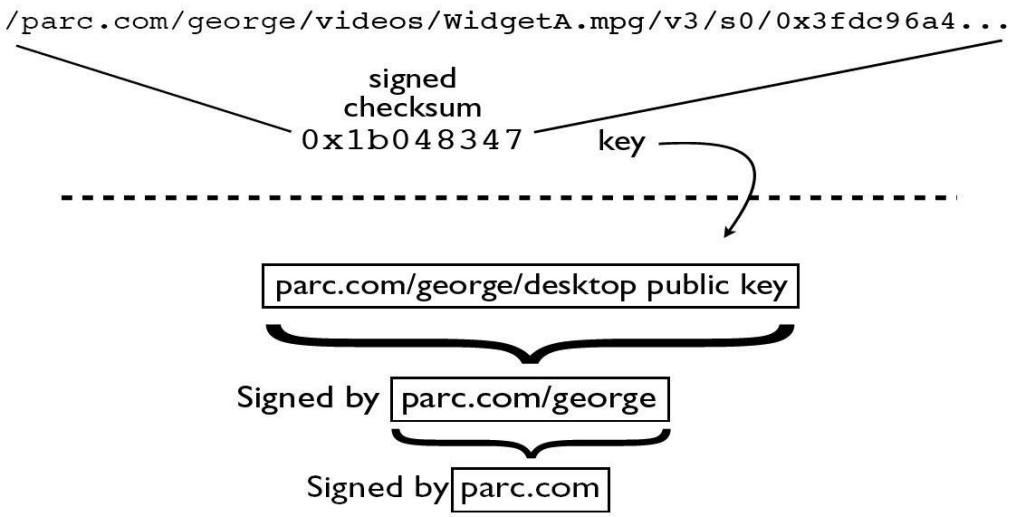


Figura 15 – Gestão da confiança, associando nomes CCN com o emissor das chaves (retirado de [Jacobson et al. 2009])

5.2.1 Controlo do Acesso a Conteúdos

Tradicionalmente, sistemas como o *Kerberos* [Neuman et al. 1993] são utilizados para autenticar e fornecer aos utilizadores credenciais de acesso a dados e serviços. Nestes sistemas, normalmente um servidor de autenticação verifica a identidade dos utilizadores através de um processo de *login* e palavra-passe. Caso essa identidade seja comprovada, o utilizador recebe credenciais para aceder os dados ou recursos que lhe forem permitidos. Para que esse

mecanismo funcione, é preciso que um servidor intercepte os pedidos de acesso a dados efetuados pelos utilizadores e avalie se deve concedê-las ou não. Atendendo a que os conteúdos, na CCN, são armazenados em qualquer nó da rede, inclusive em *routers*, e os utilizadores solicitam-nos através do nome do conteúdo e não diretamente a um servidor específico. Para [Ribeiro et al., 2012], neste cenário seria muito difícil, senão mesmo impossível, montar um sistema como o *Kerberos* na CCN. Independentemente da localização do mesmo na rede, o controlo de acesso aos conteúdos deve então ser implementado a nível do conteúdo propriamente dito. Para esse objetivo ser possível de alcançar, é necessário empregar a criptografia simétrica desses conteúdos de forma que, somente o utilizador ou grupo de utilizadores que conheçam a chave possam aceder ao conteúdo. Para distribuir uma chave simétrica, o emissor do conteúdo cujo acesso é controlado, poderia encriptá-la utilizando a chave pública do utilizador no qual o acesso ao conteúdo será permitido. Depois, essa chave simétrica encriptada poderia ser publicada na rede, através de um nome específico, da mesma forma que um conteúdo qualquer é publicado na CCN. O utilizador poderia então recuperar essa chave simétrica através do nome e utilizar sua chave privada para desencriptá-la.

As propriedades fundamentais do CCN, segundo [Cholez, 2012], não permitem apenas que as aplicações e serviços executem em dispositivos móveis, mas sobre tudo destacar a sua utilização na mobilidade e com recursos limitados.

Esta nova plataforma permite que os dados sejam personalizados para garantir privacidade e controlo de acesso, permitindo vistas diferentes dos dados para grupos diferentes ou categorias, através dos métodos de encriptação do CCN.

A informação pode ser partilhada através de *links* pré-estabelecidos ou links oportunamente criados.

Uma estratégia de confiança, de acordo com [Cholez, 2012], pode passar por:

- Uma entrega do tipo P2P, mas a segurança "*end-to-end*": confiança no emissor.
- Simular o certificado: organização como nome do conteúdo, a chave pública como dados.
- O conteúdo pode refletir confiança certificando-se um ao outro (relação de confiança, reduzir a sobrecarga na gestão da chave).

5.3 Gestão de chaves

A gestão de chaves na CCN, segundo [Cholez, 2012], passa por uma segurança baseada em evidências do tipo "*Open*", a proveniência dos dados (rastreabilidade) pode ser verificada. Assim, deve ser evitado um sistema de certificação hierárquica por causa das questões relacionadas com a confiança em entidades certificadoras, como por ex: VeriSign. Pois a confiança deverá ser individual em subárvores sem nenhuma autoridade central.

Relativamente à proteção de conteúdo, segundo [Cholez, 2012], a CCN utiliza a encriptação do conteúdo logo não são necessários servidores de confiança ou diretórios. Assim sendo não há controlo de acesso ao nível de diretório, tendo em conta que a proteção de conteúdo é efetuada através de criptografia e PKI. A gestão de chaves na CCN está facilitada porque a chave pública é facilmente obtida como dados CCN e a publicação de uma chave gera certificação. Logo podemos concluir que o conteúdo pode refletir confiança, certificando-se um ao outro (ex: página da Web e seus links, imagens, etc.).

Neste contexto pode facilmente surgir a pergunta: Que PKI para CCN?

E a resposta é que, não existe a necessidade de existir uma, pois em SDSI/SPKI: as chaves são mapeadas para identidades via "*namespaces*" = nomes CCN, logo não existe nenhuma fonte de confiança.

5.4 Segurança de Rede

5.4.1 Ataques de rede mitigados

Segundo [Cholez, 2012], podemos afirmar que o projeto CCN melhora a proteção do Hospedeiro, pois na CCN a comunicação não pode falar diretamente para o hospedeiro, desde logo é difícil o envio de pacotes maliciosos. A proteção de conteúdo, pois o conteúdo é autenticado, então será difícil de falsificar. A Política de conteúdo, temos por exemplo o conteúdo *firewall* baseado em nome do prefixo ou assinatura. E temos a agregação de dados na rede e desta forma não existe ataque DDoS.

No entanto temos a salientar algumas questões por resolver:

- DoS: ataque por inundação de interesse "*Interest*".
- Possível detecção no caminho: taxa de emissão de interesse vs dados transmitidos.

Será necessário continuar a investigação neste sentido por forma a mitigar estas novas potencialidades de ataque. Não existindo ainda há data desta dissertação qualquer trabalho de investigação divulgado que apresente alguma proposta sobre uma possível resolução.

5.4.2 Nomenclatura segura do conteúdo

Contrariamente ao que sucede com os nomes auto certificados, segundo [Ribeiro et al., 2012], o esquema de hierarquia de nomes utilizado pela CCN é legível para o ser humano e possibilita aos utilizadores um grande poder de expressão. No entanto, tais nomes não fornecem qualquer tipo de informações para podermos verificar a integridade ou proveniência dos conteúdos. De acordo com o triângulo de Zooko [Wilcox-O'Hearn 2003], os nomes só podem ter, ao mesmo tempo, no máximo duas das seguintes propriedades: unicidade global, segurança e inteligibilidade, conforme ilustrado na Figura 16. Ou seja, contrariamente ao que acontece com os nomes auto certificados, no esquema de nomes da CCN, os nomes são globalmente únicos, inseguros e inteligíveis.

Figura 16 – Triângulo de Zooko (adaptado de [Ribeiro et al., 2012])

A CCN escolheu duas propriedades para os nomes: legível e seguro.

Assim, facilmente podemos concluir que, a flexibilidade da CCN em gerar nomes inteligíveis e únicos confere a completa falta de segurança dos mesmos. Então, para simplificar este cenário, a CCN divide o processo de requisição e obtenção de conteúdos nas funções de identificador, localizador e autenticador de conteúdo.

Segundo [Ribeiro et al., 2012], a função de identificador possibilita ao ser humano especificar os conteúdos de seu interesse. A função de localizar possibilita que a rede localize um determinado conteúdo, onde quer que este se encontre. Já a função de autenticar verifica se o conteúdo recebido é válido ou não. O esquema de nomes auto certificado impõe que os nomes cumpram todas essas funções ao mesmo tempo, gerando os problemas mencionados

anteriormente. Na abordagem da CCN, os nomes desempenham as funções de identificar e localizar de conteúdos. A função de autenticar é delegada à assinatura digital do mapeamento entre o conteúdo e o seu nome. Quando um emissor deseja publicar um determinado conteúdo, ele deverá efetuá-lo, utilizando a forma da tripla M (N,P,C) = (N,C, Sign P (N,C)), onde N é o nome do conteúdo, C é o próprio conteúdo e Sign P (N,C) é a assinatura do emissor, efetuada sob o mapeamento entre o nome e o conteúdo. Então, um utilizador deve solicitar o conteúdo através de seu nome N obtendo o conteúdo C e a assinatura Sign P (N,C). Desta forma, o utilizador necessita de obter a chave pública P do emissor para verificar a assinatura, para certificar-se de que o conteúdo é íntegro e que foi efetivamente publicado pelo emissor detentor da chave pública P. Um benefício obtido com este modelo de nomes é o de possibilitar que o conteúdo seja guardado em qualquer nó da rede. A obtenção de M (N,P,C) possibilita ao utilizador a verificação da integridade e autenticidade do conteúdo C e do mapeamento entre o nome N e conteúdo C. O conteúdo pode ser guardado tanto no emissor bem como em quaisquer caches da rede, quer seja confiável ou não. Atendendo a que as caches são essenciais para a obtenção eficiente de conteúdos, esta característica é fundamentalmente importante para a CCN.

A CCN na sua abordagem não faz suposições a respeito do formato do nome N em questão e isso é um dos seus benefícios. As várias aplicações podem ter restrições diferentes, no que diz respeito ao formato dos nomes utilizados. Assim, é possível que os nomes tenham qualquer formato, atendendo às necessidades das aplicações, mas garantindo sempre a segurança do mapeamento entre os conteúdos e seus nomes, o valor e o significado atribuídos pelo padrão de nomes utilizado.

Em suma, assinar o mapeamento entre o conteúdo e seu nome, na prática é o equivalente a emitir um certificado digital para o conteúdo, assegurando que este efetivamente é íntegro e contêm o nome especificado. Este tipo de abordagem corresponde à mudança de um modelo no qual os nomes necessitam sempre de transportar consigo informações de autenticação para um modelo onde o receptor do conteúdo é quem decide se este é aceitável ou não. Contudo, tal modelo é mais vulnerável a ataques de negação de serviço, dado que o utilizador necessita, em primeiro lugar de obter o conteúdo e só depois decidir se o este é ou não válido.

Deste modo, adversários podem tentar fazer com que o utilizador sempre receba conteúdos inválidos [Ghodsi et al. 2011]. Uma outra abordagem seria exigir que os *routers* efetuassem a verificação da validade dos conteúdos antes de os armazenarem em cache. A hierarquia de nomes da CCN exige que exista uma cadeia de confiança que autentique o mapeamento entre a chave pública de um emissor e seu nome no mundo real. Mas, exigir que cada *router* percorra essa cadeia de confiança para verificar a autenticidade de uma chave pública antes de a poder utilizar para verificar a assinatura de cada conteúdo, pode ser mostrar-se demasiado excessivo. Consequentemente o excesso de processamento nos *routers* pode torna-los muito vulneráveis a ataques de negação de serviço. Pode ser consultado [Ribeiro et al., 2012], para informações mais detalhadas sobre este tema.

Em suma temos a reter que na CCN, a nomenclatura segura assenta na autenticação de ligação entre nomes e conteúdo. Não existe uma restrição na escolha de nomes e o emissor é quem certifica o nome para o conteúdo. Os dados devem ser fornecidos juntamente com a chave do emissor e o mapeamento de prova.

O sistema de nomenclatura segura da CCN é descentralizado e único. O mapeamento de nomes CCN são legíveis e seguros. Os nomes CCN são de localização de conteúdo e devem ser globalmente únicos.

5.4.3 Monitorização da CCN

A monitorização da CCN, segundo [Cholez, 2012], tem como propósito recolher informações e *status* de nós CCN, detecção de anomalias, responsabilização e a revogação de conteúdo. Para isso será necessário ultrapassar desafios, tais como adaptar o SNMP para nós CCN. Definir as informações relevantes para a monitorização do CCN. Será necessário definir a arquitetura relevante (CCN pronto) para a recolha de informações e permitir a colaboração entre os nós CCN.

5.4.4 Ataques a tabelas CCN

Segundo [Ribeiro et al., 2012], apesar de aumentar a privacidade dos utilizadores, derivada da ausência de informações de origem e destino nos pacotes, a CCN apresenta também várias vulnerabilidades. A arquitetura CCN introduz desafios e vulnerabilidades diferentes dos enfrentados atualmente na arquitetura TCP/IP. Atendendo a que um nome de conteúdo é utilizado como parâmetro para sua obtenção, este deverá ser semanticamente relacionado com próprio conteúdo. Assim, se um adversário capturar uma solicitação de um conteúdo, ele será capaz de o identificar, mesmo sem o obter. No que concerne aos ataques de negação de serviço, apesar das técnicas utilizadas na Internet atualmente se mostrarem ineficazes contra a CCN, elas poderão ser adaptadas com o objetivo de explorar diversas características desta arquitetura.

Por fim, a realização de cache de conteúdos por todos os nós da rede permite uma maior disponibilidade e eficiência na obtenção dos mesmos.

Há ainda, de acordo com [Cholez, 2012], a salientar alguns pontos fracos nos diferentes componentes da CCN:

FIB

- Anuncia domínios confiáveis, nomes não-agregáveis.
- Anuncia conteúdo existente sem o ter.

- Anuncia conteúdo que não existe.

PIT

- Fornecedores de Inundações de conteúdos ou *routers* com *Interests* falsos.
- Fornecedor de DoS de conteúdos? Rastreamento de consulta é impossível?
- *Interest* legítimo descartado no *router*? Criação de *loops* de encaminhamento?

CS

Armazenamento de conteúdo:

- Diminui a eficiência de armazenamento em cache, descarregando conteúdo impopular.
- Espiar vizinhos de rede sondando a cache.

5.4.5 Novo DoS

Existem diversos serviços atualmente na *Internet* que fazem uso de caches para aumentar sua eficiência. Por conseguinte, os conteúdos que são solicitados com maior frequência são guardados em dispositivos intermediários, de forma garantir que o acesso a esses conteúdos se faça de forma mais rápida. Os navegadores *Web*, por exemplo, armazenam as páginas solicitadas recentemente no disco local das máquinas. Assim, as futuras solicitações para estas páginas são imediatamente atendidas, sem a necessidade de serem obtidas do servidor *Web* original. Para [Ribeiro et al., 2012], a implementação de caches na rede, pode aumentar a eficiência na obtenção de conteúdos, mas também pode introduzir problemas de violação de privacidade. Um utilizador mal-intencionado que consiga obter acesso aos dados guardados na cache poderá saber que conteúdos, os utilizadores associados a esta cache estão a solicitar. Dado que os pacotes IP's contém a informação sobre o endereço de origem dos dados, um adversário poderia saber "quem", e "o que" está a ser solicitado. Além disso, a utilização da técnica de tentar extrair rastos de comunicação das caches para inferir informações sobre seus utilizadores e é por isso conhecida como cache *snooping*.

Maioritariamente os ataques de cache *snooping*, segundo [Felten and Schneider 2000], são baseados no tempo de resposta da solicitação para um determinado conteúdo.

Então, segundo [Ribeiro et al., 2012], supondo que é objetivo de um adversário saber se o utilizador A acedeu ao *site* de um utilizador B. Caso o utilizador A tenha acedido à página do utilizador B, o adversário saberá que o utilizador A provavelmente acedeu a um documento principal chamado index.html.

Portanto, o adversário efetua uma solicitação ao index.html da página *Web* do utilizador B, utilizando para o efeito um script e calcula o tempo que o conteúdo demora a ser obtido. O script é introduzido na página *Web* do adversário. O utilizador A é então levado a aceder ao script presente na página do adversário, por exemplo através de um ataque de *phishing*.

O script é executado, nesse momento e informa o adversário sobre o tempo que o utilizador A demora a receber o ficheiro index.html do utilizador B. Caso este ficheiro já esteja na cache do navegador do utilizador A, este tempo deverá ser curto, informando que o utilizador A acedeu à página *Web* do utilizador B.

A CCN tem definido como um dos seus principais pilares, garantir a obtenção eficiente e aumento da disponibilidade de conteúdos e expandir a utilização de caches para todos os *routers* e assim é natural imaginar que se a "*cache snooping"* é um problema para a Internet atual, o problema ainda será ainda maior no contexto das implementações CCN. Contrariamente às redes IP, a CCN não regista informações sobre a origem das solicitações. Consequentemente, isto implica fornecer uma maior garantia da privacidade dos utilizadores, porque mesmo que um adversário explore a cache para obter informações sobre os conteúdos presentes nela, não é possível obter nenhuma informação sobre quem os pediu. Por conseguinte, a hierarquia de caches produzida pela CCN faz com que um número reduzido de utilizadores esteja associado a uma determinada cache e isto aumenta a exposição dos utilizadores.

Há também a considerar algumas questões relativas à segurança, como p. ex: DoS de *router* através de assinaturas pesadas, obtenção de chave lenta. A Revogação de chaves corrompidas, indesejadas ou de conteúdo malicioso e robustez da criptografia corrente no futuro. Em [Ribeiro et al., 2012] podem ser consultadas informações mais detalhadas sobre esta problemática.

6 Protótipo CCNx

6.1 Apresentação

A iniciativa continua a ganhar (sucesso) impulso com uma versão de código-fonte aberto (*open source*) lançada em C e Java, uma implementação para Android, parcerias comerciais com parceiros industriais importantes, financiamento (fundos) governamental para um projeto multi-institucional, bem como a centralização da comunidade CCN com Sede na PARC (*Palo Alto Research Center*).

O CCNx é um conjunto essencial de ferramentas e funcionalidades para a podermos implementar uma rede com uma arquitetura assente na CCN, fornecendo suporte a questões como o encaminhamento, comunicação, gestão de chaves, gestão da confiança, assinatura e encriptação de conteúdos.

6.2 Características

CCNx utiliza dois tipos de mensagens para transferir dados do utilizador: uma mensagem de pedido designada por interesse "*Interest*" e uma mensagem de resposta encapsulada com o resultado, designada por uma mensagem de resposta de objeto de conteúdo "*Content Object*". CCNx usa um formato *Type-Length-Value* (TLV) para codificar todas as mensagens ao longo da cablagem. Este é um novo formato de pacote geral baseado num cabeçalho fixo, um conjunto de cabeçalhos opcionais, e uma mensagem de protocolo CCNx.

Os protocolos incluem também dois operadores: `Equals` e `ComputeContentObjectHash`. Estes dois operadores trabalham nos campos de uma mensagem de interesse "*Interest*" para determinar se corresponde ou não a um objeto de conteúdo. Uma mensagem de interesse "*Interest*" recebe, no máximo, uma mensagem de objeto de conteúdo, para que haja equilíbrio de fluxo de mensagens. Uma mensagem de interesse "*Interest*" pode ser considerada um tipo de controlo de fluxo, de como um cliente pode emitir uma série de mensagens de interesse "*Interest*" para diferentes objetos de conteúdo. O cliente abre uma janela para as respostas. Ao contrário da maioria dos protocolos de rede de dados, a janela é medida em mensagens e não em bytes.

O protocolo central funciona da seguinte forma: os consumidores emitem um pedido de conteúdo através do envio de uma mensagem de interesse com o nome do conteúdo desejado. A rede encaminha o interesse através de um conjunto de encaminhadores com base no nome usando a correspondência de prefixo mais longa. O interesse deixa o estado, uma vez que atravessa a rede. Este estado é armazenado na Tabela de Interesses Pendentes (PIT). Quando for encontrada uma correspondência, ou seja quando um interesse

corresponde a um objeto de conteúdo, o conteúdo é enviado de volta no caminho inverso do interesse, seguindo o estado PIT criado pelo interesse. Como o conteúdo é auto-identificável, através do nome e da ligação de segurança, qualquer objeto de conteúdo pode ser armazenado em cache. Mensagens de interesse podem ser comparadas com caches, conhecidos como ficheiros de conteúdo, ao longo do caminho, não apenas só nos seus emissores.

Para um objeto de conteúdo efetuar uma correspondência com o interesse, o nome do objeto de conteúdo deve ser igual ao nome do interesse. Se o interesse tem uma restrição `KeyID`, então deve ser exatamente igual à `KeyID` que valida um *Content Object* (isto não é uma operação de criptografia). Se o interesse carrega uma restrição `ContentObjectHash`, então o *Forwarder* deve calcular o SHA-256 *digest* do Objecto de Conteúdo e compara-lo com a igualdade de restrição do interesse.

***Forwarder* (Encaminhador)**:

Um consumidor de conteúdo emite um pedido de interesse de dados através da rede. O interesse é transmitido através de um conjunto de encaminhadores até o Objecto de Conteúdo ser encontrado ou tempo de vida do interesse expira.

Um encaminhador consiste em três tabelas referenciais:

Base de Informação de Encaminhamento (FIB): O FIB é a tabela de encaminhamento. É uma tabela de correspondência nome de prefixo mais longo e povoada por um protocolo de encaminhamento ou com rotas estáticas.

Tabela de interesses pendentes (PIT): O estado dos registos interesse tais como os objetos de conteúdo que satisfazem o interesse podem seguir o caminho inverso de interesses ao requerente. Isto também implementa a agregação de interesses, de modo que múltiplos interesses semelhantes não sejam redirecionadas a montante.

Armazenamento de conteúdo (CS): O armazenamento de conteúdo é uma *cache* opcional dentro da rede de conteúdo de objetos que podem ser usados para satisfazer mensagens de interesse com menos saltos.

Dependendo do seu papel na rede, um encaminhador também pode executar os protocolos adicionais, como serviços e protocolos de descoberta de conteúdo.

Como um objeto de conteúdo se move através do "caminho rápido" da rede, que faz uma verificação de correspondência com interesses pendentes em cada nó de acordo com a seguinte regra:

Um objeto de conteúdo "satisfaz" o interesse se e somente se (a) o nome do objeto de conteúdo corresponde exatamente ao nome de interesse, e (b) o Algoritmo de Validação da `KeyID` do objeto de conteúdo é exatamente igual a restrição `KeyID` de Interesses, caso tenha

sido fornecida, e (c) o ContentObjectHash equivale a restrição de *hash* objeto de conteúdo de Interesse, caso tenha sido fornecida.

Apenas os sistemas finais verificam assinaturas criptográficas, o que diminui a carga computacional em cada nó e portanto, aumenta o rendimento. No entanto, cada salto pode precisar de calcular o ContentObjectHash, se o interesse pendente inclui uma restrição ContentObjectHash.

O armazenamento de conteúdo tem regras de correspondência mais restritivas, do que o Encaminhador, a fim de evitar falhas de persistentes decorrentes de conteúdo incorreto que entra no armazenamento de maior durabilidade. Se um interesse tem uma restrição KeyID, o armazenamento de conteúdo deve verificar a assinatura de um objeto em cache antes de voltar para o requisitante. Este é um passo muito mais restritivo do que no caminho rápido. Isso significa que o interesse deve levar a chave pública desejada ou o próprio objeto de conteúdo deve levar a sua própria chave pública. Se um interesse tem uma restrição ContentObjectHash, então o armazenamento de conteúdo deve verificar a *hash* de um objeto de conteúdo antes de enviá-lo ao requisitante.

Interest (Interesse):

Um interesse carrega um nome, um identificador do emissor opcional chamado o KeyID, e uma identidade inescapável opcional chamado o ContentObjectHash, que é um *hash* criptográfico seguro de uma mensagem de objeto de conteúdo. As mensagens de interesse também contêm um campo HopLimit de 1 byte para limitar o número de saltos, a fim de evitar ciclos. Quando um encaminhador recebe um interesse com um HopLimit, ele diminui o valor do campo HopLimit. Não pode transmitir um interesse com um HopLimit de 0 para fora de uma interface externa; só pode satisfazê-la a partir do armazenamento de conteúdo *built-in* ou encaminhá-lo para uma aplicação local.

***Content Object* (Objecto de conteúdo)**:

O Objecto de Conteúdo carrega um nome e uma carga útil. Também pode incluir informações sobre como validar a mensagem. Vários algoritmos de validação possíveis podem ser utilizados, incluindo uma verificação de integridade da mensagem (MIC), um código de autenticação de mensagem (MAC) ou uma assinatura criptográfica. O ValidationAlgorithm também carrega uma KeyID, que identifica o emissor que está a autenticar o Objecto de Conteúdo. Se um interesse carrega uma KeyID além de um nome, ele limita o universo de objetos de conteúdo que correspondem ao nome daqueles que foram validados com essa KeyID. O ContentObjectHash é uma *hash* criptográfica de todo o Objecto de Conteúdo. Se usado num interesse, além de um nome, seleciona um objeto de conteúdo específico desse nome. O ContentObjectHash é a *hash* SHA-256 do objeto de conteúdo no formato de fio, a

partir da marca de abertura do objeto de conteúdo até ao fim do objeto de conteúdo. Não está incluído o pacote fixo ou cabeçalhos adicionais opcionais. O `ContentObjectHash` não é um campo explícito no pacote, mas deve ser calculado. Este cálculo fornece uma garantia de que, para o correto comportamento dos nós dentro da rede, se um utilizador solicita um objeto de conteúdo pela *hash* e a rede entrega o pacote correto.

***Packet Format* (Formato do Pacote)**:

CCNx utiliza um formato *Type-Length-Value* para codificar mensagens num "*TLV Packet*". Este é um novo formato de pacote geral composto por um cabeçalho fixo, um conjunto de cabeçalhos opcionais em formato TLV, e uma mensagem CCNx que contem conteúdo codificado em TLV. Os campos tipo e comprimento são ambos exatamente de dois bytes. Utilizando um tipo e comprimento de tamanho fixo evita problemas com *aliases* e é simples de analisar para equipamentos de alta velocidade. O formato é adequado para utilização direta sobre uma camada MAC, ou encapsulado dentro de um protocolo de rede ou de transporte. Os pacotes CCNx misturam cabeçalhos fixos com campos TLV. O formato do pacote TLV é projetado para a flexibilidade do protocolo. O cabeçalho estático fornece acesso imediato aos campos básicos necessários para a análise. Os campos de cabeçalho opcional podem ser utilizados para estender a funcionalidade do cabeçalho estático, conforme necessário.

A carga útil máxima TLV é 64KiB.

As mensagens CCNx começam com um cabeçalho de 8 bytes fixo (num formato não-TLV), seguido por um conjunto de cabeçalhos opcionais (em formato TLV) e a mensagem de protocolo (em formato TLV). O campo `HeaderLength` no cabeçalho fixo representa o comprimento combinado dos cabeçalhos fixos e opcionais, portanto, o início da mensagem de protocolo encontra-se no "início de pacote + `HeaderLength`".

A carga útil de um pacote TLV CCNx é a própria mensagem de protocolo. A *Hash* do objeto de conteúdo é calculado apenas sobre a carga útil, excluindo os cabeçalhos fixos e opcionais, e como podem mudar de salto para salto. Informação assinada, da mesma forma ou semelhança de *hashes* não devem incluir qualquer um dos cabeçalhos fixos ou opcionais. A carga útil deve ser autossuficiente no caso em que os cabeçalhos fixos e opcionais são removidos.

Um cabeçalho fixo comum é:

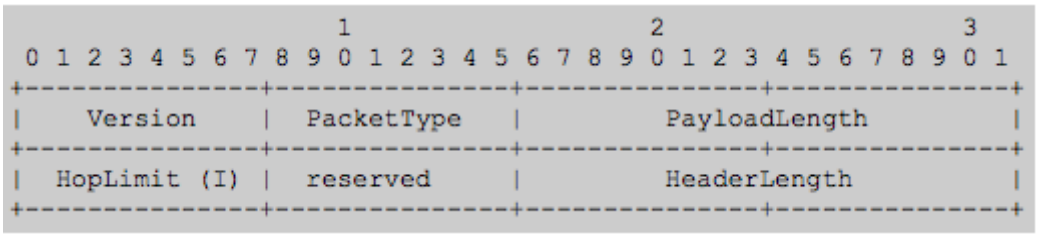


Figura 17 – Cabeçalho fixo comum (retirado de [CCNx])

Onde

- Versão: define a versão de pacote.
- `HeaderLength`: O comprimento do cabeçalho fixo (8 bytes) + cabeçalhos opcionais. O valor mínimo é "8".
- `PacketType`: 1 = *Interest*, 2 = *Content Object*.
- `PayloadLength`: Total de octetos seguintes tudo *headerss* (cabeçalho fixo mais cabeçalhos opcionais).
- `HopLimit`: Usado por mensagens de interesse para limitar o número de saltos para evitar *loops*.

Entre o cabeçalho fixo e o início da mensagem de protocolo CCNx estão os cabeçalhos opcionais em formato TLV. São para as opções processados por salto, como um campo DSCP-equivalente. Os cabeçalhos opcionais são desordenados. A ordem em que são processados não deve afectar o resultado do processamento de outros cabeçalhos opcionais.

Cabeçalhos opcionais em formato TLV seguem cabeçalhos fixos num pacote CCNx.

A mensagem de protocolo CCNx - o *Interest* ou `ContentObject` - começa imediatamente após os cabeçalhos opcionais. O corpo da mensagem é codificada com a mesma estrutura TLV que estes cabeçalhos. A mensagem de protocolo é seguida de `ValidationAlgorithm` opcional e `ValidationPayload` TLVs. O campo `ValidationAlgorithm` especifica a forma de verificar a mensagem CCNx. Os exemplos incluem a verificação com uma verificação de integridade da mensagem (MIC), um código de autenticação de mensagem (MAC), ou uma assinatura criptográfica. O campo `ValidationPayload` contém a saída de validação, tais como o código CRC32C ou a assinatura RSA.

Seguindo a mensagem CCNx com campos `ValidationAlgorithm` e `ValidationPayload` opcionais, tem várias características desejáveis. Neste esquema, a validação de uma mensagem CCNx é modular e o mesmo para todos os tipos de pacotes. É importante que todas as mensagens, incluindo interesse, possam levar pelo menos um MIC.

O pacote global é semelhante a isto:

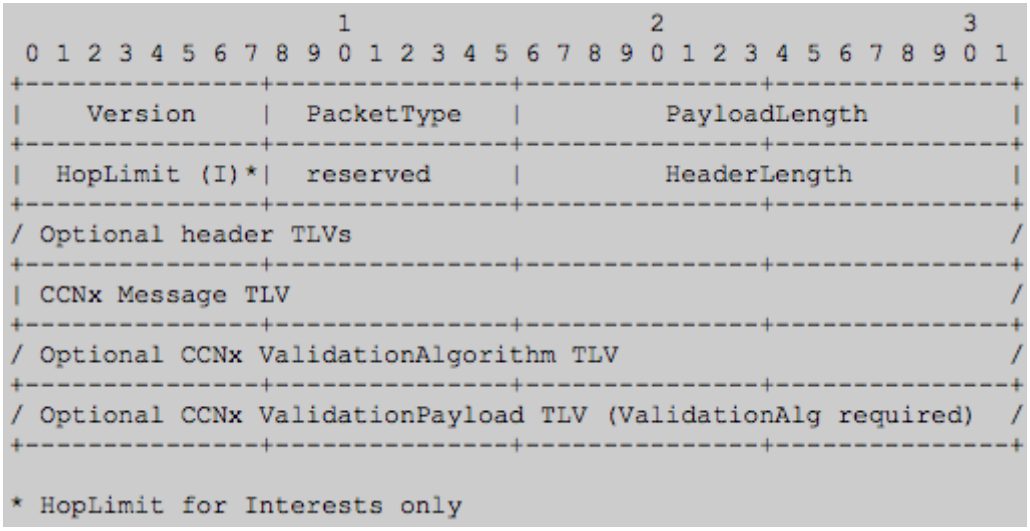


Figura 18 – Estrutura geral de um pacote global (retirado de [CCNx])

Depois de descartar os cabeçalhos fixos e opcionais a carga útil restante deve ser uma mensagem de protocolo válido. Portanto, a carga começa sempre com 4 bytes para a definição da mensagem TLV de protocolo (se é um interesse ou um objeto de conteúdo ou um tipo de mensagem futura) e seu comprimento total.

A incorporação de uma unidade de dados de protocolo autossuficiente dentro dos cabeçalhos fixos e opcionais, permite que uma pilha de rede descarte os cabeçalhos e opere apenas na mensagem incorporada.

Este protótipo, desenvolvido e gerido pelo PARC, para implementar uma rede com uma arquitetura CCN, apresenta as seguintes características:

- Suporte *multi-OS* (Linux, MacOS, FreeBSD).
- Implementação Android para *smartphones.*
- Escrito em duas linguagens:
    - Implementação C (encaminhador, repositório, armazenamento persistente de CCNx, dados, biblioteca, serviços, documentação da estrutura da API, e uma suite de testes): necessário para toda a comunicação CCNx, preferencialmente para desenvolvimento de sistemas.
    - Implementação JAVA (biblioteca, utilitários, documentação da estrutura da API, e uma suite de testes): preferencial para o desenvolvimento de aplicações.

Este protótipo encontra-se atualmente na versão 0.8.2 e está disponível em código fonte "*Open Source*" em http://www.ccnx.org, assim como toda uma vasta panóplia de documentação sobre a temática.

O CCNx é composto atualmente por uma vasta lista de comandos, que serão descritos ao longo deste capítulo, mas dos quais destacamos pelo seu grau de importância fundamental os seguintes:

- ***ccnd*** – é o CCNx *Daemon*, este comando é a raíz deste toda implementação. O `ccnd` normalmente não é executado diretamente. Utiliza-se o ccndstart para executar uma instância de `ccnd`. O `ccnd` normalmente é executado indefinidamente. Utiliza-se o `ccndsmoketest` para encerrar um `ccnd`.
  O `ccnd` é o software de encaminhamento/*router* para CCNx e é necessário para a comunicação normal no protocolo CCNx. A configuração típica é executar um `ccnd` em cada *host* e os aplicativos em execução em cada *host* irão comunicar através do `ccnd` local e ele vai comunicar através de redes interligadas (diretamente ou através de um processo transformador *link*).
  O `ccnd` não tem nenhuma opção na linha de comando. As opções básicas são controladas por variáveis de ambiente. A tabela de encaminhamento (FIB) é preenchida com protocolos de registo de sobre CCNx. Utiliza-se o `ccndc` para a configuração da FIB.
  O `ccnd` comunica através do protocolo CCNx UDP, TCP, ou Unix *domain sockets* (este último apenas para os processos locais). Também fornece uma visualização *web* simples do estado sobre HTTP, no `CCN_LOCAL_PORT`. Os pacotes "*Interests*" e de pacote "*Data*" são encapsulados em UDP

  Após o `ccnd` iniciar, o controlo do seu comportamento ocorre através de protocolos CCNx. Para mais informações, por favor consulte o `NameConventions` na documentação técnica.

  Este é o local onde o `ccnd` publica sua chave, utilizando o perfil de localização do serviço:

  ```
  ccnx: /% C1.M.S.localhost /% C1.M.SRV / ccnd
  ```

  Para o benefício dos nós vizinhos, a chave também é publicada aqui:

  ```
  ccnx: /% C1.M.S.neighborhood /% C1.M.SRV / ccnd
  ```

  O seguinte prefixo é o principal utilizado para comunicar com esta instância de `ccnd`. Aqui `<ccndid>` refere-se ao SHA256 *digest* da chave pública do `ccnd`. Note-se que, por razões históricas, isto não utiliza a convenção da chave de marcador. Cada instância de `ccnd` poderá ser assumida que está a utilizar uma chave pública distinta. Consulte o *"Registration Protocol"* para obter uma descrição dos protocolos utilizados neste *namespace*.

  ```
  ccnx: / ccnx / <ccndid>
  ```

A corrente de mudanças de *status* de interface pode ser obtida através do nome abaixo. Isto utiliza os perfis de versão e de segmentação padrão, portanto, um utilitário como o `ccncat` pode ser usado para lê-lo. De momento, apenas uma versão do fluxo é iniciado quando é solicitado pela primeira vez, portanto, um programa que precisa de o aceder, deve iniciar logo após `ccnd` e executar, uma vez que a instância `ccnd` está a executar. O formato deste fluxo é simplesmente baseado em texto.

```
ccnx:/ccnx/<ccndid>/notice.txt
```

Quando dois `ccnd`'s começam a troca de pacotes "*Interests*" e de dados "*Data*", ocorre um "aperto de mão" automático para acordar um prefixo do nome que ambos os nós podem usar para se referir ao canal de comunicação. Este prefixo é registado em cada nó de transmissão interesses para o outro. Um exemplo desse tal prefixo é:

```
ccnx: /% C1.M.FACE /% C1.MG% 00% 00XL% DB% F0% 0A% 0APn% 0F% B1% F2% 1F
```

O primeiro componente do nome é sempre o mesmo. O segundo componente é um *guid* gerado aleatoriamente, com um marcador *guid*. (Estas *guids* são abreviados abaixo para torná-los mais fáceis de ler.)

Como subproduto deste aperto de mão, cada `ccnd` cria um objeto de conteúdo que representa o seu ponto final. Por exemplo:

```
ccnx: /% C1.M.FACE /% C1.MG% 00% 00XL42 / C1.M.NODE% /% C1.MK% 00 … 1 /
face ~ 7 /% FD …
ccnx: /% C1.M.FACE /% C1.MG% 00% 00XL42 / C1.M.NODE% /% C1.MK% 00 … 2 /
face ~ 9 /% FD …
```

O terceiro componente é a constante `% C1.M.NODE`, para permitir se o restante espaço pode ser utilizado para outros fins. O quarto componente é o `ccndid`, com o marcador de chave *digest*. Estes foram abreviados no exemplo. O quinto componente é o `FaceID`, em *ascii* decimal, com um diferenciador líder da interface. Os outros dois componentes são para controlo de versões padrão e segmentação.

Atualmente não há nenhuma carga nesses objetos de conteúdo. No entanto isto está sujeito a alterações.

- ***ccndc*** - manipula a tabela de encaminhamento CCNx.
  `ccndc` é um utilitário de encaminhamento simples / *daemon* que configura a tabela de encaminhamento (FIB) num `ccnd`. Pode ser usado tanto como um comando para adicionar ou excluir entradas estáticas no CCNx FIB (aproximadamente análogo ao *route*, um utilitário que existe, para manipular uma tabela de encaminhamento IP). Quando uma interface é especificada, que tanto pode ser pelos parâmetros (máquina,

porta, etc) ou por número de interfaces. A interface pode ser criada ou destruída, sem referência a um prefixo, ou será criada automaticamente se os parâmetros são fornecidos. `ccndc` também pode ser executado como um *daemon* que irá criar dinamicamente interfaces e entradas FIB para transmitir certos interesses CCNx com base em registos do servidor DNS. Os interesses que podem ser encaminhados de forma dinâmica desta forma são aqueles que têm um componente inicial do nome que é um nome de DNS legal, para a qual existe um registo SRV DNS apontando para um ponto final para o tráfego de protocolo de encapsulamento CCNx através da Internet. `ccndc` também suporta ficheiros de configuração que contem conjuntos de comandos.

- ***ccnr*** - Repositório de CCNx.
  Um Repositório suporta a rede, preservando o conteúdo e responde a pacotes "*Interest*" que solicitam conteúdo que ele contém. Estes serviços estão disponíveis para os componentes CCN, incluindo aplicações de clientes habilitados para CCN. Um repositório pode existir em qualquer nó, e é recomendado se as aplicações nesse nó precisam de preservar os dados. Veja protocolos de repositório CCNx, para obter mais informações sobre o Repositório.

  O comando `ccnr` começa o repositório, utilizando o diretório especificado pela variável de ambiente `CCNR_DIRECTORY`. Na inicialização, o repositório recupera e aplica as opções de configuração descritas abaixo.

  O diretório nomeado por `CCNR_DIRECTORY` deve existir. Para iniciar `ccnr` como um *daemon*, redirecionar `stderr` para um ficheiro e executar

  ```
  bin / CCNR &
  ```

`ccnr` desliga-se normalmente se recebe `SIGINT` ou `SIGTERM`, ou se o `ccnd` ao qual está ligado é desligado.

Não execute dois repositórios no mesmo diretório de armazenamento de apoio, ao mesmo tempo.

O repositório usa `$CCNR_DIRECTORY/repoFile1` para armazenamento persistente de conteúdo CCN *Objects*. Um índice de disco residente facilita o rápido *start-up* e limita o consumo de memória. Se um índice não existir será construído durante a inicialização.

Um ficheiro de política especifica os *namespaces* para o qual o repositório aceita e guarda o conteúdo. O nome do ficheiro de política é a concatenação do prefixo global e "`data/policy.xml`". A menos que uma política alternativa seja explicitamente definida / publicada sob o nome da política de informação, a política padrão será /, o que

significa que as gravações serão aceites para qualquer nome e atendidas por qualquer nome para o qual não é o conteúdo.

O Repositório é configurado a partir de informações no ficheiro de texto, `$CCNR_DIRECTORY/config`. O ficheiro só é lido no momento da inicialização.

Quaisquer das variáveis no ficheiro também podem ser expressas como variáveis de ambiente que são examinados no momento da inicialização. Se a mesma variável é definida no ficheiro de configuração e uma variável de ambiente, o valor no ficheiro de configuração tem precedência.

`CCNR_BTREE_MAX_FANOUT = <Max fanout>` onde `<Max fanout>` é o número máximo de entradas no índice B-tree nós interiores. O valor máximo para `<Max fanout>` é 1999.

`CCNR_BTREE_MAX_LEAF_ENTRIES = <entradas Max folha>` onde `<entradas folha Max>` é o número máximo de entradas no índice B-tree nós folha. O valor máximo para `<entradas folha Max>` é 1999.

`CCNR_BTREE_MAX_NODE_BYTES = <tamanho Max índice>` onde `<tamanho Max índice>` é o tamanho máximo do índice de nós de árvore-B, em *bytes*. O valor máximo para `<tamanho do índice Max>` é 2097152.

`CCNR_BTREE_NODE_POOL = <nós de índice Max em cache>` onde `<nós de índice Max em cache>` é o número máximo de nós de índice B-tree em cache na memória. O valor máximo para nós de índice `<Max em cache>` é 512.

`CCNR_CONTENT_CACHE = <Max objetos em cache>`

onde `<Max objetos em cache>` é o número máximo de objetos de conteúdo em cache na memória. O valor máximo para `<Max objetos em cache>` é 4201.

`CCNR_DEBUG = <nível registo de depuração>`

onde `<nível de log de debug>` é um dos seguintes. Se a opção não for especificada, o padrão é de advertência.

- `NONE` - Nenhuma mensagens
- `SEVERE` - graves, provavelmente fatais, erros
- `ERROR` - erros
- `WARNING` - avisos
- `INFO` - mensagens informativas
- `FINE, FINER, FINEST` - depuração / rastreamento

`CCNR_DIRECTORY = <directory>`

onde <directory> é o diretório onde o armazenamento de repositório está localizado, cujo padrão é o diretório atual. CCNR_DIRECTORY é ignorado no ficheiro de configuração.

CCNR_GLOBAL_PREFIX = <URI>

onde <URI> é o CCNx URI que representa o prefixo em que data/policy.xml é armazenado, e só tem sentido se nenhum ficheiro de política existe na inicialização. <URI> é esperado por convenção a ser globalmente único e significativo, em vez de apenas localmente única e contextualmente significativa. Se não for especificado, o padrão de URI para ccnx: /parc.com/csl/ccn/Repos.

CCNR_LISTEN_ON = <lista de endereços IP>

onde <lista de endereços IP> é uma lista de endereços IP para escutar o status, no caso em que CCNR_STATUS_PORT é dado. Endereços IP podem ser em qualquer formato IPv4 (por exemplo, 127.0.0.1) ou no formato IPv6 (por exemplo, fe80 :: 226: BBFF: fe1c: 5530). Os endereços podem ser separados por espaços, vírgulas ou ponto e vírgula. Se não for especificado, o padrão é efetivamente *localhost*.

CCNR_MIN_SEND_BUFSIZE = <tamanho do buffer Min>

onde <tamanho do buffer Min> é o tamanho mínimo em *bytes* do *buffer* de tomada de saída (SO_SNDBUF) da tomada utilizada para comunicar com ccnd. O valor máximo para <tamanho do buffer Min> é 16384. Se o sistema fornecer mais do que isso, por padrão, o valor do sistema é usado.

CCNR_PROTO = <type>

onde <tipo> é o tipo de ligação, que deve ser tcp ou unix. Se <type> é tcp, Repo vai ligar para ccnd via TCP; se <type> é unix, Repo irá conectar via Unix IPC. Se não for especificado, o padrão é unix.

CCNR_STATUS_PORT = <port>

onde <port> é a porta TCP a ser usada por um servidor de status. Se essa opção não for especificada, nenhum estado é servido. Como um expediente, esta porta também pode ser usado para inserir conteúdo objetos no repositório.

CCNR_START_WRITE_SCOPE_LIMIT = <Scope limit>

onde <Scope limit> está na gama 0..3 (padrão 3). Processo de start-write(-checked) interesses com um âmbito não excedam o valor dado. 0 é efetivamente somente leitura. 3 indica ilimitado.

CCNS_DEBUG = <sincronização nível registo de depuração>

onde <sincronização nível registo de depuração> tem os mesmos valores que para CCNR_DEBUG acima. Se não for especificado, o padrão é de WARNING.

CCNS_ENABLE = <fazer sincronização>

onde <fazer sincronização> especifica se desativar (0) ou permitir (1) processamento Sync. Se não for especificado, o padrão é ativado.

CCNS_FAUX_ERROR = <simular perda aleatória>

onde <simular perda aleatória> especifica como e qual será a perda de pacotes aleatórios para simular. Se <simular perda aleatório> 0, sem perda é simulado; se na gama de 1-99, o número é a percentagem de pacotes para cair aleatoriamente. Se não for especificado, o padrão é 0 (sem perda).

CCNS_HEARTBEAT_MICROS = <batimento cardíaco>

onde <batimento cardíaco> é o número de microssegundos entre batimentos de sincronização e deve ser um número inteiro no intervalo 100,000-10000000. Se não for especificado, o padrão é 200000.

CCNS_MAX_COMPARES_BUSY = <max compara>

onde <max compara> é o número máximo de raízes de sincronização que pode estar no estado a comparar simultaneamente, e deve ser um número inteiro na gama de 1-100. Se não for especificado, o padrão é 4.

CCNS_MAX_FETCH_BUSY = <max busca>

onde <max busca> é o número máximo de nó simultâneo ou conteúdo procura por raiz Sync, e deve ser um número inteiro no intervalo 1-100. Se não for especificado, o padrão é 6.

CCNS_NODE_FETCH_LIFETIME = <nf vida>

onde <nf vida> é a quantidade máxima de tempo em segundos para esperar por uma resposta a um pedido NodeFetch, e deve ser um número inteiro no intervalo 1-30. Se não for especificado, o padrão é 4.

CCNS_NOTE_ERR = <excepcional bandeira erros>

onde <erros excepcionais bandeira> especifica se relatório excepcional de erros de sincronização está desativada (0) ou habilitado (1). Se não for especificado, o padrão é 0 (desativado).

CCNS_REPO_STORE = <armazenamento de estado bandeira>

onde <bandeira do estado loja> especifica se o armazenamento de estado de sincronização para o repositório é desativado (0) ou habilitado. Se não for especificado, o padrão é 1 (habilitado).

CCNS_ROOT_ADVISE_FRESH = <frescura>

onde <frescura> é a quantidade de tempo que uma resposta a uma sincronização RootAdvise irá ficar "fresca" (válida) na *cache* ccnd em segundos, e deve ser um número inteiro na gama de 1-30. Se não for especificado, o padrão é 4.

```
CCNS_ROOT_ADVISE_LIFETIME = <ra vida>
```

onde `<ra vida>` é a quantidade máxima de tempo em segundos para esperar por uma resposta a um pedido `RootAdvise`, e deve ser um número inteiro no intervalo 1-30. Se não for especificado, o padrão é 20.

```
CCNS_STABLE_ENABLED = <Bandeira estável>
```

onde `<Bandeira estável>` especifica se o armazenamento de sincronização de pontos estáveis ao repositório é desativado (0) ou habilitado (1). Se não for especificado, o padrão é 1 (habilitado).

```
CCNS_SYNC_SCOPE = <âmbito sincronização>
```

onde `<âmbito sincronização>` é o *"scope"* aplicado para sincronizar interesses gerados (`RootAdvise` e procura remota). O valor deve ser 0 (sem *scope*), 1 (host local), ou 2 (próximo *host*). Se não for especificado o padrão é 2.

Embora todos os comandos sejam necessários, apenas demos acima um destaque aos principais e um nível de detalhe mais aprofundado e apresentamos a seguir uma lista completa.

Lista completa de comandos do CCNx:

- ***ccn_ccnbtoxml*** - converte CCN codificados em formato XML.
- ***ccn_xmltoccnb*** - converte XML em dados binários codificados CCN (CCNB).
- ***ccnacl*** - mostra e modifica listas de controlo de acesso (ACLs) para controlo de acesso de um *namespace* de conteúdo CCNx.
- ***ccngroup*** - mostra e modifica grupos de controlo de acesso para controlo de acesso de um *namespace* de conteúdo CCNx.
- ***ccnc*** - um programa de *chat* de comunidade de texto simples.
- ***ccnd*** - CCNx *Daemon*.
- ***ccndc*** - manipula a tabela de encaminhamento CCNx.
- **ccndcontrol** - manipula a tabela de encaminhamento CCNx (Java).
- ***ccndlogging*** - altera o nível de log de um `ccnd` em execução.
- ***ccndsmoketest*** - Programa simples para o *smoke-test* de `ccnd`.
- ***ccndstart*** - Inicia o `ccnd` em segundo plano e configura o encaminhamento de acordo com a configuração.
- ***ccndstatus*** - Exibi o status de um `ccnd` em execução.
- ***ccndstop*** - Para a execução do `ccnd`.

- ***ccnexplore*** - Java Swing GUI para explorar o conteúdo armazenado em repositórios.
- ***ccnls*** - Tenta listar os componentes de nomes disponíveis no próximo nível da hierarquia de um determinado prefixo de nome CCNx pedido.

- ***ccnlsrepo*** - Explora o conteúdo armazenado num determinado prefixo num ou mais repositórios.
- ***ccngetfile*** - Obtém um ficheiro publicado como conteúdo CCNx e guarda-o num ficheiro local.
- ***ccnputfile*** - Publica um ficheiro como conteúdo CCNx.
- ***ccnputmeta*** - Associa um ficheiro com conteúdo CCNx como metadados.
- ***ccngetmeta*** - Obtém metadados associados ao conteúdo CCNx especificado e guarda-o num ficheiro local.
- ***ccnlibtest*** - Executa testes de script da biblioteca em linguagem C do ccnx .
- ***ccnnamelist*** - Extrai nomes de um ficheiro de dados codificados de CCNB .
- ***ccnrm*** - Marca como obsoletos quaisquer objetos de conteúdo em cache localmente que correspondem aos prefixos pedidos.
- ***ccnseqwriter*** - Envia dados do *stdin* utilizando o versionamento CCN e a segmentação.
- ***ccn_repo*** - Utilitário para iniciar, parar e sinalizar aplicação Java de repositórios CCNx .
- ***ccncat*** - Lê fluxos de conteúdo CCNx e escreve para o *stdout*.
- ***ccnslurp*** - Imprime nomes de todo o conteúdo de uma parte específica do *namespace* CCNx.
- ***ccnchat*** - iniciar (ou juntar-se a) uma sala de *chat*.
- ***ccnpeek*** - Obtém um item de conteúdo que corresponde ao prefixo do nome e escreve-o para *stdout*.
- ***ccnpoke*** - Lê dados do *stdin* e envia-os como um único `ContentObject` em resposta a um interesse.
- ***ccnr*** - Repositório de CCNx.
- ***ccnrpolicyedit*** - edita o ficheiro policy.xml
- ***ccnsyncslice*** - Manipula as fatias de configuração de sincronização.
- ***ccnsyncwatch*** - Monitoriza as operações de sincronização e relata novos objetos de conteúdo.
- ***ccnlink*** - Cria um novo link para um destino URI.
- ***ccnprintlink*** - Imprime informações sobre os links indicados.
- ***ccnrepoimport*** - Importa um ficheiro para um Repositório.
- ***ccntestloop*** - Executa os testes da unidade CCNx repetidamente.
- ***ccninitkeystore*** - Inicializa uma *keystore* CCNx com determinados parâmetros.
- ***ccninitaeskeystore*** - Inicializa uma *keystore* CCNx AES (simétrica) com determinados parâmetros.

6.3 Instalação e experiências laboratoriais

O CCNx foi instalado num sistema Ubuntu 14.04. Tendo em conta que o código Java corre sobre o `ccnd` (CCNx *Daemon*) a partir do código C ++. Portanto, devemos primeiro compilar o código fonte em C, que então é posteriormente chamado pelo Java.

Para compilar o código C a partir de uma instalação do Ubuntu 14.04 foi necessário instalar as dependências:

```
sudo apt-get update

sudo apt-get install git-core tcpdump

sudo apt-get install gcc make libc6-dev autoconf autotools-dev libtool

sudo apt-get install libssl-dev libxml2-dev libxml2-utils libexpat1-dev
libcrypto++9 expat libpcap-dev libxml2 athena-jot python-dev wireshark

sudo apt-get install ant openjdk-7-jdk openjdk-7-jre
```

Para proceder à instalação, descarregue o CCNx do Projeto CCNx em www.ccnx.org

```
wget http://www.ccnx.org/releases/ccnx-0.8.2.tar.gz
```

Abra uma janela de terminal e descompacte-o

```
tar -xzf ccnx-0.8.2.tar.gz

cd ccnx-0.8.2
```

Agora navegue para o diretório de descompactação

Execute:

```
CFLAGS='-g -DCCN_URI_DEFAULT_ESCAPE=2'

./configure
```

Efetue o *build* do CCN com:

```
make
```

Teste com:

```
make test
```

Após a fase de testes, cujos *printscreen's* serão apresentados mais adiante, finalize a instalação com

```
sudo make install
```

Tudo em CCN requer uma CCNx *Daemon*. Atualmente isto só é construído em código C. Para iniciar a `ccnd` que acabamos de construir com o código de execução C:

```
sudo ccndstart &
```

Os programas de teste são executados em Java, que usam o `ccnd` em C. Então, para começar a executar *app* de *chat* baseado em Java:

```
ccnchat ccnx:/test_room
```

Também podemos ver a interface *Web* do `ccnd` em:

```
http://localhost:9695
```

Na Figura 19 podemos ver uma janela de estatísticas do CCNx, que pode ser consultada por *Web browser* `htpp://localhost:9695`

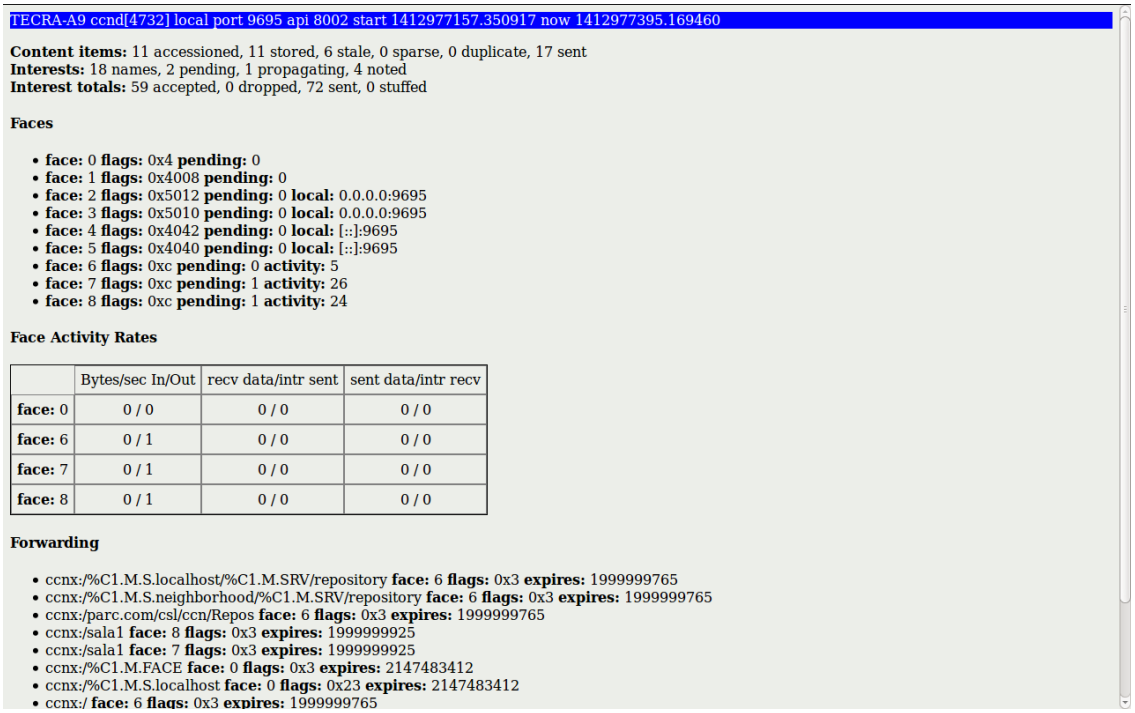


Figura 19 – Exemplo de janela estatísticas num *Web browser*

Na figura 20 é possível ver a utilização do chat a partir de terminais diferentes,

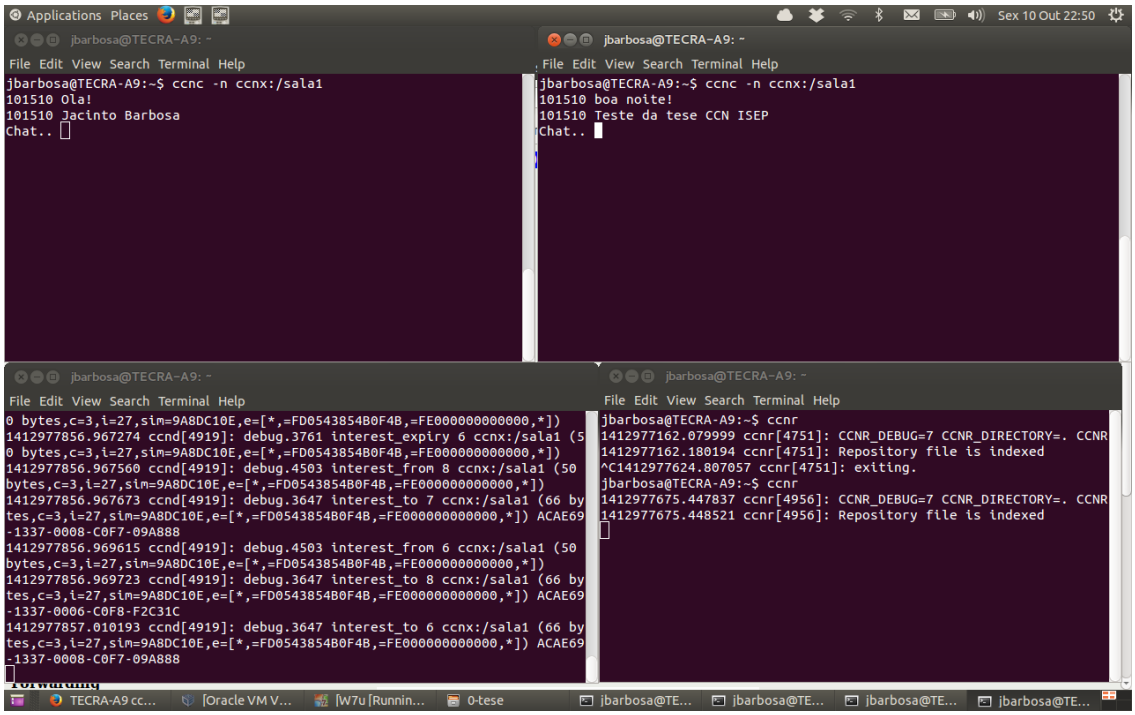


Figura 20 – Exemplo de utilização do CCN chat

A seguir, na Figura 21, ilustram-se exemplos de testes de vários cenários de tráfego criado pela CCNx durante a fase do `make test` visualizadas em `http://127.0.0.1:63000/`

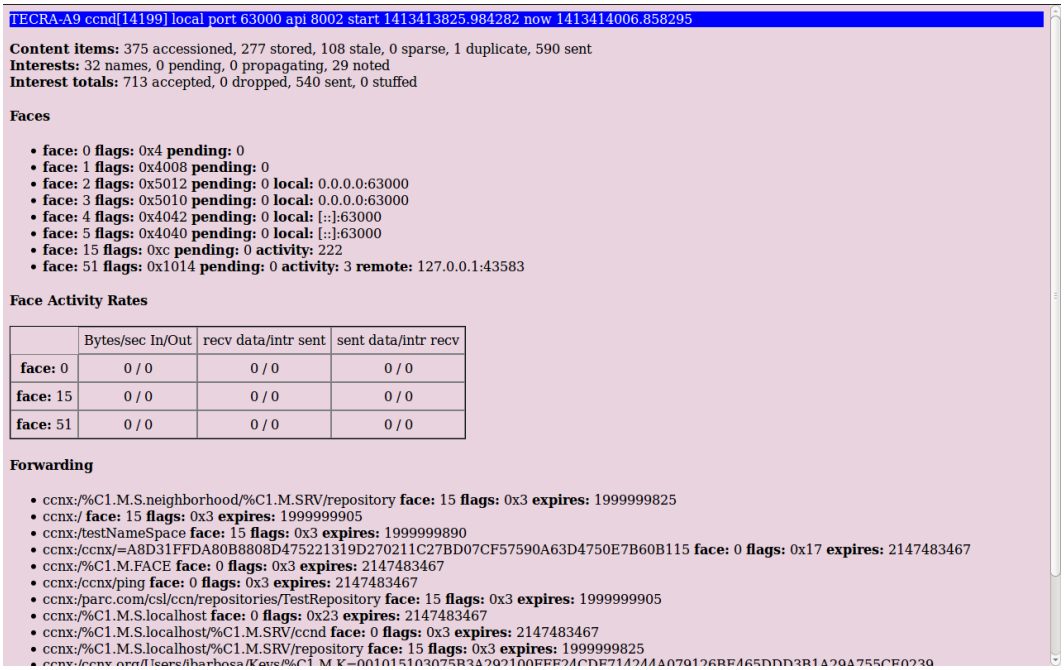


Figura 21 – Janela de estatísticas na fase do "`make test`"

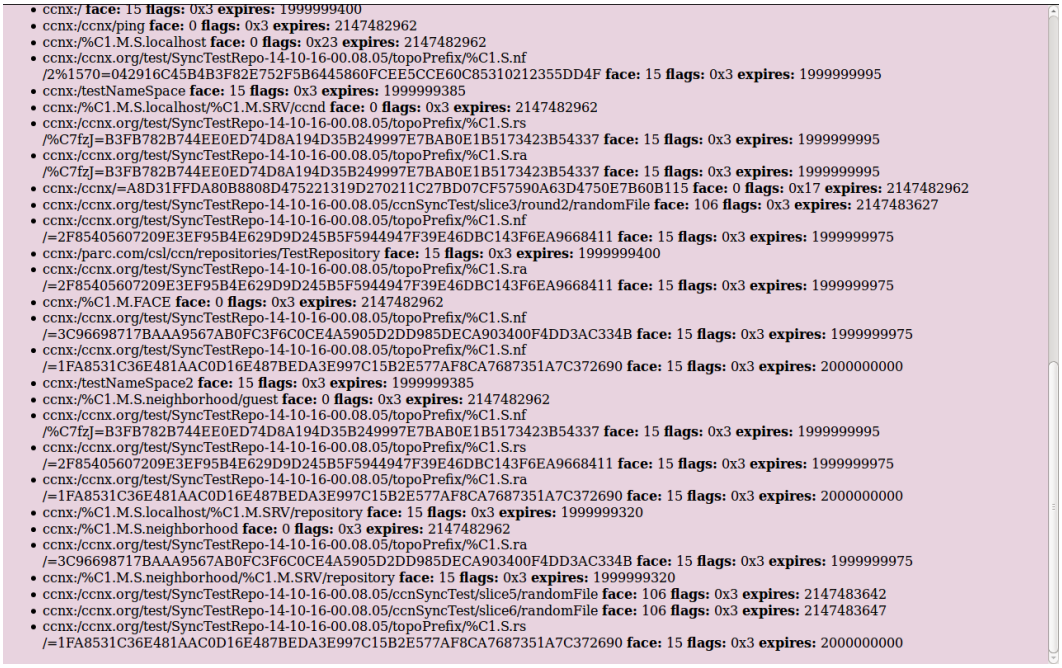
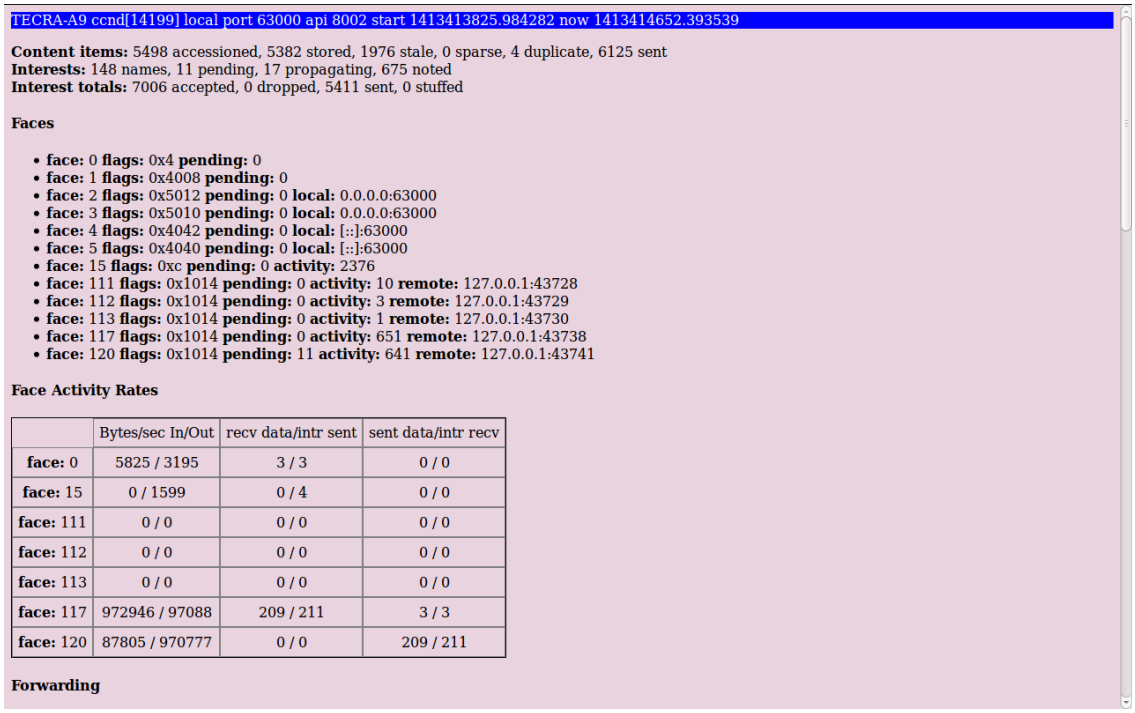


Figura 22 - Janela de estatísticas na fase do "`make test`" (continuação)

Conforme podemos observar entre as janelas iniciais dos testes, ilustradas na Figura 21 e Figura 22, no início dos testes os itens "*Content items*", "*Interests*", "*Faces*" e "*Faces Activity Rates*" apresentavam valores iniciais das simulações, para o repositório criado com o nome "*test*". Durante a fase de demonstração de testes são efetuadas as operações mais comuns que poderemos efetuar na implementação e simulação de uma rede baseada no paradigma CCN, desde a criação de um repositório, adicionar conteúdos, procurar conteúdos e obtenção de conteúdos do repositório. No final dos testes, na Figura 23 podemos observar que os resultados das estatísticas recolhidas mostram-nos o total de itens acedidos no "*Content items*", o total de "*interests*" satisfeitos, os pendentes e os propagados. Podemos também recolher uma leitura da atividade em cada interface, no que diz respeito à quantidade de informação enviada e recebida em "*Face activity Rates*" na coluna "*Bytes/sec In/Out*" e colunas adjacentes "*recv data/intr sent*" e "*sent data/intr recv*". Apesar de a seguir efetuarmos algumas experiências na implementação e utilização de uma rede sobre o paradigma CCN, esta parte de testes disponibilizada pelo protótipo CCNx, demonstra as potencialidades do paradigma, para que seja mais intuitivo aos investigadores que vão utilizar esta ferramenta, perceberem que tipo de experiências poderão realizar isoladamente.



TECRA-A9 ccnd[14199] local port 63000 api 8002 start 1413413825.984282 now 1413414652.393539

Content items: 5498 accessioned, 5382 stored, 1976 stale, 0 sparse, 4 duplicate, 6125 sent
Interests: 148 names, 11 pending, 17 propagating, 675 noted
Interest totals: 7006 accepted, 0 dropped, 5411 sent, 0 stuffed

Faces

- **face:** 0 **flags:** 0x4 **pending:** 0
- **face:** 1 **flags:** 0x4008 **pending:** 0
- **face:** 2 **flags:** 0x5012 **pending:** 0 **local:** 0.0.0.0:63000
- **face:** 3 **flags:** 0x5010 **pending:** 0 **local:** 0.0.0.0:63000
- **face:** 4 **flags:** 0x4042 **pending:** 0 **local:** [::]:63000
- **face:** 5 **flags:** 0x4040 **pending:** 0 **local:** [::]:63000
- **face:** 15 **flags:** 0xc **pending:** 0 **activity:** 2376
- **face:** 111 **flags:** 0x1014 **pending:** 0 **activity:** 10 **remote:** 127.0.0.1:43728
- **face:** 112 **flags:** 0x1014 **pending:** 0 **activity:** 3 **remote:** 127.0.0.1:43729
- **face:** 113 **flags:** 0x1014 **pending:** 0 **activity:** 1 **remote:** 127.0.0.1:43730
- **face:** 117 **flags:** 0x1014 **pending:** 0 **activity:** 651 **remote:** 127.0.0.1:43738
- **face:** 120 **flags:** 0x1014 **pending:** 11 **activity:** 641 **remote:** 127.0.0.1:43741

Face Activity Rates

	Bytes/sec In/Out	recv data/intr sent	sent data/intr recv
face: 0	5825 / 3195	3 / 3	0 / 0
face: 15	0 / 1599	0 / 4	0 / 0
face: 111	0 / 0	0 / 0	0 / 0
face: 112	0 / 0	0 / 0	0 / 0
face: 113	0 / 0	0 / 0	0 / 0
face: 117	972946 / 97088	209 / 211	3 / 3
face: 120	87805 / 970777	0 / 0	209 / 211

Forwarding

Figura 23 - Janela de estatísticas na fase do "`make test`" no fim dos testes

A seguir mostram-se algumas imagens de teste de inserção de conteúdos num repositório, listagem do seu conteúdo em determinado momento da experiência e a obtenção de ficheiros solicitados ao repositório.

Para experimentar a utilização de repositório, procedeu-se a algumas experiências.

Antes de iniciar é necessário definir algumas variáveis:

```
ccninitkeystore -f -v 365
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
export CCND_DEBUG=221
export CCNR_DIRECTORY=/home/jbarbosa/CCN_shared
export CCND_LOG=/home/jbarbosa/ccnd.log
```

Criar um local para montar o repositório:

```
mkdir -p /home/jbarbosa/CCN_shared
```

Caso não esteja já em execução será necessário executar o *Daemon* `ccnd`

```
sudo ccndstart &
sudo sleep 5
```

Após o arranque do Daemon ccnd pode-se visualizar a janela de estatísticas em `http://localhost:9695`

Caso o seu sistema não possua uma interface gráfica (GUI) pode utilizar o seguinte comando para visualizar as estatísticas no terminal.

```
ssh -L 9695:127.0.0.1:9695 jbarbosa@TECRA-A9
```

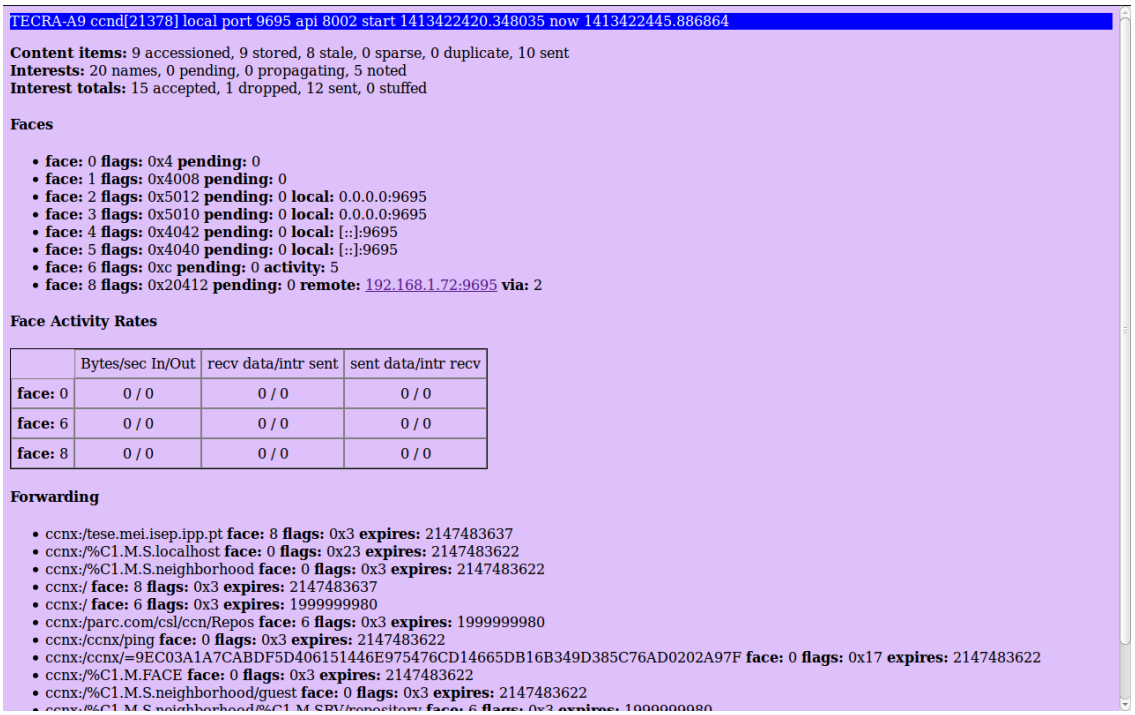


Figura 24 - Janela de estatísticas após inicio do "ccnd"

Executar o repositório e configurá-lo:

```
sudo ccnr &
sleep 10
sudo echo add ccnx:/tese.mei.isep.ipp.pt/ udp 192.168.1.72 >
/home/jbarbosa/ccnd_route.conf
sudo echo add ccnx:/ udp 192.168.1.72 >> /home/jbarbosa/ccnd_route.conf
sudo ccndc -f /home/jbarbosa/ccnd_route.conf
```

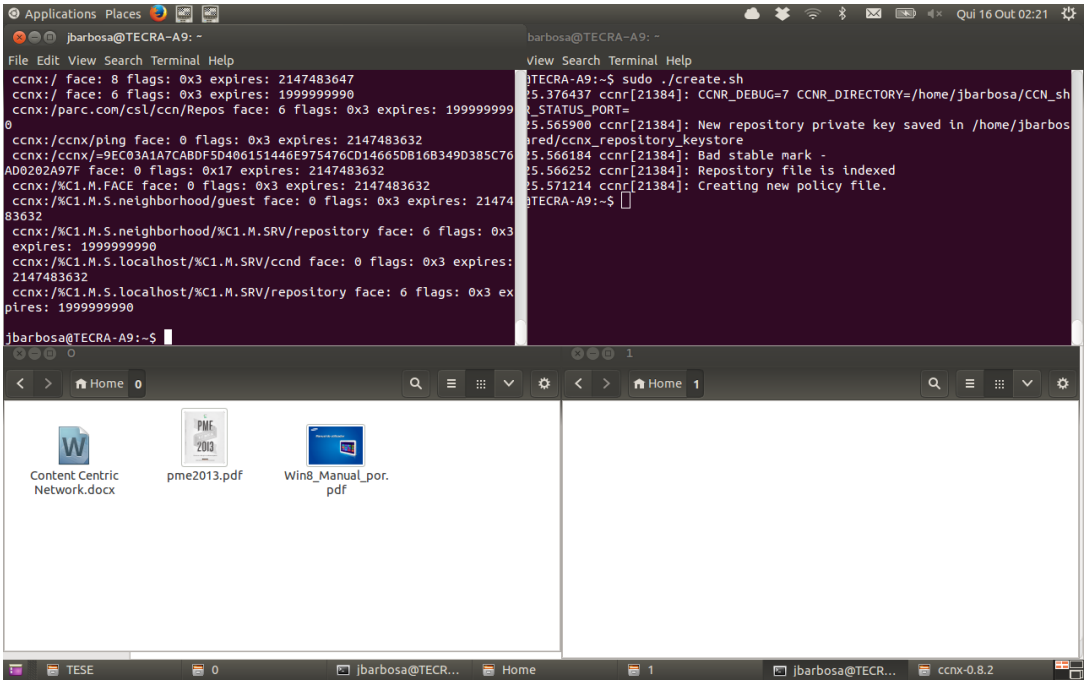


Figura 25 - Janela de exemplo de criação de repositório

Listar o conteúdo do repositório:

```
ccnlsrepo ccnx:/tese.mei.isep.ipp.pt
```

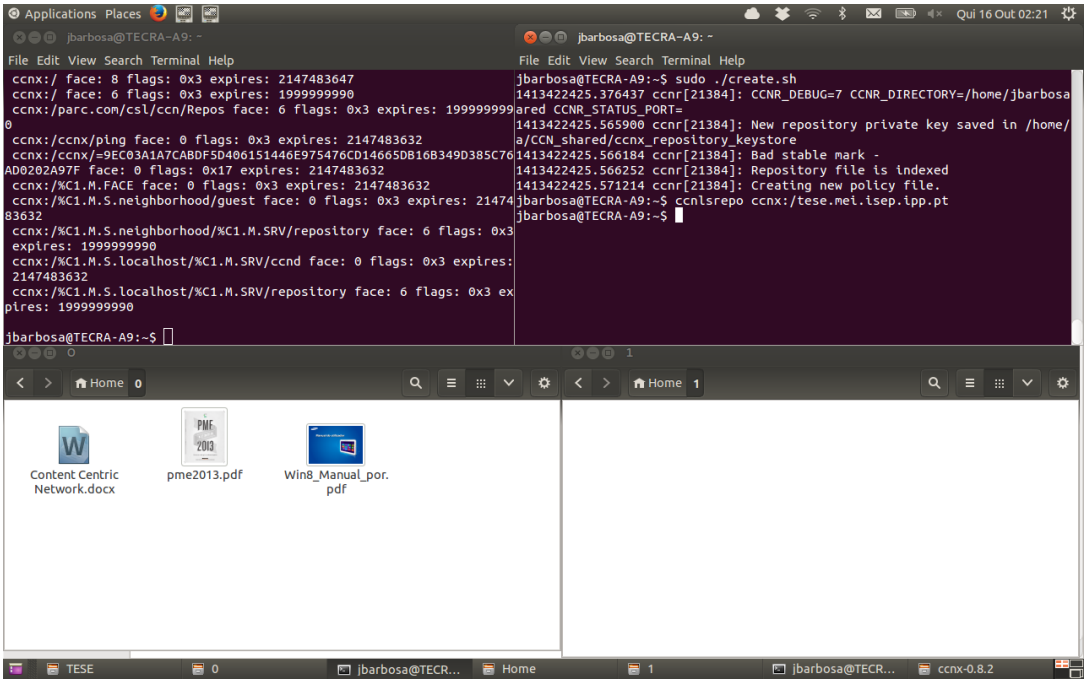


Figura 26 - Janela de exemplo de listagem de repositório

Adicionar conteúdo ao repositório:

```
ccnputfile ccnx:/tese.mei.isep.ipp.pt/Wind8_Manual_por.pdf
1/Win8_Manual_get.pdf
```

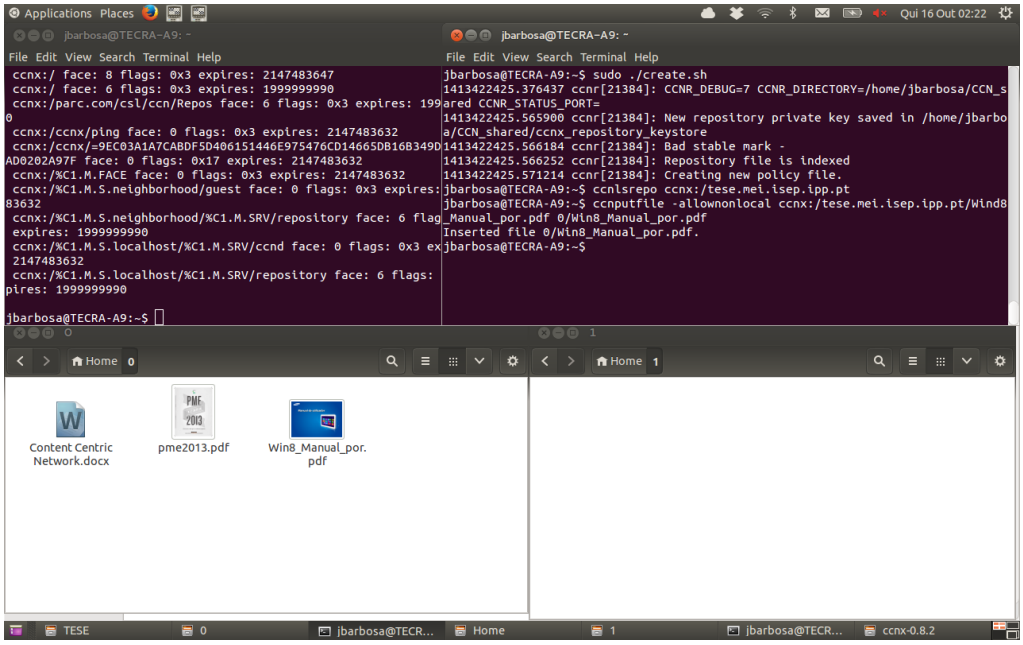


Figura 27 - Janela de exemplo de inserção de conteúdo no repositório

Listar o conteúdo do repositório:

```
ccnlsrepo ccnx:/tese.mei.isep.ipp.pt
```

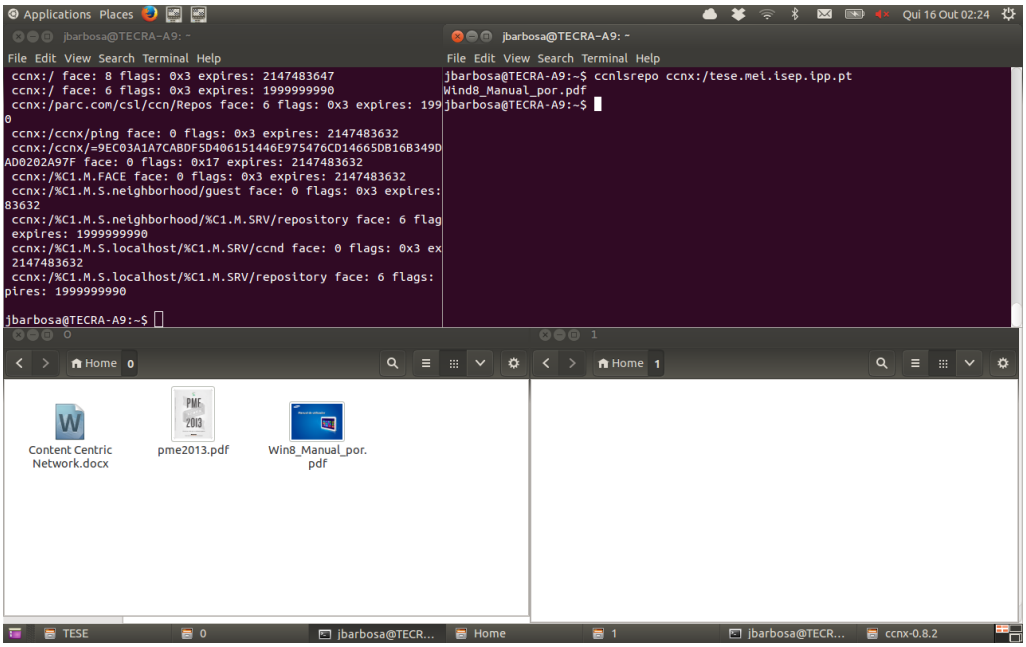


Figura 28 - Janela de exemplo de listagem de conteúdo inserido no repositório

Obter conteúdo do repositório:

```
ccngetfile ccnx:/tese.mei.isep.ipp.pt/Wind8_Manual_por.pdf
1/Win8_Manual_get.pdf
```

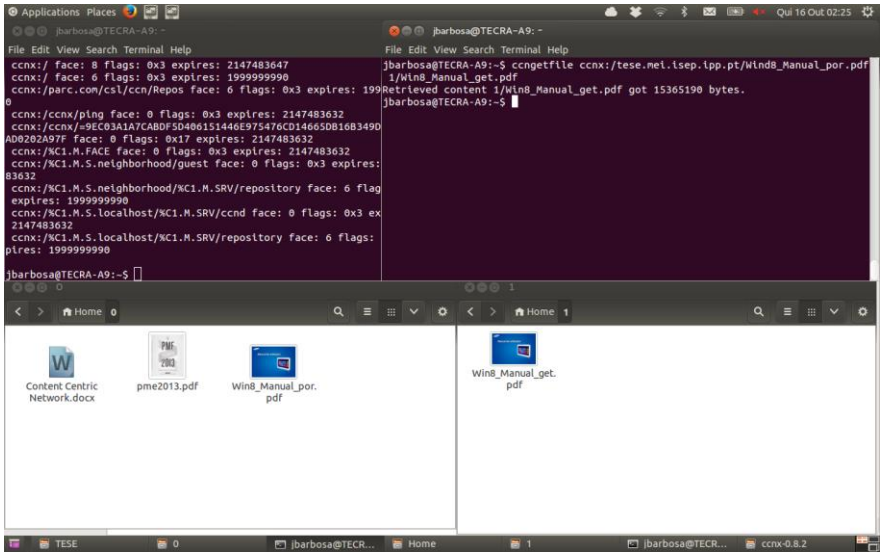


Figura 29 - Janela de exemplo de obtenção de conteúdo

Adicionar conteúdo ao repositório:

```
ccnputfile -allownonlocal ccnx:/tese.mei.isep.ipp.pt/pme2013.pdf
0/pme2013.pdf
```

Listar o conteúdo do repositório:

```
ccnlsrepo ccnx:/tese.mei.isep.ipp.pt
```

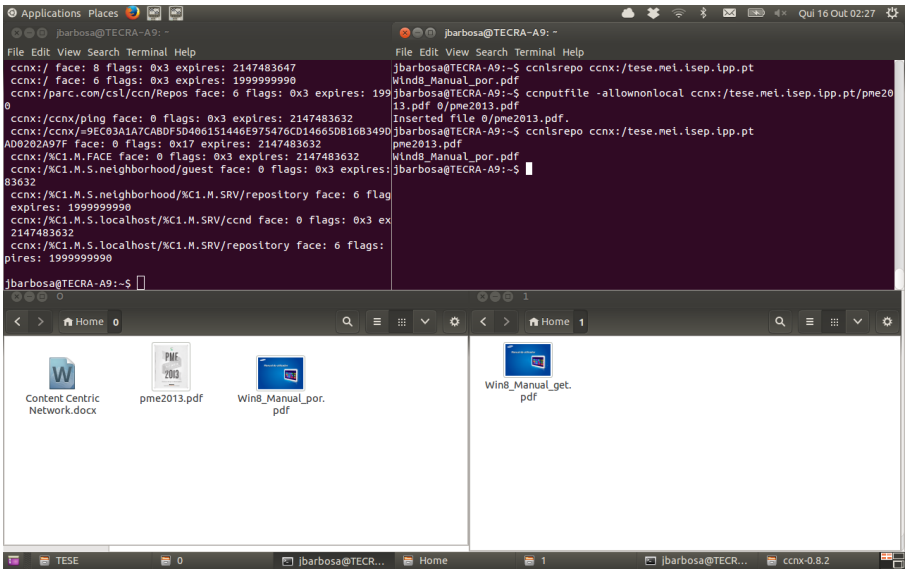


Figura 30 - Janela de exemplo de inserção e listagem do conteúdo no respositório

Listar o conteúdo do repositório:

```
ccnlsrepo ccnx:/tese.mei.isep.ipp.pt
```

Obter conteúdo do repositório:

```
ccngetfile ccnx:/tese.mei.isep.ipp.pt/pme2013.pdf 1/pme2013_get.pdf
```

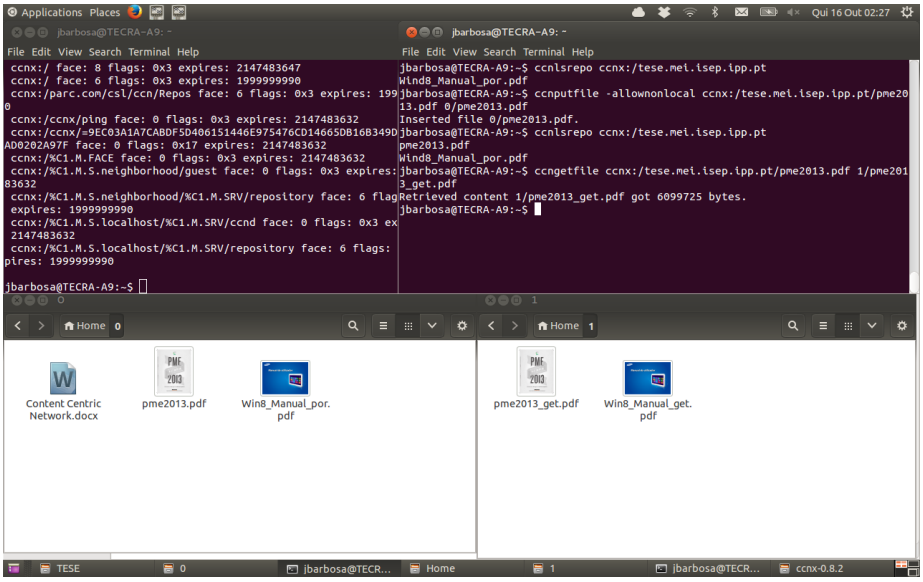


Figura 31 - Janela de exemplo de obtenção de novo conteúdo do repositório

Visualizar a janela de estatísticas em `http://localhost:9695`

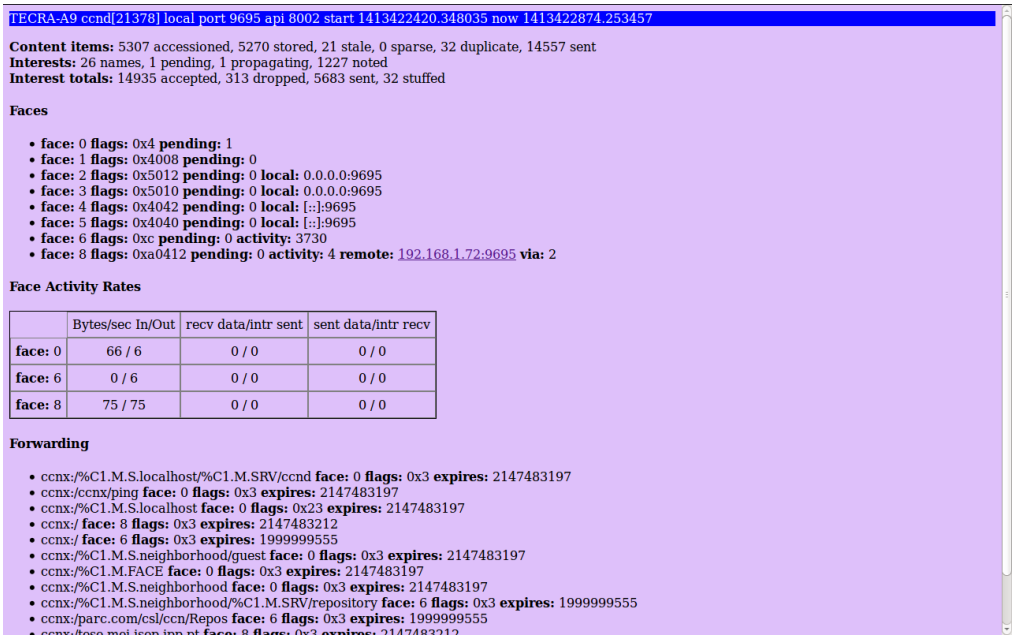


Figura 32 - Janela de exemplo de estatísticas no fim das operações anteriormente descritas

Agora iremos demonstrar de que forma, poderemos utilizar o CCNx através de interface GUI (*Graphic User Interface*) programada em JAVA, para por exemplo explorar o conteúdo de um repositório. O utilitário que permite esta utilização com interface gráfico é o `ccnexplore`.

O utilitário `ccnexplore` é um aplicativo experimental ainda em desenvolvimento. Este aplicativo explora `ContentObjects` disponíveis usando o aplicativo Java CCNx, `ContentExplorer` e tem como ponto de partida "/" para apresentar o conteúdo e começar a enumeração de nomes. O `ccnexplore` usa a `CCNNameEnumeration` para preencher o GUI e pode mostrar o conteúdo de ficheiros ".txt" num painel de visualização ou janela separada. O `ContentExplorer` também pode ser usado para armazenar ficheiros num repositório, utilizando o botão "**Save to Repo…**". Para substituir o ponto de partida padrão, execute-o com a opção `-root` com um `ccnxname`. O `ccnxeplore` destina-se a ser utilizado como um primeiro teste de funcionalidade `AccessControl` na CCN. O seu desenvolvimento ainda está num estado extremamente prematuro e será atualizado em versões futuras. O `ccnexplore` também pode ser usado para explorar toda a hierarquia de nomes dos objetos armazenados nos repositórios. Se o executar com a opção `-debugMode` todos os componentes `ContentName`, incluindo versões, segmentos e *digest*, serão mostrados.

Para o executarmos devemos abrir uma janela de terminal e correr o seguinte comando:

```
ccnexplore -root ccnx:/tese.mei.isep.ipp.pt –accessControl
```

Após a execução deste comando será mostrada a janela da Figura 33.

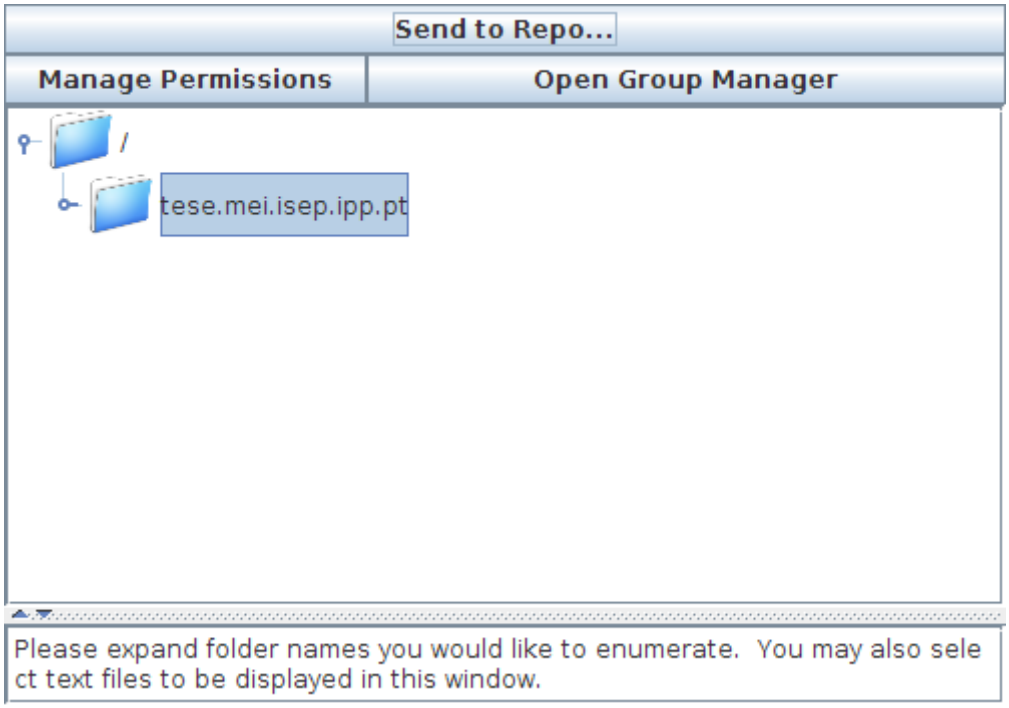


Figura 33 – Janela de interface gráfico de um repositório

Esta ferramenta permite visualizar e explorar o conteúdo de um repositório através de um interface gráfico programado em JAVA. Neste caso em concreto, executamos o comando `ccnexplore` para nos mostrar o conteúdo do nosso repositório com o nome que escolhemos anteriormente, conforme documentado nas experiências da criação de um repositório ou seja `tese.mei.isep.ipp.pt`, além disso também inclui no topo da janela um conjunto de botões com algumas funcionalidades que iremos demonstrar mais adiante.

Para visualizar o conteúdo do repositório `tese.mei.isep.ipp.pt`, basta efetuar duplo clique sobre o ícone do repositório que é representado sobre a forma de uma pasta, tal como são representados os diretórios em Linux em ambiente gráfico.

Após efetuar duplo clique no repositório `tese.mei.isep.ipp.pt`, é mostrado o seu conteúdo no momento desta operação, conforme a Figura 34.

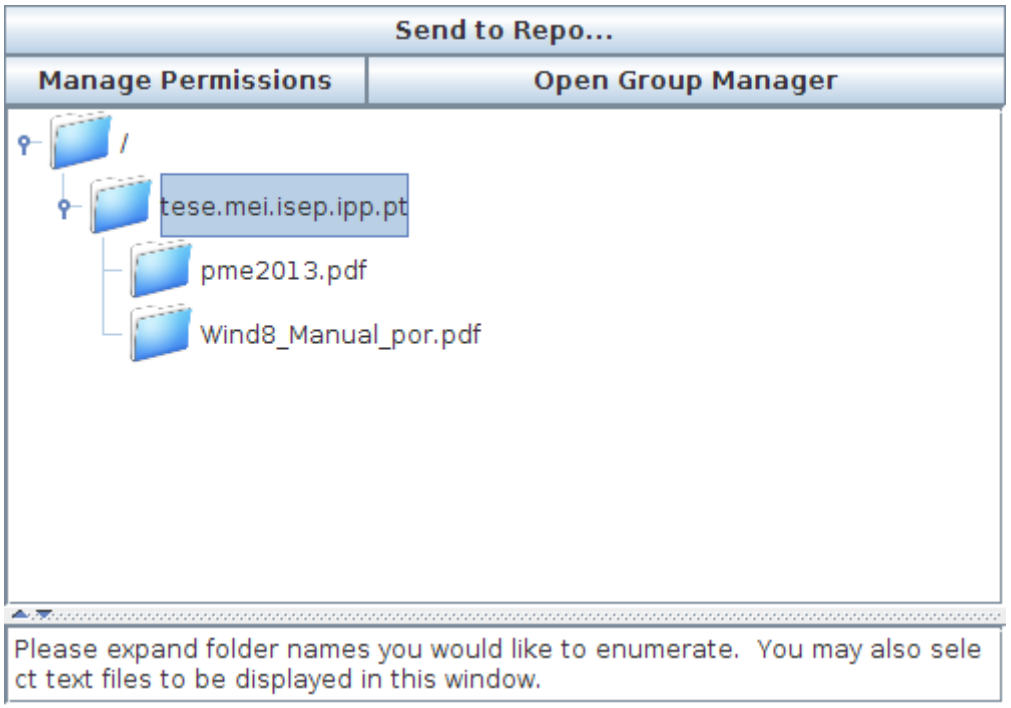


Figura 34 – Listagem do conteúdo do repositório

De facto os conteúdos listados na Figura 34 são efetivamente os ficheiros que adicionamos nas experiências anteriormente documentadas.

Esta interface gráfica também permite adicionar conteúdos ao repositório e para isso temos que utilizar o botão "**Send to Repo…**" que se encontra no topo da janela.

Ao clicarmos no botão "**Send to Repo…**" abre-se a janela da Figura 35 que nos permite selecionar a localização e escolher o ficheiro que pretendemos adicionar ao repositório.

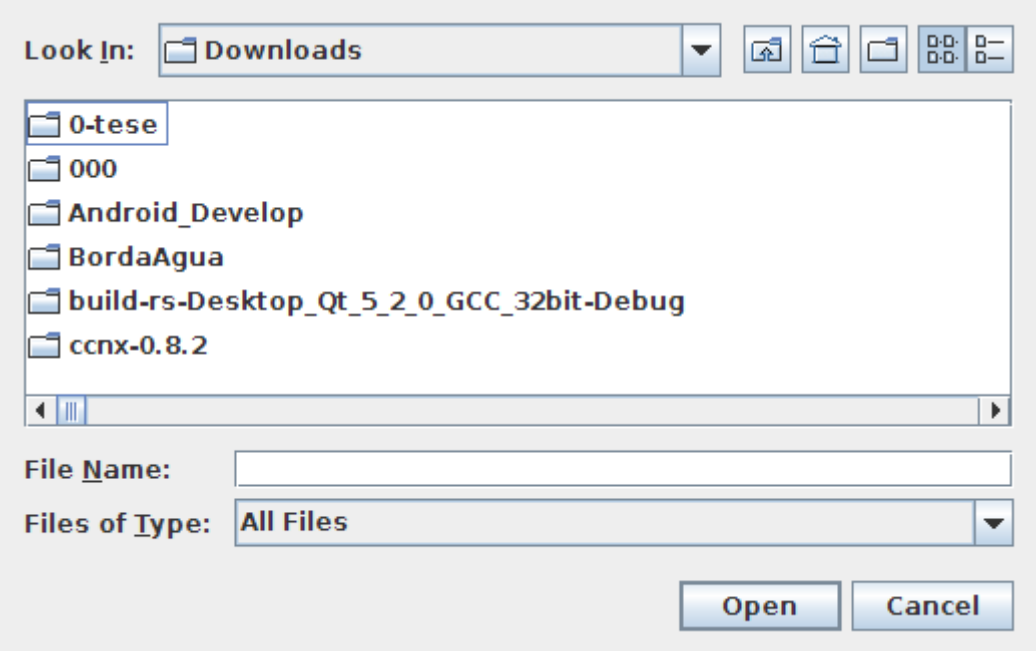


Figura 35 – Janela mostrada quando se clica no botão "**Send to Repo…**"

Seguidamente escolhemos qual a localização e o ficheiro a adicionar ao repositório conforme a Figura 36.

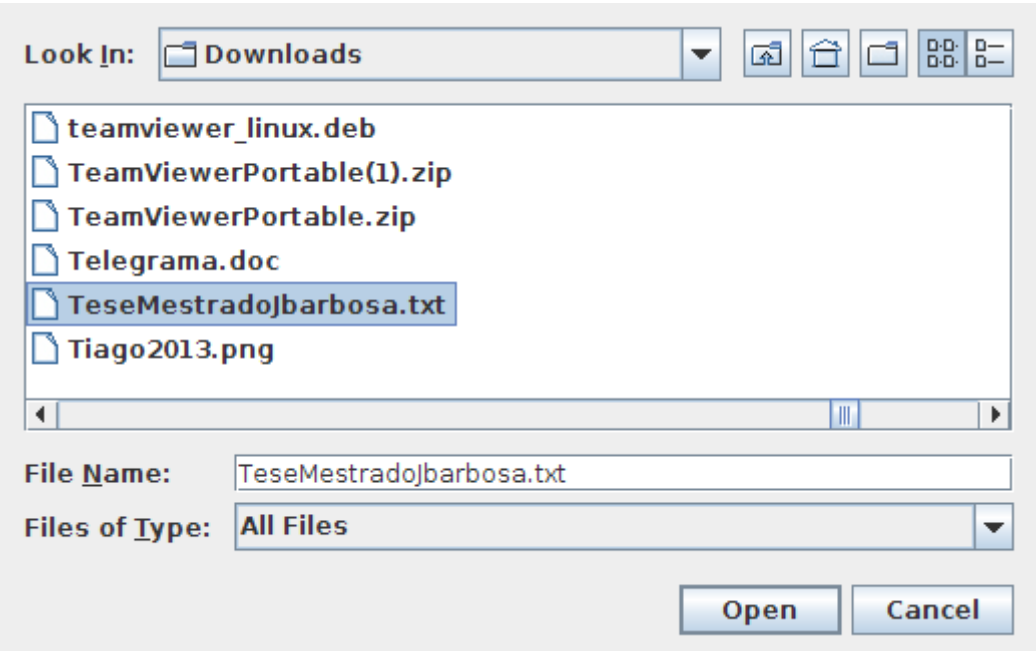


Figura 36 – Escolher ficheiro a adicionar ao repositório

E clicamos em “**Open**”.

Seguidamente aparece-nos a janela da Figura 37, dando-nos a possibilidade de dar um nome ao conteúdo a adicionar que poderá ser diferente, do nome original do ficheiro que temos guardado localmente no nosso computador.

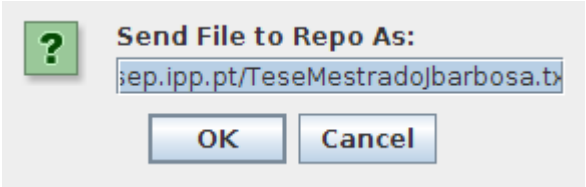


Figura 37 – Escolha do nome para publicação do conteúdo no repositório

No caso de não querermos alterar nada, basta clicar no botão “**OK**” e o conteúdo é adicionado e listado automaticamente conforme mostra a Figura 38.

Figura 38 – Listagem após inserção do ficheiro TeseMestradoJbarbosa.txt

Esta ferramenta possibilita também a pré-visualização de ficheiros de texto, bastando para isso apenas clicar uma vez sobre o ficheiro de texto que se pretende visualizar. Assim não foi

aleatória a razão pela qual acabamos de adicionar, no exercício anterior, um ficheiro de texto, mas sim especificamente com o propósito de o demonstrar agora.

Se clicarmos sobre um ficheiro de texto esta ferramenta permite-nos pré-visualizar o seu conteúdo conforme apresenta a Figura 39.

Figura 39 – Pré-visualização de um ficheiro de texto.

Esta ferramenta também permite descarregar ficheiros do repositório para a nossa máquina local. Para isso clicamos com a tecla direita do rato sobre o ficheiro desejado e escolhemos a opção “Save File” conforme apresentado na Figura 40.

Figura 40 – Descarregar um ficheiro do repositório para uma pasta local

Além disso também podemos ver a versão do ficheiro selecionado conforme mostra a figura 41.

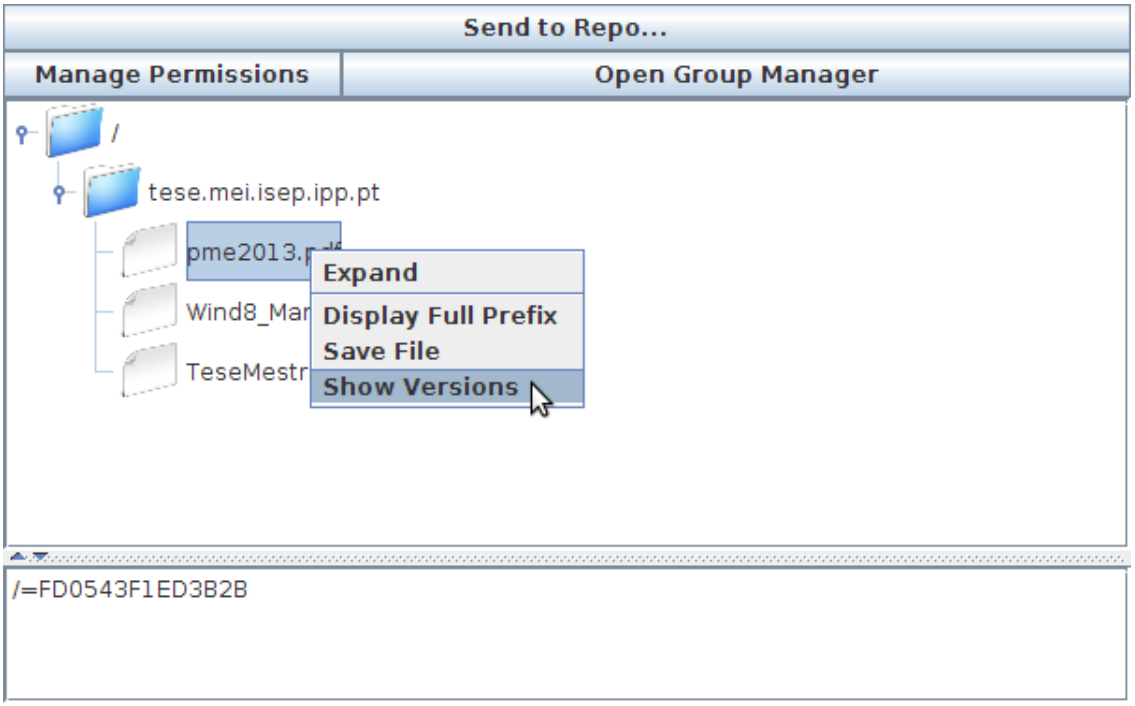


Figura 41 – Ver a versão do conteúdo selecionado

Podemos também ver qual é o caminho completo do ficheiro selecionado dentro do repositório conforme apresenta a Figura 42.

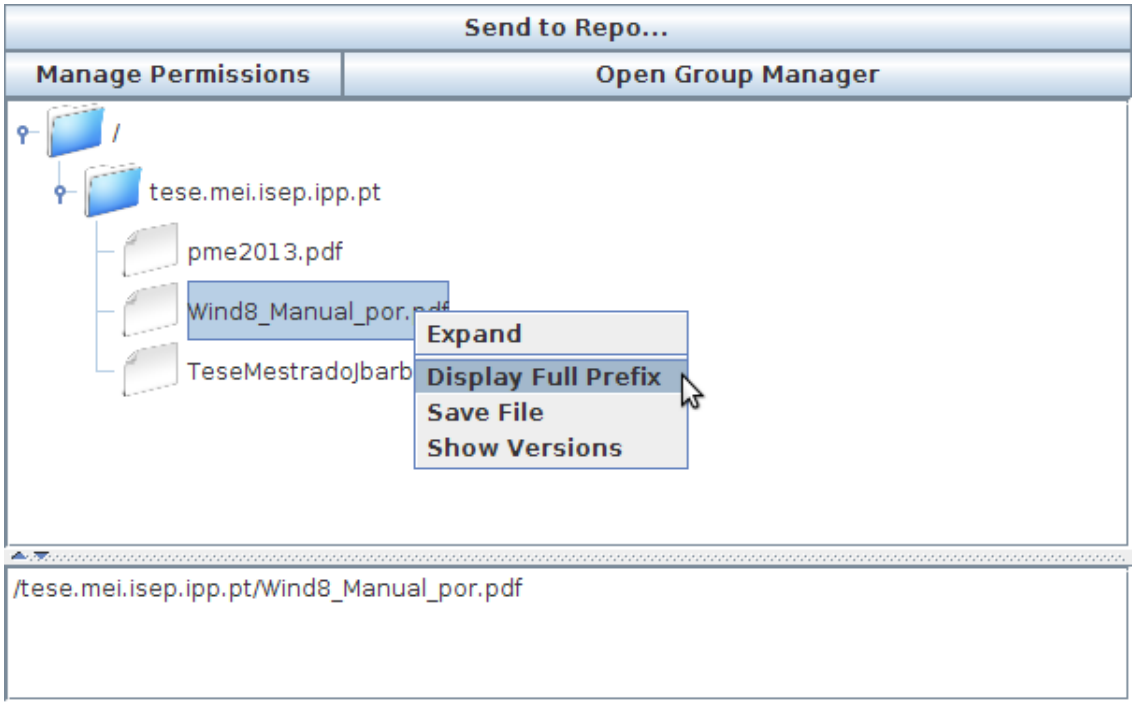


Figura 42 – Apresentação do caminho completo dentro do repositório.

Agora vamos apresentar a gestão de grupos e para isso vamos utilizar o botão "**Open Group Manager**" e será mostrada a Figura 43.

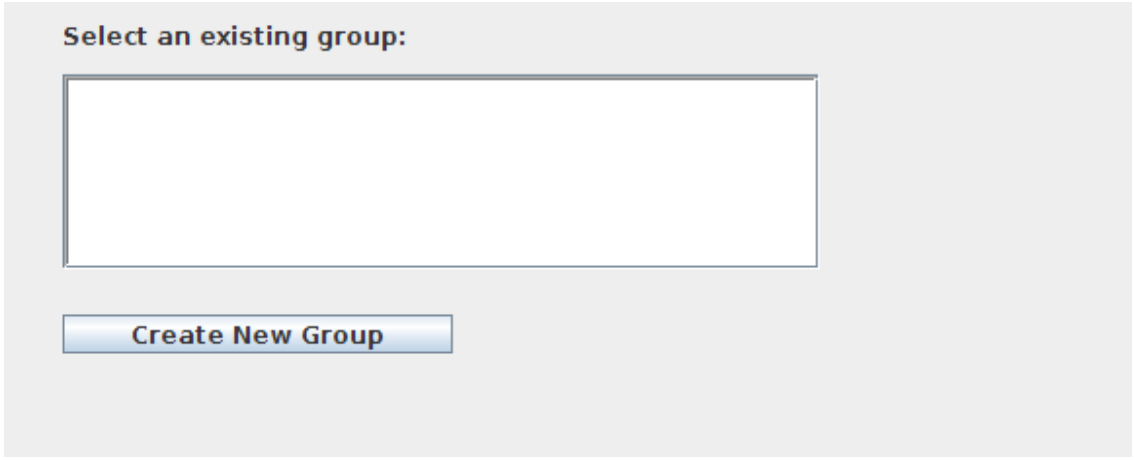


Figura 43 – Janela do "Group Manager"

Como não existe nenhum grupo vamos criar um e para isso clicamos no botão "**Create New Group**" sendo mostrada a Figura 44.

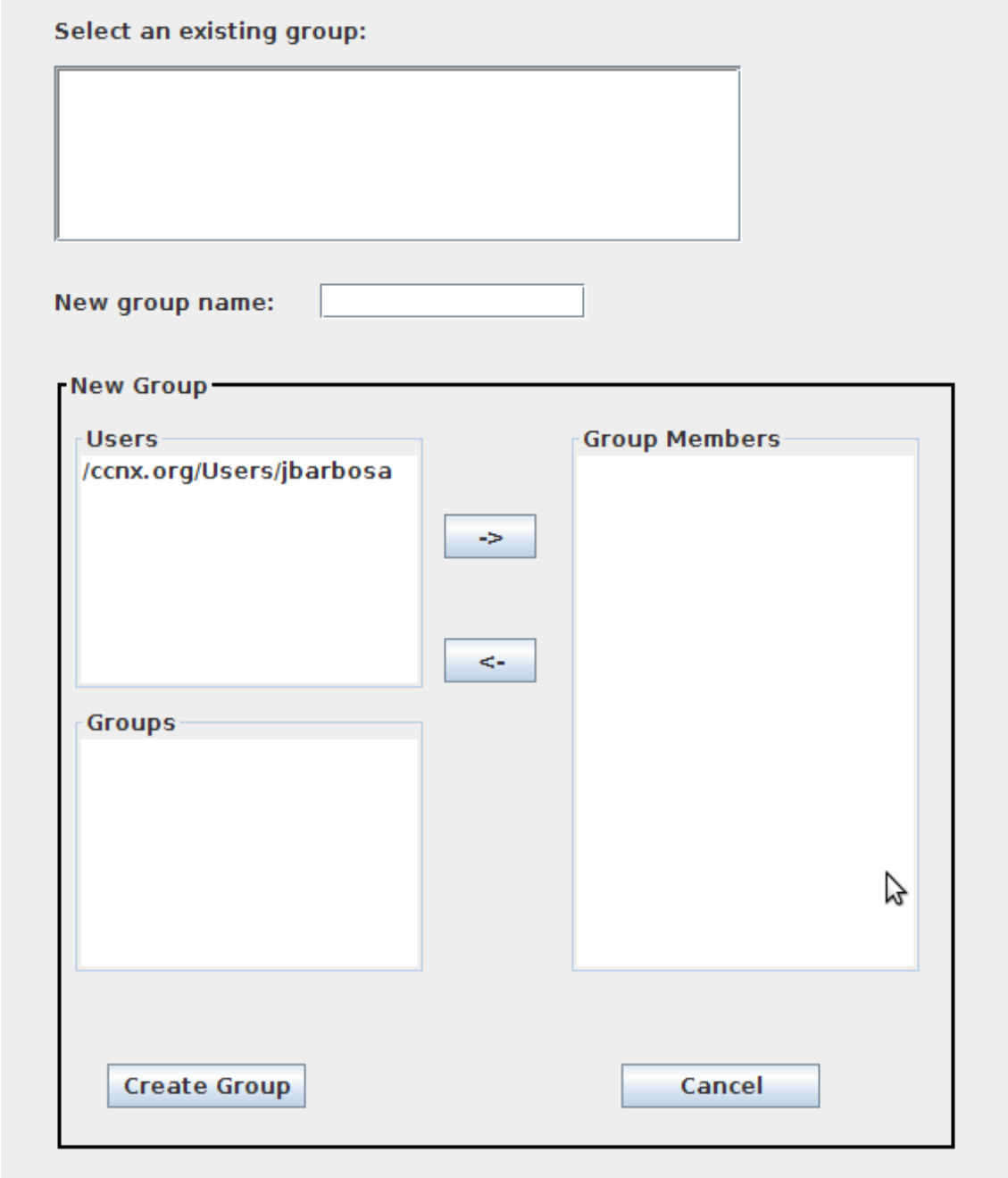


Figura 44 – Criar novo grupo

Digitamos "Teste" no nome do novo grupo a criar na caixa de texto, disponibilizada para esse efeito, e clicamos no botão "**Create Group**".

Tal como é referido na descrição do manual deste utilitário, o aplicativo ainda se encontra numa fase muito prematura de desenvolvimento, pelo que durantes as nossas experiências laboratoriais constatamos que após o passo de clicar no botão criar grupo a aplicação volta

para a janela da Figura 43, e assim sendo não possibilita a continuação da experiência. Também o botão *"Manage Permissions"* ainda não faz nada. Tal como referido no manual este utilitário será alvo de atualizações e esperamos que em versões futuras essas funcionalidades já se encontrem totalmente implementadas.

Para parar o *Daemon*:

```
sudo ccndstop
```

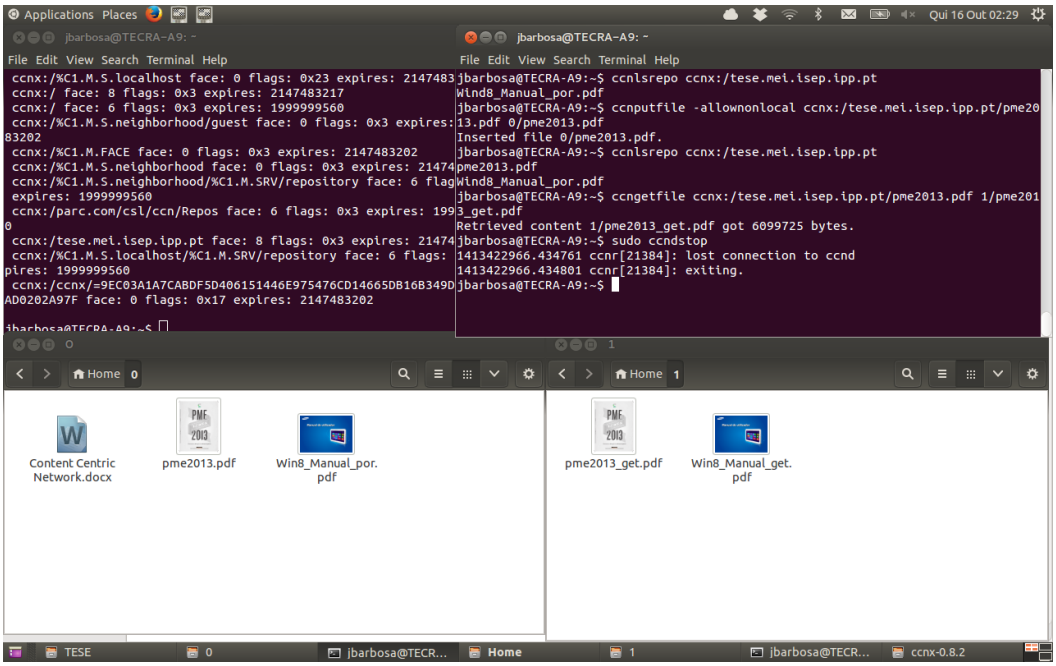


Figura 45 - Janela de exemplo para terminar o "ccnd"

6.4 Considerações Finais

Atualmente, sensivelmente desde abril de 2014, o CCNx está a ser desenvolvido na versão 1.0 mas ainda não está disponível para download. Apenas está a ser publicada a documentação onde estão a ser introduzidas as alterações.

Atualmente foi efetuado um *fork* à versão 0.7x do CCNx com o nome de NDNx. Esta bifurcação do *software* resulta da evolução das nomenclaturas que estão a ser utilizadas para abordar este tipo de redes centradas na informação (ICN) como é o caso da CCN, que inicialmente começou por se chamar *"Content Centric Networking"*, mais tarde apareceu sobre a designação de *"Netwok Named Content"* e atualmente o projeto gerido pelo Palo Alto Research Center (PARC) esta a ser frequentemente designado por *"Named Data Networking"*.

O projeto *Named-Data Networking* - NDN, mantém o desenvolvimento da CCN e estabelece diversas áreas de investigação, onde uma delas é a segurança. Apesar de a CCN não fornecer uma solução para todos os desafios em segurança de redes, estabelece novos mecanismos para a validação de conteúdos e a determinação da sua proveniência.

7 Conclusões e trabalho futuro

Hoje em dia a utilização das redes, nomeadamente a internet, centra-se à volta da movimentação de conteúdos, mas estas redes que utilizamos não foram pensadas, nem desenhadas, para esta nova realidade, pois elas continuam a funcionar baseadas em conversações "*host-to-host*".

Nas abordagens de redes centradas na informação (ICN), conforme ilustrado na Figura 34, o que é proposto é o desenvolvimento de redes, cujo foco centra-se na informação propriamente dita em vez de se centrar nos hosts como acontece atualmente, tendo como preocupações principais a segurança, a escalabilidade, a eficiência, a qualidade e o suporte às comunicações em tempo real. Este tipo de abordagens pode ser classificado como clean slate, uma vez que redesenham totalmente a maneira, atualmente, como efetuamos a troca de informação na Internet. A principal função deste tipo de redes é interligar em grande escala os produtores e os consumidores de conteúdos, aumentando assim a eficiência no acesso e na proliferação da informação.

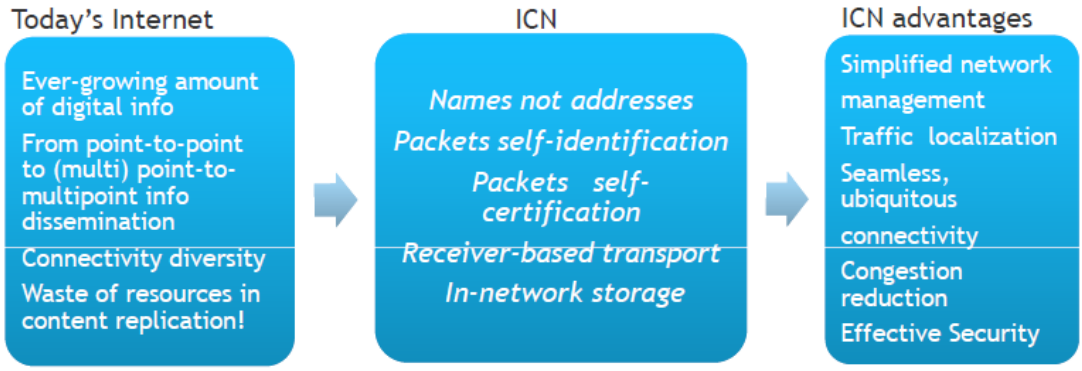


Figura 46 – Abordagem da ICN ((retirado de [Carofiglio, 2012])

A CCN é um dos vários modelos de arquiteturas abordadas pela ICN, com uma arquitetura de rede construída com base nos princípios de engenharia do IP, conforme demonstrado no capítulo 2, subcapítulo 2.3 Modelo de camadas, mas utilizando conteúdos com nomes em vez de identificadores para hosts. O resultado retém a simplicidade e escalabilidade do IP mas aposta numa eficiência de disponibilidade e uma maior tolerância a interrupções. A CCN está concebida para substituir a atual arquitetura de redes IP, mas a sua implementação, pode ser efetuada de forma incremental como uma camada sobre este, o que faz com que as vantagens em termos funcionais fiquem acessíveis para as aplicações sem uma obrigatoriedade de implementação global.

Este modelo de arquitetura, de rede, tem sido amplamente debatida por investigadores nos últimos anos. A escolha do paradigma orientado ao conteúdo, onde utilizadores solicitam conteúdos pelo seu nome e compete à rede procurá-los onde quer que eles se encontrem, faz com que este tipo de arquitetura seja adaptado ao novo perfil de utilizadores e das aplicações. Por forma a conseguir uma maior eficiência no acesso a conteúdos, os routers da CCN guardam em cache os conteúdos que forem pedidos, permitindo desta forma que pedidos futuros, do mesmo conteúdo, não precisem ser encaminhadas até a fonte, conseguindo-se assim que o tempo de espera da resposta e a utilização da largura de banda seja bastante reduzido.

As redes centradas na informação têm como grande desafio definir um padrão para a nomenclatura dos conteúdos de uma forma que seja eficiente e consistente simultaneamente. Existe também o desafio, de ter de se encontrar, uma ou mais soluções para a sua localização e de que forma efetuar o encaminhamento dos conteúdos, tendo por base os nomes atribuídos à informação. Pode-se concluir que o desfio principal, com que se defronta este tipo de redes, prende-se com a idealização de um sistema de resolução de nomes, uma vez que os restantes ou seja a segurança, a escalabilidade e o encaminhamento este estritamente dependentes do esquema de atribuição de nomes que venha a ser escolhido.

A CCN utiliza um esquema de nomenclatura de conteúdos de uma forma hierárquica e desta forma consegue eliminar a associação entre os conteúdos e a sua localização na rede. Ao efetuar a remoção da associação entre os conteúdos e a sua localização física, consegue-se que a segurança dos conteúdos seja possível de implementar de forma mais fácil. Ao contrário daquilo que faz o modelo atual de redes IP, que tenta garantir a autenticidade e integridade do nó que armazena esse conteúdo e do canal utilizado por essa comunicação, a CCN define que todos os conteúdos têm que ser assinados pelos seus emissores. Com isto, se necessitarmos de verificar a autenticidade e a integridade de um conteúdo, apenas necessitamos de obter a chave pública do emissor para efetuarmos a verificação da assinatura do conteúdo.

Um dos principais problemas na Internet, segundo [Ribeiro et al., 2012], são os ataques DDoS. O modelo orientado à localização, no qual as redes TCP/IP estão implementadas, é bastante vulnerável a este tipo de ataque. Por outro lado, características como a agregação e o encaminhamento dinâmico de pacotes *"Interest"*, tornam a CCN bastante resistente aos ataques DDoS frequentemente utilizados contra a Internet dos dias de hoje. Apesar de permitir a implementação da segurança de conteúdos de forma mais simples e de mitigar grande parte dos ataques DDoS efetuados atualmente na Internet, a CCN não consegue impedir que alguns de seus mecanismos sejam explorados por ataques projetados especialmente para arquitetura TCP/IP. Além disso, ataques fortemente estudados envolvendo caches, como a cache *snooping*, passam a ter um maior impacto na CCN em comparação com a Internet atual.

O projeto *Named-Data Networking* - NDN, mantém o desenvolvimento da CCN e estabelece diversas áreas de investigação, onde uma delas é a segurança. Apesar de a CCN não fornecer

uma solução para todos os desafios em segurança de redes, estabelece novos mecanismos para a validação de conteúdos e a determinação da sua proveniência.

Contudo, segundo [Zhang et al. 2010], conteúdos maliciosos ou indesejados que contenham assinaturas válidas, podem ser gerados por emissores legítimos. Para garantir a autenticidade da relação entre um emissor de conteúdos e a sua chave pública, a CCN necessita que seja implementado algum mecanismo de gestão de confiança. Apesar dos utilizadores serem livres para utilizar quaisquer modelos existentes, o projeto NDN sugere a utilização do sistema SDSI/SPKI.

Neste contexto, uma questão importante é o processo de revogação de chaves. Uma vez que as chaves públicas também são tratadas como conteúdos, estas poderão ser armazenadas em caches na rede. Assim, quando um consumidor obtiver uma chave, este não saberá se a mesma é atual, ou se é uma cópia revogada obtida de alguma cache.

O modo de funcionamento da CCN requer a manutenção de estado nos seus *routers*. Esta característica pode ser vista como uma vulnerabilidade que pode ser explorada por atacantes para efetuar um ataque DoS por inundação de pacotes *"Interest"*. Este tipo de ataques, podem provocar o esgotamento dos recursos dos *routers* impedindo o atendimento de solicitações de utilizadores legítimos.

Tal como atualmente a Internet, a CCN não fornece um mecanismo de responsabilização para os consumidores de conteúdos. A responsabilização é definida como "a propriedade que garante que as ações de uma entidade possam ser rastreadas de forma exclusiva até esta entidade"[Shirey 2000]. Assim, apesar dos pacotes "Data" serem assinados pelos emissores, os pacotes *"Interest"* podem ser emitidos por qualquer utilizador com a garantia do anonimato. Como os pacotes *"Interest"* podem ser usados como veículos nos ataques de negação de serviço, a fonte deste tipo de ataques torna-se ainda mais difícil de rastrear do que na Internet atual. Esta dificuldade poderá ser resolvida se exigirmos que os pacotes *"Interest"* também sejam assinados. Porém, esta técnica resulta em falhas de privacidade. E assim sendo, cria-se um braço de ferro entre a problemática da detecção de ataques DoS e a garantia da privacidade dos utilizadores.

Assim, podemos concluir que a CCN representa uma grande evolução em relação a Internet atual, principalmente no que diz respeito a segurança. Por conseguinte, novas tecnologias normalmente são acompanhadas por novas vulnerabilidades, que por sua vez podem originar novos ataques. E conforme foi demonstrado ao longo deste estudo, a CCN não é uma exceção e a sua adoção em larga escala depende ainda da superação de diversos desafios, nomeadamente no que concerne à segurança.

Resumidamente a CCN focaliza-se nos nomes de conteúdo em vez de nomes de *hosts*. Introduz uma melhor segurança, eficiência na entrega, a mobilidade e tolerância à interrupção do TCP/IP. A CCN pode ser incrementalmente implementável. Tem disponível o CCNx que é

fruto da investigação de uma grande comunidade e as suas ferramentas para a experimentação.

O objetivo da CCN poderá ter como efeitos colaterais a transformação da Internet num serviço público de divulgação de dados e que seja ISP amigável em oposição ao P2P.

Pode-se concluir também que as empresas de difusão de conteúdo não vão apoiar a CCN. Pois essas empresas dependem dos dados valiosos que todos nós colocamos nos seus servidores. Sendo assim evidente que essas empresas não têm interesse, absolutamente nenhum, em devolver o poder à rede.

Os fornecedores de internet (ISP) precisarão de fortes incentivos para fomentarem a implementação deste tipo de arquitetura nas suas redes, como por exemplo a relação entre os custos de largura de banda vs custo de implementação de nós de armazenamento. Além disso a maioria dessas empresas já têm implementadas soluções de cache e como é do conhecimento geral as empresas não gostam de mudanças e investimentos.

Relativamente à posição académica, podemos ponderar as seguintes questões:

Será a inércia da Internet muito alta para uma mudança natural?

Deverá a investigação académica apoiar ativamente a CCN ou seja a construção da primeira Rede WAN CCN?

Qual é a principal força evolutiva para CCN? Quer do ponto de vista Ético, quer do ponto de vista Técnico, no que concerne a direitos de autor.

De um modo geral podemos também todos nós, em termos futuros, meditar sobre a temática e questionarmo-nos sobre:

Que tipo de Internet é que nós queremos?

Será a do modelo atual, limitado a

1. Conversas com servidores que mantém mesmo conteúdo privado?
2. Ineficiente e com comunicações P2P inseguras?

De que forma podemos tornar livre a difusão de conteúdos, sem por em causa a escalabilidade e a segurança?

Apesar da CCN ter vindo a ser analisada de forma extensiva, a verdade é que o verdadeiro impacto que terá na prática não foi ainda completamente explorado. O propósito e o contributo desta dissertação foi o de efetuar um estudo das principais propostas realizadas nesta área e desenvolver um entendimento dos principais desafios que se colocam na utilização de redes centradas na informação.

Em termos futuros podemos afirmar que há ainda muito trabalho duro a fazer.

Referências

[Adamson et al. 2008	Adamson, B., Bormann, C., Handley, M. and Macker., J., Multicast Negative-Acknowledgement (NACK) Building Blocks. IETF, November 2008. RFC 5401.
[Felten and Schneider 2000]	Felten, E. W. e Schneider, M. A., 2000. Timing attacks on web privacy. Em ACM Conference on Computer and Communications Security, p. 25–32.
[Ghodsi et al. 2011]	Ghodsi, A., Koponen, T., Rajahalme, J., Sarolahti, P. e Shenker, S.,2011b. Naming in Content-Oriented Architectures. In ACM SIGCOMM workshop on Information-centric networking, p. 1–6.
[Carofiglio, 2012	Carofiglio, G., August 22nd, 2012, ASIA FI Summer school
[Cholez, 2012]	Cholez, T., Introduction to Content Centric Networking and the CCNx framework, Lab of the 6th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2012), Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg, 2012
[Jacobson et al. 2009]	Jacobson, V., Smetters, D., Briggers, N., Plass, M., Stewart, P., Thornton, J. and Braynard, R., VoCCN: Voice Over Content-Centric Networks, ACM ReArch'09, December 2009.
[Jacobson et al. 2009]	Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., Braynard, R., (PARC) Networking Named Content, CoNEXT 2009, Rome, December, 2009.
[Martins and Alberti, 2011]	Martins, B., Alberti, A., Redes Centradas na Informação: Uma Comparação de Abordagens, Instituto Nacional de Telecomunicações - INATEL, Brazil, 2011
[Neuman et al. 1993]	Neuman, C., Kohl, J., Yu, T., Hartman, S. e Raeburn, K., 1993. The Kerberos Network Authentication Service (V5). Technical Report.
[Perino and Varvello, 2011]	Perino, D., Varvello, M., A reality check for content centric networking, ACM Sigcomm Workshop on Information-Centric Networking (ICN) 2011
[Pournaghshband and Natarajan, 2011]	Pournaghshband, V. and Natarajan, K. In International Conference on Security and Management – SAM, July 2001.
[Ribeiro et al., 2012]	Ribeiro, I., Guimarães, F., Kazienko, J., Rocha, A., Velloso, P., Moraes, I., Albuquerque, C., Cap. 3 - Segurança em Redes Centradas em Conteúdo: Vulnerabilidades, Ataques e Contramedidas, Minicursos do XII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais — SBSeg 2012, Instituto de Computação, PGC-TCC, Universidade Federal Fluminense, 2012
[Severence, 2013]	Severance, C., Van Jacobson: Content-Centric Networking, Computer , vol.46, no.1, p.11,13, Jan. 2013
[Shirey, 2000]	Shirey, R.,2000. Internet Security Glossary.
[Smetters and Jacobson 2009]	Smetters, D. e Jacobson, V., 2009. Securing Network Content. Technical Report TR-2009-1, Xerox Palo Alto Research Center - PARC.
[Wilcox-O'Hearn 2003]	Wilcox-O'Hearn, Z., 2003. Names: Decentralized, secure, human-meaningful: Choose two. http://web.archive.org/web/20011020191610/http://zooko.com/distnames.html [último acesso: Out. 2014]
[Yi et al. 2012]	Yi, C., Afanasyev, A., Wang, L., Zhang, B. e Zhang, L., 2012. Adaptive Forwarding in Named Data Networking. ACM SIGCOMM Computer Communication Review, 42(3): p. 62–67.
[Zhang et al. 2010]	Zhang, L., Estrin, D., Bruke, J., Jacobson, V., Thornton, J., Smetters, D., Zhang, B., Tsudik, G., Claffy, K., Massey, D., Papadopoulos, C., Abdelzaher, T., Wang, L., Crowley, P. e Yeh, E., 2010. Named Data Networking (NDN) Project. Technical Report NDN-0001, NDN.
IETF, 1998	IETF. RFC 2328 – OSPF Version 2, https://www.ietf.org/rfc/rfc2328.txt [último

acesso: Out. 2014].

IETF, 2004 — IETF. RFC 3787 – Recommendations for Interoperable IP Networks using Intermediate System to Intermediate System (IS-IS), https://tools.ietf.org/html/rfc3787 [último acesso: Out. 2014].

IETF, 2007 — IETF. RFC 4971 – Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information, https://tools.ietf.org/html/rfc4971 [último acesso: Out. 2014]

Pablo Rodriguez, 2010 — http://www.rodriguezrodriguez.com/content-centric-networking/ [último acesso: Out. 2014].

PARC, 2014 — https://www.parc.com/work/focus-area/content-centric-networking/ [último acesso: Out. 2014].

Project CCNx, 2014 — http://www.ccnx.org [último acesso: Out. 2014]. https://www.ccnx.org/wiki/CCNx/CCNxPresentations [último acesso: Out. 2014]. http://redmine.ccnx.org/projects/ccn [último acesso: Out. 2014].

Project Named Data, 2014 — http://named-data.net/ [último acesso: Out. 2014].

XCONOMY, 2012 — http://www.xconomy.com/san-francisco/2012/08/07/the-next-internet-inside-parcs-vision-of-content-centric-networking/?single_page=true [último acesso: Out. 2014]

Zooko O'Whielacronx, 2010 — http://www.eros-os.org/pipermail/cap-talk/2010-December/014336.html [último acesso: Out. 2014].