Information-Centric Networking: A Implementação através de *CCNx Project*

André Diegues - 201206858
Fábio Teixeira - 201305725
Tópicos Avançados em Redes - CC4037
Departamento de Ciencia de Computadores
Faculdade de Ciencias da Universidade do Porto

22 de Dezembro de 2016

Tabela de Conteúdos

1 *CCNx Project*

O *CCNx Project* foi desenvolvido pelo PARC[1] e a sua implementação corresponde à definição de *Content-Centric Networking* [4]. A concretização do *CCNx Project* promove uma maneira nova de comunicar na *Internet*, nomeadamente através de conteúdo.

A arquitetura do *CCNx Project* tem definidos vários módulos, sendo os principais o *naming* do conteúdo, mensagens de rede (*Interest, Content Object e InterestReturn*), *Matching* desses conteúdos, verificação e encaminhamento [3].

O *CCNx Distillery* é a segunda versão do *CCNx Project* que liga estes módulos e monta-os de forma a criar o *software* funcional. A maior alteração desta versão para a sua precedente é que agora os módulos são independentes uns dos outros, desta forma tornou-se mais eficiente.

O *CCNx Project* tem implementado os protocolos necessários para os interessados nesta vertente de comunicação *Internet* poderem fazer experiências, pesquisas e desenvolver sobre a tecnologia.

O protocolo base utiliza dois tipos de mensagem:

[1] `https://www.parc.com`

- *Interest*

  Serve para pedir um *Content Object* à rede e é encaminhado através das rotas que conhece.

- *Content Object*

  Serve para enviar o conteúdo requerido por mensagens *Interest* provenientes da rede. Este conteúdo tem de ser autenticado devidamente pelo utilizador que dispõe o conteúdo.

Uma rede CCNx contém [2] *Content Producers*, *Content Publishers*, *Content Consumers*, *Caches* e *Forwarders*.

Como os próprios nomes indicam, um *Content Producer* produz conteúdo, um *Content Publisher* pega nesse conteúdo produzido e transforma-o em *Content Objects* com a autenticidade do *Content Producer* e um *Content Consumer* pede o conteúdo através do protocolo CCNx com uma mensagem *Interest*. O encaminhamento de mensagens é feito através do *Forwarder* enquanto que os *in-network caches* armazenam os *Content Objects* temporariamente (um tempo que é especificado pelo *Content Publisher* no momento que cria o *Content Object*).

2 Objetivos

Entender o estado da arte

O tráfego na *Internet* hoje em dia corresponde maioritariamente a transmissão de conteúdo entre *hosts*. Apesar da atual arquitetura já ter soluções para este tipo de comunicação não é uma arquitetura que está preparada para que o tráfego de conteúdo cresça como tem vindo a crescer ao longo dos anos e, neste aspeto, uma arquitetura *data-centric* baseada no ICN, como o CCNx, seria muito vantajosa [1].

O CCNx procura substituir a arquitetura da *Internet* - um modelo de comunicação *host-to-host* - por uma arquitetura baseada num modelo *data-centric*, tratando o conteúdo como entidade principal na arquitetura das redes. Uma rede com este tipo de arquitetura ganha inúmeras vantagens em relação ao modelo *host-to-host*, nomeadamente, na distribuição de conteúdo, segurança e desenvolvimento de aplicações [1].

Gostaríamos, com este trabalho laboratorial, de perceber se a mudança de arquitetura seria algo fácil e praticável ou se, por outro lado, esta tecnologia ainda necessita de mais tempo para se tornar numa opção credível à arquitetura atual.

Executar um exemplo simples

O objetivo fulcral deste trabalho laboratorial é ver como funciona uma rede baseada em ICN, como tal, gostariamos de pôr em prática primeiro um exemplo simples de forma a estudar como se comporta a rede e as suas interações. A realizar este objetivo também concretizámos o objetivo de criar uma rede ICN.

Aplicação a um caso real

Objective 3 text

3 Configuração do Sistema

A configuração do sistema dividiu-se em duas importantes partes. A primeira foi a instalação do CCNx, já a segunda prendeu-se com o estabelecimento da interação com o módulo que faz o encaminhamento na rede, o Metis. Para isso, guiámo-nos por instruções básicas para instalar o *CCNx Binary Release*[2].

3.1 *CCNx Project*

Primeiramente, foi necessário fazer um *update* do Ubuntu e instalar algumas bibliotecas:

```
sudo apt-get update -y
apt-get install uncrustify doxygen -y
apt-get install libssl-dev openssl libevent-dev -y
```

Com isto, o *hardware* e o Sistema Operativo ficaram preparados para instalar o *software* do CCNx, distribuído pelo seu domínio oficial (`CCNx.org`), na forma de um ficheiro zipado, com extensão .tgz. Para ter acesso ao ficheiro zipado, usou-se o `curl`:

[2] `http://blogs.parc.com/ccnx/howto-install-ccnx-binary-release/`

```
curl -O http://www.ccnx.org/releases/distillery-ccnx-absinthe-
1.0.20150516-Linux-x86_64.tgz
```

E para o extrair, o `tar`:

```
sudo tar -xzf distillery-ccnx-absinthe-1.0.20150516-Linux-x86_64.
tgz -C /
```

Para finalizar a instalação do CCNx, bastou correr o distillery-post-install, presente no diretório bin, do ccnx:

```
/usr/local/ccnx/bin/distillery-post-install
```

E, assim, o CCNx ficou pronto para ser utilizado. Para facilitar o tratamento de caminhos pelos ficheiros, entregou-se a algumas variáveis pequenos *paths*, utilizando o `export`:

```
export PARC_HOME=/usr/local/parc
export CCNX_HOME=/usr/local/ccnx
export PATH=$PATH:$PARC_HOME/bin:$CCNX_HOME/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$PARC_HOME/lib:$CCNX_HOME
/lib
```

No entanto, mais à frente, optou-se por uma versão otimizada do CCNx, com melhorias de *performance* mas com maiores dificuldades em fazer *debugging*. Para isso, fez-se *download* do seguinte *link*:

```
wget http://ccnx.org/releases/distillery-ccnx-absinthe-
1.0.20150516-Linux-x86_64.optimized.tgz
```

Sendo que posteriormente, foi preciso extraí-lo:

```
sudo tar -xzf distillery-ccnx-absinthe-1.0.20150516-Linux-x86_64.
optimized.tgz -C /
```

E novamente, para instalar, recorrer ao mesmo ficheiro:

```
/usr/local/ccnx/bin/distillery-post-install
```

Uma vez feito isto, foi tempo de correr o *daemon* responsável por fazer o *forwarding* do CCNx, com *root* iniciado, que iria depois ficar em *background*:

3.2 Metis

```
wget http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/metis-5.1.0.tar.gz
gunzip metis-5.1.0.tar.gz
tar -xvf metis-5.1.0.tar
cd metis-5.1.0
apt-get install cmake
apt-get update && apt-get install build-essential
make config
make
make install
```

4 Dados Experimentais

4.1 Exemplo simples

```
metis_daemon -capacity 0 &
```

O passo seguinte foi criar uma chave, com um limite de 2048 bits e uma validade de 1000 dias, para depois usar nos pedidos:

```
parc_publickey create .ccnx_keystore.p12 <password> <username>
2048 1000
```

Neste caso foi necessário substituir os campos `<password>` e `<username>` por algo específico, característico do utilizador. Para que um *Publisher* colocasse algo na rede, fez-se:

```
ccnx_server -identity .ccnx_keystore.p12 -password <password>
lci:/date /bin/date &
```

Sendo que o processo também ficou em *background*. Representativamente, o que está anexado ao `"lci:"`, simboliza o nome do ficheiro que fica na rede, e o que vem depois, a localização do ficheiro no diretório corrente. Para um *Consumer* pedir conteúdo na rede (neste caso o `/date`), fez-se:

```
ccnx_client -identity .ccnx_keystore.p12 -password <password>
lci:/date
```

Uma vez que os processos do ccnx_server e do metis ficaram em *background*, é gerada imediatamente uma mensagem sempre que um *Consumer* pede por conteúdo na rede. Para ver (e listar) a tabela de *routing* usada pelo *daemon*, fez-se:

```
ccnx_client -identity .ccnx_keystore.p12 -password <password>
lci:/date
```

4.2 Aplicação a um caso real

5 Resultados e Conclusões

Bibliography

[1] André Diegues and Fábio Teixeira. Information centric networking. 2016.

[2] Marc Mosko. Ccnx 1.0 protocol introduction. Technical report.

[3] Marc Mosko, Ignacio Solis, Ersin Uzun, and Christopher Wood. Ccnx 1.0 protocol architecture. Technical report, 2015.

[4] Fabian Oehlmann. Content-centric networking. *Network*, 43, 2013.