INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Redes de Distribuição de Conteúdos

Integração com IPTV e Estudo sobre Codificação de Rede

Rui Dinis Mangueira Bento

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente:	Prof. Luís Eduardo Teixeira Rodrigues
Orientador:	Prof. Rodrigo Seromenho Miragaia Rodrigues
Vogais:	Prof.ª Teresa Maria Sá Ferreira Vazão

Setembro, 2007

Redes de Distribuição de Conteúdos

Integração de IPTV e Estudo sobre Codificação de Rede

Rui Dinis Mangueira Bento

AGRADECIMENTOS

Gostaria de aproveitar esta para oportunidade para expressar a minha gratidão a todos aqueles que me ajudaram e tornaram possível a realização deste projecto.

Família e amigos, pelo seu apoio contínuo e encorajamento nas fases menos fáceis.

Ao meu orientador, Professor Rodrigo Miragaia Rodrigues, pela orientação e apoio incondicionais ao longo de todo o trabalho, pela vontade de levar este projecto a bom porto, por me ter oferecido a oportunidade de investigar um tema extremamente interessante que conferiu mais profundidade a este trabalho (e que permitiu toda a elaboração da segunda parte do mesmo). Foi um privilégio.

A todos os elementos da empresa Mobbit Systems com quem tive a oportunidade de trabalhar e que tornaram este projecto possível. Um agradecimento especial aos Engenheiros João Tacanho e João Oliveira, com quem tive uma colaboração mais estreita.

À empresa Cisco Systems, em especial ao Engenheiro Jorge Portugal com quem trabalhei de forma directa, por ter viabilizado a primeira fase deste projecto e pelo suporte que sempre me deu ao longo do mesmo.

Chris Harrison, membro do Human Computer Interaction Institute da Carnegie Mellon University, por ter gentilmente cedido a imagem utilizada na capa desta Dissertação. Esta consiste numa representação visual de como as cidades espalhadas por todo o mundo estão interligadas numa rede que constitui a Internet.

Aos Professores Luís Rodrigues e Teresa Vazão, por terem participado na avaliação desta dissertação e terem permitido e sugerido que fosse levado a cabo um processo de revisão da mesma de forma a elevar a qualidade do documento final.

Por fim, gostaria de agradecer à instituição Instituto Superior Técnico por cinco proveitosos anos de aprendizagens e vivências.

RESUMO (EM PORTUGUÊS)

A Internet conheceu um forte crescimento nos últimos anos. Todavia, a banda disponível na rede permanece sobreocupada pela sua forte utilização global e pelo crescimento do volume de conteúdos com elevado consumo de recursos. As Redes de Distribuição de Conteúdos (CDNs – *Content Distribution Networks*) constituem uma abordagem eficiente para melhorar a qualidade de serviço na Internet ao replicar conteúdos desde o local de origem para servidores tão próximos quanto possível dos utilizadores.

Esta Dissertação tem como tema central o estudo das CDN. É composta por três capítulos fundamentais: enquadramento do tema estudado no âmbito desta dissertação e trabalho relacionado; estudo relacionado com a aplicação prática de uma CDN centralizada a um sistema em produção; estudo de Codificação de Rede como uma estratégia utilizada para aumentar a eficiência da disseminação de conteúdos em CDNs *peer-to-peer*.

No primeiro dos capítulos apresentados, procura-se definir uma arquitectura base sobre a qual as CDNs operam, independentemente dos seus detalhes de implementação. Endereça-se um conjunto de desafios importantes, bem como estratégias usadas mais frequentemente para os ultrapassar. Por fim, uma vez que a arquitectura e os desafios apresentados se prendem com uma filosofia centralizada do tipo cliente/servidor, particulariza-se o estudo para CDNs *peer-to-peer*.

A aplicação prática de uma CDN centralizada a um sistema em produção consiste na integração da uma CDN proprietária, da Cisco Systems, com um sistema comercial de *Corporate TV*, da Mobbit Systems. Este último oferece um serviço de televisão por IP com gestão centralizada. Sem que exista um processo adequado de distribuição de conteúdos, pode pôr-se o caso de os mesmos conteúdos serem transmitidos várias vezes através da mesma ligação, num uso ineficiente dos recursos de largura de banda.

Pela experiência anterior, constatou-se a importância de uma topologia estática e pré-definida da simplificação do desenho e eficiência da CDN. Por conseguinte, alguns problemas relativos a sistemas dinâmicos foram estudados na segunda parte desta dissertação, nomeadamente quais os benefícios que podem advir da aplicação de métodos de Codificação de Rede em CDNs *peer-to-peer*, em particular o BitTorrent.

Palavras-Chave: Redes de Distribuição de Conteúdos (CDNs), Arquitectura de uma CDN, Televisão por IP (IPTV), *Corporate TV*, Codificação de Rede, *Peer-to-Peer*, BitTorrent.

ABSTRACT (IN ENGLISH)

The Internet has experienced a strong growth over the last years. However, the available bandwidth remains over occupied due to its intense global usage and the increasing volume of contents with high resource consumption. The Content Distribution Networks (CDNs) are an efficient approach to improve the quality of service in the Internet by replicating contents from their original locales to servers as near from the users as possible.

This dissertation has CDNs as its central subject of discussion. It comprises three core chapters: description of how a CDN operates and which are the main challenges regarding the design of an efficient CDN; analysis of a practical application of a centralized CDN to a commercial system; study of Network Coding as a strategy to improve the dissemination of contents over peer-to-peer CDNs such as BitTorrent.

In the first of the listed chapters, an approach to a common architecture underlying the operation of CDNs regardless of their implementation details is provided. A number of relevant challenges are addressed, as well as the most frequently used strategies to overcome them. Finally, as the provided architecture and challenges pertain to a centralized client/server conceptualization, some light will be shed over peer-to-peer CDNs.

The practical application of a centralized CDN to a commercial system consists on the application of a Cisco Systems CDN to a centralized corporate TV system, developed by Mobbit Systems. Without an adequate content distribution strategy it might occur that the same content has to be transferred through the same link more than once, inefficiently using the available bandwidth resources.

After deploying this system, it was realized that the design of the CDN was substantially simplified by the existence of a relatively static and predefined distribution tree. Hence, in the second part of the thesis, some of the problems that arise in more dynamic deployments were studied, in particular, the impact of Network Coding as a form of increasing the efficiency content distribution.

Keywords: Content Distribution Networks (CDNs), CDN Architecture, Television over IP (IPTV), Corporate TV, Network Coding, Peer-to-Peer, BitTorrent.

ÍNDICE

ANEXOS

1. Estrutura dos ficheiros XML de configuração do sistema Mobbit InSight 2.0

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÓNIMOS E ABREVIAÇÕES

- ACNS: *Application and Content Networking System*. Rede de distribuição de conteúdos desenvolvida e distribuída pela empresa Cisco Systems.
- CIFS: *a Common Internet File System*. Sistema de ficheiros distribuído apresentado pela Microsoft em 1996 como uma versão mais avançada do SMB[Cod01].
- CDM: *Content Distribution Manager*. Componente da rede Cisco ACNS a partir do qual é realizada toda a gestão da mesma.
- CDN: *Content Distribution Network*. Acrónimo para o termo inglês para Rede de Distribuição de Conteúdos.
- DNS: *Domain Name System*. Sistema que armazena e associa várias informações a nomes de domínios, traduzindo-os para endereços IP.
- DTV: *Digital Television*. Sistema de telecomunicações utilizado para transmitir e receber imagem e som através de sinais digitais, ao invés de fazer uso de sinais analógicos como nos sistemas tradicionais de transmissão de televisão.
- DVR: *Digital Video Recorder*. Aparelho que permite a gravação de vídeo em formato digital num disco ou outro meio de armazenamento.
- FQDN: *Fully Qualified Domain Name*. Nome de domínio não ambíguo que especifica a posição de um nó de forma absoluta na árvore hierárquica de DNS.
- FTP: *File Transfer Protocol*. Protocolo para transferência de ficheiros, entre dois computadores, na Internet.
- HDTV: *High-definition Television*. Sistema de emissão digital de televisão que oferece uma resolução significativamente melhor quando comparada com os formatos tradicionais (NTSC, SECAM ou PAL). Embora tenha surgido como um formato analógico na Europa e Japão, é habitualmente emitida através da Internet uma vez que a televisão digital (DTV) requer muito menos largura de banda se for utilizado um nível suficiente de compressão de vídeo.
- HTML: *Hypertext Markup Language*. Linguagem predominante para a criação de páginas na *World Wide Web*.
- HTTP: *Hypertext Transfer Protocol*. Protocolo para veiculação de informação na *World Wide Web*. O seu propósito original era a definição de uma forma para publicar e receber páginas HTML.
- HTTPS: *Hypertext Transfer Protocol Secure*. Protocolo para veiculação de informação na *World Wide Web* construído sobre uma sessão SSL (em que informação é trocada de forma cifrada, oferecendo um nível extra de segurança).
- ICC: *Instant Channel Change*. Funcionalidade de um serviço de televisão digital que consiste na troca rápida de canais de televisão. Esta funcionalidade surge para melhorar a experiência

do utilizador já que nos tradicionais sistemas de IPTV, existia um atraso temporal considerável inerente à troca de canal.

- IETF: *Internet Engineering Task Force*. Organização que desenvolve e promove padrões para a Internet.
- IP: *Internet Protocol*. Protocolo orientado a dados utilizado para comunicação numa rede de troca de pacotes. É utilizado na camada de rede da Internet.
- IPTV: *Internet Protocol Television*. Serviço de televisão digital distribuído com recurso ao *Internet Protocol*.
- IRC: *Internet Relay Chat*. Sistema de *chat* em tempo real baseado em texto. Foi desenhado sobretudo para comunicações em grupo (um-para-muitos), em fóruns denominados canais, permitindo também comunicação individual (um-para-um).
- ISP: *Internet Service Provider*. Organização que oferece aos consumidores acesso à Internet e outros serviços relacionados.
- LAN: *Local Area Network*. Rede de computadores que cobre uma área geográfica pequena.
- NTSC: *National Television System Comittee*. Sistema analógico de televisão utilizado em países como os Estados Unidos da América, Canadá, Japão, Coreia do Sul, Filipinas, México, entre alguns outros. Foi estabelecido em 1940 nos Estados Unidos da América pela *Federal Communications Commission* como uma forma de resolver os conflitos existentes entre várias companhias na transmissão de sinais televisivos pelo país.
- OS X: linha de sistemas operativos desenvolvidos, promovidos e vendidos pela empresa *Apple, Inc.*
- PAC: *Proxy Auto-Config*. Ficheiro que define como *web browsers* podem escolher, de forma automática, o *proxy server* a utilizar para um determinado URL.
- PAL: *Phase Alternating Line*. Sistema de codificação utilizado para a transmissão de televisão a cores criado na Alemanha em 1963. Desenvolvido por Walter Bruch da Telefunken, este *standard* surgiu como uma alternativa ao NTSC que, por motivos técnicos relacionados com as redes eléctricas da Europa, não era aplicável nesses países.
- PAP: *Picture and Picture*. Funcionalidade comum a alguns receptores de televisão em que dois ou mais canais televisivos dividem o ecrã, sendo apresentados simultaneamente. Alguns provedores de IPTV por *set-top box* oferecem este serviço.
- PIP: *Picture in Picture*. Funcionalidade semelhante ao PAP em que um canal de televisão é apresentado ocupando o ecrã completo, enquanto que um ou mais canais são apresentados em pequenas janelas que se sobrepõem. Alguns provedores de IPTV por *set-top box* oferecem este serviço.
- RTSP: *Real Time Streaming Protocol*. Protocolo desenvolvido pela IETF para a utilização em sistemas de *streaming* de conteúdos multimédia, que permite que um cliente controle remotamente o servidor de *streaming*.

- SECAM: *Séquentiel Coleur à Mémoire*. Protocolo analógico de televisão a cores criado na França no ano de 1961. Criado por Henri de France, da *Compagnie Française de Télévision*, este foi o primeiro *standard* para emissão de televisão a cores na Europa. É utilizado na França, Bélgica, Rússia, Países de Leste, Médio Oriente, entre outros locais.
- SMB: *Server Message Block*. Protocolo do nível da aplicação que permite a partilha de dados e recursos no contexto de uma rede de computadores, entre vários intervenientes [Cod01].
- SSH-2: *Secure Shell 2.* Protocolo que permite o estabelecimento de um canal seguro entre um computador local e um computador remoto. Usa criptografia simétrica para autenticar o computador remoto e permite que o computador remoto autentique o computador local.
- SSL: *Secure Sockets Layer*. Protocolo criptográfico que oferece comunicação segura na Internet para uma miríade de aplicações como, por exemplo, *web browsing*.
- TCP: *Transmission Control Protocol*. Protocolo nuclear da Internet que pertence ao seu nível de transporte. Permite que nós de rede criem conexões entre si.
- TFT: *Tit-for-Tat*. Estratégia, oriunda da teoria de jogos e presente no sistema *BitTorrent*, que visa a justiça numa rede para a distribuição de um conteúdo, obrigado a que cada nó participe activamente ao servir esse conteúdo para que o possa obter.
- URI: *Uniform Resourse Identifier*. *String* de caracteres definida para a identificação de um nome ou recurso.
- URL: *Uniform Resource Locator*. Utilizado como sinónimo de URI embora, de uma forma rigorosa, não tenha exactamente o mesmo significado.
- VOD: *Video on Demand*. Sistema que permite os utilizadores seleccionarem e verem conteúdos vídeo e multimédia sobre uma rede como parte de um serviço de televisão interactivo. Estes sistemas enviam o conteúdo em tempo real (*streaming*) enquanto o apresentam ao utilizador ou fazem *download* do mesmo para uma posterior visualização.
- VoIP: *Voice over Internet Protocol*. Encaminhamento de conversações de voz através da Internet ou de outra qualquer rede IP.
- WAN: *Wide Area Network*. Rede de computadores que cobre uma área geográfica de grandes dimensões.
- WCCP: *Web Cache Communication Protocol*. Protocolo proprietário da Cisco Systems para encaminhamento transparente de pedidos HTTP.
- WWW: *World Wide Web*. Sistema de documentos em *hipertexto* interligados suportado pela Internet. Estes conteúdos podem ser consultados com recurso a *Web Browsers*.
- XML: *Extensive Markup Language*. Linguagem do tipo *markup* que suposta um conjunto vasto de aplicações.

Página intencionalmente deixada em branco.

1 INTRODUÇÃO

> É como se instalássemos um sistema de auto-estradas nacional em que inicialmente era dada a autorização para conduzir a 80, depois a 90, posteriormente a 100, mais tarde a 120 e, por fim, talvez a 240 km/h. As auto-estradas mantinham-se as mesmas, bem como o risco – nulo de acidentes. A diferença é que esta auto-estrada não era apenas nacional. Cobria o mundo inteiro.

Thomas L. Friedman, *The World is Flat: A Brief of the Twenty-first Century*, 2006
Sobre a evolução das ligações digitais de elevada largura de banda por todo o Mundo.

1.1 Redes de Distribuição de Conteúdos

No espaço de cerca de duas décadas, a Internet estabeleceu-se como o principal veículo global de partilha de informação. Hoje, conta com mais de 1100 milhões de utilizadores, o que constitui aproximadamente 18% da população mundial[Int], e regista um crescimento anual na ordem dos 100%[CO01]. Embora o tempo para obtenção de conteúdos a partir da Internet tenha melhorado ao longo das últimos anos, a par do seu ininterrupto crescimento em número de utilizadores, a latência do acesso a conteúdos na *World Wide Web* permanece relativamente, e de um modo geral, elevada, afectando a qualidade de serviço percebida pelos seus utilizadores[Pen03].

A reduzida qualidade do serviço da Internet, tipicamente representada pelos longos atrasos na entrega de conteúdos, deve-se essencialmente a dois factores. Em primeiro lugar, a inexistência de uma coordenação central: embora o facto de que a Internet opera em larga escala sem uma gestão centralizada tenha possibilitado o seu rápido desenvolvimento e crescimento, torna-se difícil assegurar um serviço estável e uma *performance* adequada (note-se que, dada a dimensão da Internet, assegurar efectivamente uma coordenação central seria impraticável). Em segundo lugar, o fluxo de dados e a riqueza dos conteúdos tem conhecido um forte aumento, o qual continua a prevalecer sobre os aumentos na largura de banda e capacidade de servidores que eventualmente se verificam. Ambos os factores contribuem não só para um atraso no acesso a conteúdos disponibilizados pela Internet, mas tornam também a latência inerente a este acesso imprevisível[Pen03].

Por outro lado, servir conteúdos na Internet a partir de um único local determina a existência de um ponto único de falha, não sendo uma solução escalável e dando origem a problemas de fiabilidade e *performance*. À medida que os sítios *Web* se tornam mais populares, tornam-se também mais vulneráveis a fenómenos como o efeito *flash crowd*[SMB02]. Este fenómeno prende-se com picos no volume de pedidos feitos a um determinado sítio, excedendo a sua capacidade a um determinado nível (como a capacidade do servidor *Web* para lidar com pedidos concorrentes, o equipamento de rede utilizado ou a largura de banda disponível), o que pode levar a tempos de resposta extraordinariamente elevados ou mesmo ao *crash* do sítio. Um caso particular e bastante usual deste efeito denomina-se *Slashdot effect*: o nome foi atribuído com base no caso real de um sítio

extremamente popular que publica regularmente ligações a sítios com menos recursos, conduzindo a estes níveis de tráfego com os quais não conseguem lidar.

A abordagem sugerida pelas Redes de Distribuição de Conteúdos (CDN) prende-se essencialmente com a colocação dos conteúdos junto dos utilizadores finais. Ao servir conteúdos a partir de réplicas locais, tira-se partido dos princípios da localidade espacial e temporal, o que resulta numa melhoria significativa da latência percebida pelo utilizador, aumentando também a taxa de transferência de dados. Por outro lado, a carga exercida sobre os servidores originais decresce, pelo que estes vêem os requisitos a nível de infra-estruturas reduzidos. De um modo geral, o tráfego global da rede é também reduzido[SGD+02].

Por conseguinte, é possível definir uma Rede de Distribuição de Conteúdos como uma rede optimizada para distribuir conteúdos específicos, como páginas *Web* estáticas, sítios *Web* baseados em transacções, conteúdos estáticos diversos (desde *software* a conteúdos multimédia), *streaming* de conteúdos multimédia ou mesmo emissão em tempo real de áudio e vídeo. O seu propósito é o de fornecer aos utilizadores os conteúdos mais actualizados, de forma rápida e com elevada disponibilidade, recorrendo para isso à replicação dos mesmos para pontos tão próximos do cliente final quanto possível.

1.2 Objectivos da Dissertação

A Dissertação apresentada neste relatório ocupa-se do estudo das CDN como uma estratégia para a disseminação de conteúdos de grandes dimensões através da Internet. É composto por três capítulos principais: a descrição do modo de operação de uma CDN e dos principais desafios inerentes ao desenho de uma CDN eficiente, bem como uma apresentação de trabalho relacionado; estudo relacionado com a aplicação prática de uma CDN centralizada a um sistema em produção; estudo de Codificação de Rede como uma estratégia utilizada para aumentar a eficiência da disseminação de conteúdos em CDNs *peer-to-peer*. Nas subsecções 1.2.1, 1.2.2 e 1.2.3 são descritos, respectivamente, cada um destes objectivos.

1.2.1 Operação de uma CDN

Embora as Redes de Distribuição de Conteúdos apresentem particularidades distintas de acordo com as suas diferentes implementações, é possível identificar um conjunto de princípios comuns às mesmas, o que permite definir uma arquitectura subjacente ao modo como operam. Por outro lado, existe também um conjunto de desafios importantes com os quais as diferentes CDNs necessitam de como, por exemplo, onde colocar as réplicas dos servidores, quais os conteúdos a distribuir para as réplicas, como encaminhar os pedidos dos utilizadores para os melhores conteúdos garantindo uma latência baixa no processo ou como lidar com conteúdos desactualizados ou incompletos.

Neste capítulo pretende-se definir uma arquitectura de base ao funcionamento das diferentes CDNs, independentemente dos seus detalhes de implementação. Pretende-se também apresentar um conjunto de importantes e frequentes desafios colocados a estes sistemas, bem

como diferentes abordagens adoptadas para os ultrapassar. Por fim, é endereçado um tipo particular destes sistemas, nomeadamente CDNs *peer-to-peer*.

1.2.2 Estudo acerca da implementação prática de uma CDN centralizada

O sistema InSight, da empresa Mobbit Systems, tem como propósito oferecer um serviço de *Corporate TV*, ao qual estão associados diferentes canais, a um conjunto de utilizadores. Um sistema de *Corporate TV* consiste num sistema de televisão digital disponibilizado a utilizadores da Internet através de ligações de banda larga. Por conseguinte, é possível definir o sistema InSight como um sistema cujo propósito é a gestão de um conjunto de canais multimédia, bem como a distribuição dos seus conteúdos por todos os terminais associados aos mesmos. Estes últimos podem apresentar a dispersão geográfica, uma vez que a disseminação de conteúdos é assegurada através da Internet. Dadas as restrições impostas à Internet relativamente à limitação de velocidade e largura de banda na transmissão de informação, torna-se necessária a utilização eficiente dos recursos disponíveis[Mobb].

De forma a melhorar a eficiência da transmissão de dados no contexto do sistema InSight, foi sugerida a integração com um sistema, da Cisco Systems, cujo propósito é a optimização da disseminação de conteúdos pela Internet. Este sistema denomina-se Cisco ACNS, constituindo uma rede *overlay* para a distribuição de conteúdos a partir de um determinado servidor-fonte (o servidor central do sistema InSight) para um número arbitrário de clientes (os terminais do sistema InSight).

O projecto descrito neste relatório tem então como objectivo a integração do sistema InSight, da Mobbit Systems, com o sistema Cisco ACNS, da Cisco Systems. Tal integração permitirá o uso eficiente dos recursos disponíveis ao sistema InSight, recorrendo a um sistema proprietário que oferece um conjunto importante de garantias de qualidade. Pretende-se também que a integração imponha o mínimo de alterações ao sistema InSight: quanto menor for o volume de funcionalidades, menor o custo de adaptação a novos processos; por outro lado, modificações no contexto das redes de clientes para a actualização do sistema podem ser difíceis ou mesmo inviáveis.

O plano do projecto consiste em três fases distintas. A primeira fase prende-se com uma análise dos sistemas InSight e Cisco ACNS para que sejam identificados os pontos a partir dos quais é possível fazer uma integração. A segunda fase, por seu turno, consiste na elaboração dos módulos de software necessários para que seja concretizada a introdução dos sistemas. A terceira fase consiste na aplicação do sistema em casos de teste (sistemas piloto) e em casos reais (junto de clientes).

1.2.3 Estudo da aplicação de Codificação de Rede a CDNs *Peer-to-Peer*

Uma das observações ditadas pela experiência da implementação anterior foi a importância de uma topologia estática e pré-definida na simplificação do desenho e eficiência da CDN. Tal motivou um estudo subsequente acerca de quais as consequências ao nível das escolhas de desenho da CDN em contextos dinâmicos. A atenção foi então focada em sistemas *peer-to-peer*.

É possível afirmar que a Internet conheceu durante as últimas duas décadas, período em que se tornou parte integrante e fundamental da vida quotidiana à semelhança das infra-estruturas telefónicas ou de estradas, duas grandes revoluções[ERM05]: o aparecimento da *World Wide Web* e a emergência das redes *peer-to-peer* de partilha de conteúdos.

A criação da *World Wide Web*, por Tim Berners-Lee do CERN em 1989 (apesar de o primeiro sítio ser datado de 6 de Agosto de 1991[Fri06], *http://info.cern.ch*), trouxe uma forma inovadora e sem precedentes de aceder a informação em qualquer parte do globo, a partir de qualquer parte do globo. Durante os primeiros anos após o arranque da *World Wide Web*, o número de utilizadores da Internet cresceu de 600 mil para 40 milhões[Fri06].

Em Junho de 1999, quase dez anos depois do advento da WWW, Shawn Fanning, na época um jovem de 18 anos de idade, apresentou ao mundo a sua criação: o Napster[HIM04]. Este sistema inovador visava permitir a troca de música, em formato MP3, entre os seus utilizadores. Em pouco mais de um ano, o número de utilizadores deste sistema cresceu de 30 (um grupo restrito de amigos de Fanning que se prestaram a testar o Napster) para 60 milhões, ultrapassando a America Online (27 milhões) e o Yahoo! (54 milhões)[Men01]. A questão mais proeminente que se levantou com o surgimento desta nova tecnologia centrou-se na transgressão de direitos de autor, uma vez que o seu modelo de negócio consistia na distribuição e partilha de obras protegidas por estes direitos. Por conseguinte, o Napster foi rapidamente processado e encerrado em 2002. Todavia, a sua breve existência trouxe à Internet uma nova forma de troca de informação: o *peer-to-peer*. A partir deste momento, cada utilizador poderia trocar, em larga escala, informação com outros utilizadores, desafiando o tradicional modelo cliente-servidor (apesar de, no entanto, o Napster consistir uma rede *peer-to-peer* centralizada, o que significa que necessitava sempre de um servidor para efectuar determinadas tarefas, como a localização de um ficheiro pretendido)[DNB03].

O *Napster* foi desactivado, mas deixou marcas profundas que para sempre mudariam a forma de trocar informação na Internet. Com efeito, pouco depois surgiram inúmeras outras aplicações com o mesmo propósito, apesar de introduzirem várias variantes às suas arquitecturas e modos de funcionamento. Hoje em dia, atribui-se cerca de 80% do tráfego total numa ligação *backbone* IP de alta velocidade a tráfego *peer-to-peer*[ERM05].

É possível afirmar que o Napster foi a primeira CDN *Peer-to-Peer* a ser utilizada em larga escala. Tal como sucedia com as CDNs tradicionais, o seu objectivo era o de replicar os conteúdos tanto quanto possível, aumentando a sua disponibilidade e diminuindo o tempo de *download* e a latência no acesso aos mesmos por parte dos utilizadores finais. Todavia, este sistema não se baseava numa arquitectura cliente-servidor (em que existem dois tipos claros de entidades – os que servem e os que obtêm conteúdos), mas numa arquitectura *peer-to-peer* (em que cada utilizador simultaneamente serve e obtêm conteúdos).

Embora numerosos sistemas para partilha de ficheiros em redes *peer-to-peer* tenham vindo a ser propostos e implementados ao longo dos últimos anos, apenas um reduzido número subsistiu aos testes de utilização intensiva e é actualmente utilizado de forma global. O sistema BitTorrent, responsável por mais de 50% de todo o tráfego *peer-to-peer* na Internet, é um desses sistemas[DL07]. Disponibilizado livremente na Internet a 2 de Julho de 2001, pela mão de um jovem programador Norte-Americano de nome Bram Cohen, o BitTorrent surgiu como uma alternativa

mais eficiente perante outros sistemas como o eDonkey2K, FastTrack, Gnutella ou Overnet ao focar-se na replicação do conteúdo ao invés da localização do mesmo[LUM06]. Enquanto que a generalidade das CDNs *peer-to-peer* empreendia um maior esforço na localização de um determinado conteúdo junto de uma comunidade de nós, o BitTorrent apresentou uma abordagem inovadora ao agrupar utilizadores que partilham um dado conteúdo numa rede exclusiva. Por conseguinte, o problema da localização de conteúdos era assim ultrapassado, dando mais espaço à determinação de estratégias para a distribuição eficiente dos mesmos. Segundo o BitTorrent, um ficheiro é subdividido em pequenos pedaços, permitindo que cada um destes possa ser obtido em paralelo, junto de diferentes nós. Para além disto, existem ainda duas estratégias na base do sistema: a procura constante pelo bloco mais raro (*Rarest First Algorithm*), de forma a uniformizar a distribuição de pedaços junto dos utilizadores, e a disponibilização de pedaços a outros utilizadores que também os disponibilizem (*Choke Algorithm*), seguindo uma estratégia do tipo *tit-for-tat*, de forma a incentivar a participação de todos, maximizando os recursos de largura de banda disponíveis[Coh03].

Recentemente, soluções com recurso a Codificação de Rede têm vindo a ser propostas para aumentar a eficiência de sistemas actuais para partilha de ficheiros, como o BitTorrent[GR05]. O *Network Coding* consiste num mecanismo que visa maximizar a quantidade de dados que é possível fazer fluir numa rede ao permitir que cada nó da rede possa gerar e transmitir blocos codificados de informação. Idealmente, de acordo com este sistema é possível que um determinado nó possa reconstruir o ficheiro original caso tenha tantos blocos aleatórios codificados quanto os blocos originais existentes[FBW06]. Consequentemente, as preocupações relativas à localização e determinação de que blocos devem ser obtidos a cada instante deixam de existir, eliminando-se o problema da determinação de um algoritmo eficiente para a definição de qual o bloco a obter a cada momento para potenciar a prestação do sistema. Contudo, não existe ainda uma aplicação prática desta abordagem, pelo que a discussão acerca dos seus eventuais benefícios permanece num plano hipotético e de investigação. Embora este método ofereça uma significativa vantagem ao eliminar o problema da selecção do próximo bloco a obter, acarreta consigo três diferentes tipos de *overheads*: cada bloco passa agora a ser trocado com informação extra relativa à sua codificação, o procedimento para reconstrução do conteúdo original com base em blocos originados por métodos de Codificação de Rede é computacionalmente *pesado* e a complexidade do código necessário para a implementação aumenta, tornando-se mais propenso a *bugs* informáticos[LUM06].

O objectivo desta segunda fase do Trabalho Final de Curso é então o de determinar a utilidade de *Network Coding* numa CDN *peer-to-peer* (mais concretamente, o BitTorrent) recorrendo, para o efeito, a simulações de ambas as abordagens em diferentes contextos e situações.

1.3 Estrutura do Relatório

Esta dissertação é composta por seis capítulos e um anexo. No capítulo presente é apresentada motivação subjacente a esta dissertação sob a forma de uma introdução ao conceito e ao contexto

em que surgem as Redes de Distribuição de Conteúdos, quais os objectivos dos três principais capítulos deste trabalho e quais os conceitos essenciais à leitura e compreensão deste relatório.

Na secção dois é apresentado um estudo sobre as CDNs, nomeadamente através da definição de uma arquitectura base comum a estes sistemas, da identificação dos principais desafios inerentes a estes sistemas e abordagens mais frequentemente adoptadas para resolução dos mesmos e, finalmente, um estudo acerca da particularização destes sistemas de acordo com um paradigma *peer-to-peer*. É também feito um levantamento de trabalho relacionado a ser presentemente levado a cabo.

Na secção três é apresentado o estudo e descrição da concretização da integração de uma CDN centralizada a um sistema de *Corporate TV* em produção.

Na quarta secção é levado a cabo um estudo do impacto de métodos de codificação de rede em redes de distribuição de conteúdos *peer-to-peer*, mais concretamente no BitTorrent.

Na secção cinco é apresentada uma breve análise acerca de trabalho relacionado com as áreas abrangidas nesta dissertação.

Na sexta secção é apresentado um conjunto de breves conclusões relativas ao trabalho efectuado. É composta por duas subsecções nas quais são abordados, respectivamente, o impacto da utilização de CDNs de um modo geral e o impacto da utilização de métodos de Codificação de Rede nestes sistemas.

O anexo consiste na estrutura dos ficheiros XML utilizados para disseminação de conteúdos no sistema Mobbit InSight. O segundo consiste num artigo escrito no âmbito da disciplina de Introdução à Investigação dedicado a redes de distribuição de conteúdo e que serve de suporte a esta Dissertação.

2 ENQUADRAMENTO E TRABALHO RELACIONADO

Nesta secção serão abordados os fundamentos relativos à operação de uma Rede de Distribuição de Conteúdos, bem como trabalho relacionado com o tema. Em particular, na subsecção 2.1 é apresentada uma visão geral relativa à operação destes sistemas, a qual é focada na infra-estrutura destes sistemas, na definição de uma arquitectura base comum aos mesmos e independente de detalhes de implementação e nos aspectos referentes à colocação de réplicas, consistência de dados e encaminhamento de pedidos. Na subsecção 2.2 é endereçado o problema da distribuição de servidores, sendo apresentadas abordagens teóricas e heurísticas para a resolução do mesmo. Na subsecção 2.3 discutem-se as duas vertentes do problema da distribuição de réplicas: a sua colocação e criação. Na subsecção 2.4 é focado o problema da consistência das réplicas, sendo abordados diferentes modelos de consistência e mecanismos de distribuição de conteúdos. Na secção 2.5 são focados os dois sub-problemas associados ao pedido de serviços – a localização de servidores e o encaminhamento de pedidos – bem como o método de *anycasting*. Na secção 2.6 são analisadas as CDNs *peer-to-peer*, sendo a abordagem realizada de acordo com o seu modelo de rede: estruturadas ou não estruturadas. Por fim, na secção 2.7, são analisados diferentes sistemas *multicast* para a disseminação eficiente de conteúdos de grandes dimensões ou *streaming* de conteúdos multimédia para uma população arbitrária de clientes.

2.1 Visão Geral

Embora os detalhes inerentes ao funcionamento das CDN variem de entre as várias implementações existentes das mesmas (cujos propósitos são também frequentemente distintos), é possível definir uma arquitectura base sobre as quais estas operam, bem como um conjunto de desafios a que têm de responder, tipicamente derivados do princípio que lhes é subjacente: a replicação de conteúdos.

Nesta secção vamos então procurar definir duas abordagens distintas para a definição de uma CDN (modelos *overlay* e de rede), uma arquitectura base (os seus componentes e como estes interagem entre si) e um conjunto de desafios importantes, nomeadamente:

a. *Onde* colocar as réplicas?
b. *Quando* replicar conteúdos?
c. Como garantir a *consistência* de todas as réplicas?
d. Como *encaminhar* os pedidos dos clientes?

2.1.1 Infra-Estrutura de uma Rede de Distribuição de Conteúdos

É possível identificar duas abordagens gerais para a construção da infra-estrutura de uma CDN: o modelo *overlay* e o modelo de rede.

De acordo com o modelo *overlay*, a CDN é construída sobre uma outra rede como, por exemplo, a Internet. É então possível considerar que os nós de uma CDN, de acordo com este modelo, se encontram ligados entre si por caminhos virtuais, os quais poderão consistir em

numerosas ligações físicas. Segundo esta abordagem, vários servidores específicos do nível da aplicação são colocados em vários pontos da rede de forma a gerirem a distribuição de conteúdos específicos. A infra-estrutura da rede sobre a qual é construída a CDN não tem qualquer papel no processo de distribuição de conteúdos, fornecendo apenas a conectividade básica entre nós da CDN e, eventualmente, assegurando aspectos da qualidade do serviço (QoS) para tipos particulares de tráfego. Bons exemplos desta abordagem são implementações por organizações como a Akamai, Digital Island ou Speedera[Wet02], em que o conteúdo é replicado por milhares de servidores espalhados por todo o Mundo. A abordagem *overlay* simplifica a gestão e cria oportunidades para novos serviços, uma vez que as entidades responsáveis pelo fornecimento de conteúdos não controlam a infra-estrutura da rede sobre a qual assenta a CDN, ficando isentas de responsabilidade a esse nível. O processo de oferecer novos serviços é simplificado ao ponto de consistir simplesmente na distribuição de código pelos servidores da CDN.

O modelo de rede consiste na colocação de código em *routers* e *switches*, o que permite que estes reconheçam tipos aplicacionais específicos, tomando decisões de encaminhamento de pedidos e de tráfego de acordo com políticas pré-especificadas. Exemplos desta abordagem consistem em nós que encaminham pedidos para *caches* locais ou fazem a multiplexagem de pedidos chegados a data centres para servidores específicos optimizados para servir determinados tipos de conteúdos.

Em determinados casos, ambas as abordagens são usadas em conjunto como, por exemplo, no caso da rede Akamai, em que existem switches no front-end de uma *farm* de servidores com o propósito de redireccionarem pedidos http provenientes de utilizadores finais para servidores Akamai que lhes estejam mais próximos[DMP+02]. Todavia, as CDN recorrem predominantemente à abordagem *overlay*, podendo inclusivamente serem consideradas, de uma forma geral, como redes elaboradas com o propósito de levarem conteúdos para junto de um público distribuído, construídas como uma camada sobre uma infra-estrutura de rede já existente. A infra-estrutura de rede contém componentes de transporte de dados, tais como *routing IP* (que compreende um conjunto de protocolos para determinar o caminho seguido por um pacote de informação deste a sua origem ao seu destino), QoS (mecanismos de controlo que garantem um determinado nível de performance relativo ao fluxo de dados), *multicasting* (entrega de informação a um conjunto de destinos de forma simultânea numa determinada rede, criando cópias dessa informação apenas quando os caminhos para os destinos se separam), entre outros.

2.1.2 Arquitectura Geral de uma Rede de Distribuição de Conteúdos

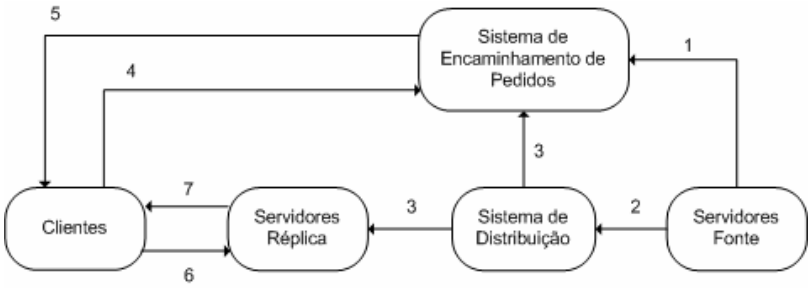


fig. 1: arquitectura geral de uma Rede de Distribuição de Conteúdos

A figura 1 apresenta um diagrama com a arquitectura geral de uma CDN (nomeadamente o mapeamento de componentes e conectores em elementos de *hardware*). Esta consiste em cinco componentes distintos: os *servidores de origem*, o *sistema de distribuição de conteúdos*, *o sistema de encaminhamento de pedidos*, os *servidores réplica* e o *utilizador final*.

As relações entre os componentes, representadas na figura por linhas direccionais indexadas, são as que se seguem:

1. Os *servidores de origem* delegam o espaço de nomes dos identificadores dos recursos (URIs) a serem distribuídos pela CDN ao *sistema de encaminhamento de pedidos*.
2. Os *servidores de origem* publicam os conteúdos a serem distribuídos pela CDN no *sistema de distribuição*.
3. O *sistema de distribuição* move os conteúdos publicados para os *servidores réplica*.
4. O *utilizador final* solicita os conteúdos que pretende ao que este julga ser o *servidor de origem*. Todavia, devido à delegação do espaço de nomes dos URIs dos conteúdos, o pedido é direccionado para o *sistema de encaminhamento de pedidos*.
5. O *sistema de encaminhamento de pedidos* encaminha o *utilizador final* para o *servidor réplica* da CDN mais adequado para servir o pedido.
6. O *utilizador final* solicita os conteúdos que pretende ao *servidor réplica* para o qual foi encaminhado.
7. O *servidor réplica* responde ao *utilizador final* com o conteúdo que este requisitou.

2.1.3 Colocação de Réplicas

A colocação de réplicas lida essencialmente com o número de réplicas que um determinado objecto tem e onde estas devem ser colocadas no contexto da rede. Estas questões dizem respeito a dois problemas distintos: o problema dos servidores-réplica e o problema dos objectos-réplica. O primeiro lida com a selecção de locais na rede onde colocar servidores-réplica, isto é,

servidores que contém réplicas dos objectos que se encontram nos servidores originais. O segundo prende-se com a escolha de qual ou quais os servidores-réplica em que devem ser colocadas réplicas de um determinado objecto e com os mecanismos inerentes à criação das réplicas.

Intuitivamente, os servidores-réplica devem ser colocados tão próximo quanto possível dos utilizadores finais, reduzindo dessa forma a latência percebida por estes aquando da obtenção de um determinado conteúdo, bem o consumo médio de largura de banda na rede. Existem várias abordagens teóricas que visam modelar este problema, as quais são baseadas ou consistem em variações do *problema de colocação ao centro*. Devido à elevada complexidade computacional destes algoritmos, foram definidas heurísticas. Estas, que constituem algoritmos sub-óptimos para a resolução do problema em questão, levam em conta a informação existente acerca da CDN, tal como padrões de carga (origem e destino dos pedidos feitos) ou a topologia da rede.

A escolha de que servidores-réplica devem albergar uma réplica de um determinado objecto tem como propósito melhorar a qualidade do serviço proporcionado aos clientes, lidando com as contingências impostas à CDN, o que habitualmente implica uma utilização eficiente dos recursos que esta tem ao seu dispor. Por conseguinte, os algoritmos de selecção de qual o subconjunto dos servidores de uma CDN em que réplicas de um determinado objecto devem existir são habitualmente função dos padrões de utilização dos clientes, bem como do padrão de actualização das réplicas.

Podem ser utilizados vários mecanismos para informar um servidor-réplica aquando da criação de uma nova réplica que este deve albergar. Os mecanismos mais habitualmente usados para este propósito são *pull-based caching* e *push replication*: o primeiro prende-se com a criação de réplicas de acordo com solicitações por parte dos servidores-réplica (isto é, quando estes têm se servir uma réplica ao cliente que não têm armazenada, sendo forçados a requisitá-la ao servidor), enquanto que o segundo se prende com a difusão de réplicas dos servidores para os servidores-réplica.

2.1.4 Consistência dos Dados

Um dos princípios presentes na base das CDN é o de responder a pedidos de clientes com conteúdos tão actualizados quanto possível. Manter a consistência entre as réplicas de um determinado objecto pode introduzir um *overhead* significativo ao sistema, em especial quando a aplicação lida com um número de réplicas de objectos elevado e requer consistência forte, isto é, quando os clientes são intolerantes a dados que não sejam *frescos*.

Dados os requisitos de consistência de uma determinada aplicação, é necessário determinar quais os *modelos de consistência*, as *politicas de consistência* e os *mecanismos de distribuição de conteúdos* que os satisfazem. Um modelo de consistência define quais as propriedades de consistência dos objectos distribuídos pelos sistemas aos seus clientes, as quais são geralmente função de atributos como o seu tempo de existência, o seu valor ou a quantidade de transacções sobre este executadas. Um modelo de consistência é habitualmente adoptado por uma politica de consistência que, com base neste, define como, quando e quais os mecanismos

de distribuição de conteúdos a serem aplicados. Estes últimos especificam os protocolos através dos quais os servidores-réplica trocam actualizações de objectos entre si.

A título de exemplo, podemos considerar uma rede de distribuição de conteúdos com o propósito de optimizar a latência percebida pelos clientes de um conjunto de sítios da *World Wide Web*. Este sistema pode adoptar um modelo de consistência baseado no tempo de existência das réplicas e empregar uma politica que garante que os utilizadores nunca serão servidos com réplicas cujo tempo de existência seja superior a uma hora relativamente ao estado mais recente dos objectos originais. Esta política pode ser aplicada por diferentes mecanismos de distribuição de conteúdos.

2.1.5 Encaminhamento de Pedidos

O sistema de encaminhamento de pedidos prende-se com a selecção de um servidor réplica adequado, no contexto de uma CDN, para servir um determinado pedido de um utilizador final. O sistema encaminha os pedidos dos utilizadores para os servidores mais próximos que estejam disponíveis e que, provavelmente, tenham o conteúdo pretendido. Consequentemente, é necessário lidar com os seguintes assuntos:

- Determinar a distância entre o utilizador que faz o pedido e o servidor. A distância é habitualmente função da topologia da rede, sendo mais frequentemente utilizados como métricas o número de *hops* (isto é, o número de nós por onde passa um pacote de dados desde a sua origem ao seu destino) ou o tempo de *round-trip* (ou seja, o tempo que leva um nó A leva a fazer um pedido a um nó B e a receber a respectiva resposta). Os métodos mais frequentemente utilizados para determinar estas métricas são, respectivamente, o *traceroute* e o *ping*. Todavia, nenhuma destas métricas é suficientemente precisa para determinar a proximidade entre dois nós, uma vez que o número de *hops* não entra em linha de conta com o tráfego existente na rede e o tempo de *round-trip* é altamente variável.

- Determinar a disponibilidade de um determinado servidor. A disponibilidade é uma função do volume de pedidos com que o servidor lida a cada instante (isto é, a carga) e da banda disponível na rede. Por conseguinte, considera-se que um servidor sujeito a demasiada carga num dado instante ou que está a servir pedidos próximo do seu limite de largura de banda se encontra indisponível para servir mais clientes. As técnicas mais comuns para determinar a disponibilidade de um cliente são o *server push* e o *client probe*. De acordo com a primeira técnica, os servidores propagam a informação da carga a que estão sujeitos ao sistema de encaminhamento de pedidos. Segundo a segunda técnica, o sistema de encaminhamento de pedidos sonda o estado dos servidores periodicamente.

- Determinar a probabilidade de um servidor conter um determinado conteúdo. Este factor é função de um conjunto de politicas inerentes ao funcionamento da CDN,

nomeadamente o tipo de conteúdos servidos por um dado servidor, a taxa de actualização dos conteúdos, os recursos a nível de armazenamento que este tem disponível, entre outros.

São utilizadas várias técnicas distintas para encaminhar os clientes para um servidor em particular de entre um conjunto de servidores-réplicas. Estas podem ser agrupadas em cinco categorias distintas: multiplexagem no cliente, redirecção por HTTP, indirecção DNS, *anycasting* e encaminhamento peer-to-peer.

2.2 Distribuição de Servidores

O problema de distribuição de servidores consiste na escolha de um determinado subconjunto de potenciais locais, no contexto de uma rede, para a colocação de servidores-réplica de forma a minimizar a latência média do acesso a conteúdos percebida pelos clientes, bem como o consumo global de largura de banda na transferência de réplicas de servidores para clientes. Ambas as métricas estão inversamente correlacionadas, pelo que optimização de uma delas leva geralmente à minimização da outra. Nesta secção será feita uma análise às principais propostas para solução deste problema.

O problema da distribuição de servidores pode ser modelado com recurso à teoria dos grafos: ao modelar a rede em análise como um grafo ligado, não direccionado e ponderado, assumindo também que cada cliente só faz pedidos ao servidor-réplica mais próximo de si, é possível mapear este problema num *problema de colocação ao centro*. De uma forma sumária, é possível definir este problema do seguinte modo: para a colocação de um determinado número de *centros* (que, no contexto em análise, consistem em servidores-réplica), consideramos a métrica da minimização da distância máxima entre um *nó* (ou seja, um cliente que tem o poder de fazer pedidos) e o seu centro mais próximo.

De forma a discutir as diferentes abordagens ao problema, consideram-se os seguintes pressupostos e definições:

- Em cada uma das abordagens, o problema é mapeado num grafo ligado dirigido *G(V,E)*, em que *V* define o conjunto dos vértices (isto é, centros ou nós no contexto do problema) e *E* o conjunto de arestas (ou seja, ligações entre os nós ou centros da rede).
- Define-se *N* como o número total de vértices, isto é, $N = |V|$.
- Define-se *V'* como um subconjunto de *V* contendo todos os vértices em que se podem estabelecer centros.
- Define-se o peso de uma aresta $e_{i,j}$ como o custo de associação entre os vértices *i* e *j* (em que um constituirá um centro e o outro um nó) de acordo com uma determinada métrica (como a latência de pedidos entre os dois nós, o número de saltos, etc.); note-se que os algoritmos considerados operam da mesma forma independentemente da métrica considerada.

- Define-se que cada nó (isto é, cada vértice de *V* que não consiste num centro) deve ser associado a um centro.

2.2.1 Abordagens Teóricas

O problema descrito, também conhecido por *minimum K-center*[JJJ+00], é NP-completo. Nesta secção abordar-se-ão duas variantes distintas deste problema, por uma questão de completude, de uma forma tão sintética quanto possível.

2.2.1.1 Facility Location Problem

Tendo em conta as notações e assumpções previamente definidas, de acordo com o *Facility Location Problem*[QPV01], considera-se que a colocação de um centro num determinado vértice *i* pertencente a *V'* incorre num custo f_i. Este algoritmo determina também que a associação de um determinado nó *j* a um centro *i* incorre num custo $d_j e_{i,j}$ em que d_j denota uma métrica associada ao nó *j* (como, por exemplo, a quantidade de pedidos que este origina).

O objectivo desta abordagem consiste na definição de um número de centros e das suas localizações tal que a função global de custo seja mínima.

2.2.1.2 Minimum K-median Problem

De acordo com o *Minimum K-median Problem*[Pen06], é definido um número inteiro positivo *k* que consiste na quantidade de centros a serem definidos no grafo. À semelhança do *Facility Location Problem*, este algoritmo define que a associação de um nó *j* a um centro *i* incorre num custo $d_j e_{i,j}$ em que d_j denota uma métrica associada ao nó *j* (como, por exemplo, a quantidade de pedidos que este origina).

O objectivo do *Minimum K-median Problem* consiste na determinação de um conjunto de *k* centros tal que a função global de custo seja mínima. A diferença entre ambas as abordagens consideradas reside no facto de que a primeira determina o número e locais onde devem ser constituídos centros (recorrendo, para o efeito, a um custo associado à construção de um centro num determinado vértice), enquanto que a segunda define a localização óptima de um número de centros previamente determinado (desprezando, por conseguinte, o custo de construção de um centro).

2.2.2 Abordagens Heurísticas

As abordagens teóricas consideradas são *NP*-completas, o que torna extremamente difícil a sua aplicação na prática ou mesmo inviável para sistemas CDN reais. Por conseguinte, foram propostas abordagens algorítmicas sub-óptimas ao problema de colocação ao centro e abordagens heurísticas que tiram partido da informação existente acerca da rede (tal como padrões de tráfego de pedidos ou a própria topologia da rede), as quais produzem soluções fazíveis em tempo polinomial.

Nesta subsecção serão discutidas algumas dessas abordagens, introduzidas na literatura. Note-se que as abordagens algorítmicas a serem discutidas consistem em aproximações simples ao problema de colocação ao centro, na medida em que é estabelecida uma restrição ao número total de servidores na rede mas não à quantidade de pedidos que estes servem. Considera-se esta abordagem razoável uma vez que, na prática, o aumento do número de localizações para servidores-réplica é significativamente mais complexo do que o aumento da capacidade de um determinado local onde estes já estejam estabelecidos (o qual é facilmente resolvido através de expansão vertical ou horizontal, isto é, respectivamente a substituição de máquinas existentes por máquinas melhores ou o aumento do número de máquinas num dado local).

2.2.2.1 Algoritmo Baseado em Árvores

Este algoritmo baseado em árvores, proposto por X. Deng et al.[DLGI99], baseia-se na assumpção de que as topologias subjacentes a uma determinada rede são árvores, tendo sido modelado como um problema de programação dinâmica. Originalmente desenhado para colocação de *proxies* na Web, é também aplicável à colocação de servidores-réplica para uma CDN.

De uma forma sucinta, o algoritmo consiste na divisão da árvore T em várias árvores de menor dimensão T_i, mostrando que a melhor forma de colocar t (em que $t > 1$) centros na árvore T é dada pela melhor colocação de t'_i centros em cada uma das sub-árvores T_i em que $\Sigma_i\ t'_i = t$. O algoritmo produz uma solução óptima quando as topologias subjacentes são, de facto, árvores e os pedidos dos nós são feitos ao centro mais próximo no caminho entre estes e o servidor original (ou seja, o cliente só pode fazer pedidos a um único centro). Todavia, ambas as assumpções feitas podem impedir que seja encontrada uma solução de distribuição de servidores melhor, o que limita fortemente a eficiência do algoritmo em condições que não as enunciadas.

A complexidade computacional deste algoritmo é de $O(N^3K^2)$.

2.2.2.2 Algoritmo *Greedy*

Um algoritmo *greedy* foi proposto por P. Krishnan et al.[KPS00] para o problema da localidade nas *caches*. Este foi adaptado por Padmanabhan et al.[QPV01] para o problema da colocação de servidores-réplica em redes CDN.

De acordo com este algoritmo, pretende-se seleccionar *K* de entre *N* vértices para estabelecer o conjunto de centros. Na primeira iteração, consideram-se cada um dos *N* vértices de forma a avaliar a sua adequação ao papel de centro – para o efeito, determina-se a função total de custo para cada potencial centro, supondo que os pedidos de todos os outros nós convergem para o mesmo, escolhendo-se o nó com a função de custo mais baixa. Nas restantes *K-1* iterações procura-se o nó que, em conjunto com os escolhidos para os papéis de centro nas iterações anteriores, resulta num custo total mais baixo. Em geral, para a determinação do custo, assume-se que cada nó acede apenas ao centro mais próximo de si. São feitas, no total, *K* iterações, altura em que terão sido determinados *K* centros.

A complexidade computacional deste algoritmo é de $O(N^2K)$.

2.2.2.3 Algoritmo Aleatório

O algoritmo aleatório é alheio à carga de pedidos na rede, escolhendo aleatoriamente *K* centros de entre *N* vértices. De forma a melhorar a *performance*, o algoritmo é habitualmente executado várias vezes para que se escolha o conjunto de centros que resulte num custo total mais baixo.

A complexidade computacional deste algoritmo é de $O(NK)$.

2.2.2.4 Algoritmo *Hot Spot*

O algoritmo *Hot Spot* tem como propósito a colocação de centros junto dos nós que geram o maior volume de carga. Por conseguinte, os *N* possíveis centros são listados decrescentemente de acordo com o volume de tráfego gerado na sua vizinhança, sendo escolhidos para centros os *K* vértices que mais tráfego geram. A vizinhança de um determinado nó *i* é definida como o círculo centrado em *i* com um dado raio.

A complexidade computacional deste algoritmo é de $N^2 + min(NlogN + NK)$.

2.3 Distribuição de Réplicas

No âmbito da distribuição de conteúdos, é possível analisar dois sub-problemas distintos: a colocação de réplicas e a criação de réplicas. O primeiro prende-se com a selecção de um conjunto de servidores-réplica que devem guardar a réplica de um determinado objecto enquanto que o segundo, por seu turno, se prende com a selecção de um mecanismo para notificar um servidor-réplica acerca de criação de novas réplicas.

2.3.1 Colocação de Réplicas

À semelhança do problema da distribuição de servidores, o problema de colocação de conteúdos pode ser modelado como um problema de colocação ao centro, o que inviabiliza o recurso a algoritmos que produzem soluções óptimas dado que o problema é *NP*-completo. Todavia, uma vez que os algoritmos de colocação de réplicas de conteúdos são executados com uma frequência bastante superior à de execução de algoritmos para a colocação de servidores, a adopção de aproximações a este problema pode também ser difícil uma vez que estas são, por norma, computacionalmente complexas. Por conseguinte, surge a necessidade de se recorrer a soluções mais simples.

As abordagens discutidas nesta subsecção são gerais, pelo que estratégias adoptadas para tipos específicos de conteúdos (como, por exemplo, o uso de *multicast* para a disseminação de conteúdos ricos) serão analisadas em secções subsequentes deste relatório.

2.3.2.1 Heurísticas de Replicação

Kangasharju et al.[KRR01] modelaram a situação em análise como um problema combinatorial de optimização, cujo objectivo consiste na colocação de *K* réplicas em *N* servidores, num esforço de minimização dos saltos entre nós da rede que devem ser percorridos por um pedido para que este seja servido. Assume-se, nesta abordagem, que cada servidor-réplica consiste num nó com capacidade finita para replicação de objectos, pelo que as restrições associadas ao armazenamento de informação nos servidores devem também ser respeitadas. O problema em questão é *NP*-completo, pelo que foram propostas quatro heurísticas para o endereçar (de entre as quais se dá mais ênfase às últimas três, visto que a primeira consiste numa aproximação relevante sobretudo para testes de comparação de *performance* das mesmas).

A primeira heurística proposta consiste numa abordagem aleatória: são associadas réplicas de objectos a servidores-réplica, respeitando as restrições de armazenamento estabelecidas. Escolhe-se então um objecto e um servidor-réplica para onde este deve ser replicado, de acordo com probabilidades uniformes. Caso o servidor-réplica escolhido já contenha uma réplica do objecto em questão, escolhe-se então um novo objecto e um novo

nó. Como resultado, um objecto pode ser associado a vários nós, mas um nó tem no máximo uma réplica de um determinado objecto.

A segunda heurística define que cada servidor-réplica armazena réplicas dos objectos mais populares entre os seus clientes. Por conseguinte, cada servidor-réplica ordena os vários objectos decrescentemente de acordo com a sua popularidade, a qual é estimada através dos pedidos feitos pelos seus clientes, armazenamento réplicas dos primeiros *K* objectos que forem possíveis dadas as suas restrições de armazenamento.

A terceira heurística determina que cada servidor-réplica armazena réplicas dos *K* nós possíveis de armazenar, dadas as suas restrições de armazenamento, que maximizem uma determinada função de custo. Essa função de custo é dada pelo produto entre a popularidade de um objecto, obtida da mesma forma do que na segunda heurística, com a distância do servidor-réplica em análise ao servidor-fonte de cada conteúdo (assumindo que conteúdos diferentes podem residir em diferentes servidores-fonte). A ideia subjacente a esta heurística é a de que cada servidor-réplica deve armazenar réplicas dos conteúdos mais populares e que estejam também mais longe do seu servidor-fonte, num esforço de minimização da latência percebida pelo cliente. Esta heurística procura então minimizar a distância entre os clientes e os objectos pedidos, sendo alheia à presença de outros servidores-réplica que adoptam o mesmo comportamento.

A quarta heurística procura ultrapassar a limitação da terceira heurística ao determinar a localização das réplicas de uma forma globalmente coordenada. Por conseguinte, a rede CDN, ao invés de cada servidor-réplica em particular, determina uma função de custo $C_{i,j}$ para cada servidor-réplica *i* e para cada objecto *j*. É então escolhido, para cada objecto, o servidor-réplica que maximiza a função de custo para alojar uma réplica do mesmo. A função de custo consiste no produto da popularidade de cada objecto, na distância de cada servidor-réplica ao servidor-fonte do objecto e na quantidade total de pedidos recebidos por cada servidor-réplica.

De acordo com simulações, os autores demonstram que a quarta heurística apresenta uma eficiência superior à das restantes (de notar que a heurística aleatória é substancialmente pior do que todas as outras), sendo a sua principal contrapartida a elevada complexidade computacional que lhe é inerente.

2.3.2.2 Colocação Dinâmica de Réplicas

Chen et al.[CKK02] propuseram um novo sistema de distribuição de conteúdos dinâmicos na a *Web* denominado *dissemination tree* (árvore de disseminação). De acordo com este sistema, assume-se a existência de um único servidor-fonte, uma vez que os conteúdos a serem disseminados são dinâmicos, bem como um conjunto de servidores-réplica, responsáveis pelo armazenamento de réplicas de conteúdos originais que se assumem sempre actualizadas. Ambos os componentes – servidor-fonte e servidores-réplica auto-organizam-se de acordo com uma topologia em árvore, utilizando *multicast* ao nível aplicacional para disseminar cópias da fonte para as réplicas. Este sistema recorre também a uma infra-estrutura de localização e encaminhamento denominada *Tapestry*[ZKJ01]: esta infra-estrutura

overlay oferece encaminhamento de mensagens, independentemente da sua localização, para a cópia mais próxima de um objecto ou serviço, recorrendo apenas a ligações ponto-a-ponto e sem recursos centralizados.

O algoritmo subjacente ao sistema descrito tem como propósito a colocação dinâmica de réplicas, organizando-as numa árvore *multicast* com conhecimento limitado ao nível da topologia de rede. Este procura satisfazer as restrições associadas à latência dos clientes, bem como à capacidade dos servidores (tanto do servidor-fonte como dos servidores-réplica), tendo como objectivo a minimização do número de réplicas, associadas a um dado objecto, existentes.

De acordo com este algoritmo, ao fazer um pedido, cada cliente é associado a um servidor pai que o serve. Numa fase inicial, um determinado cliente *c* faz um pedido à rede que é encaminhado, através da infra-estrutura *Tapestry*, para um servidor *s* que contém uma réplica do objecto pedido. Segundo uma abordagem ingénua, caso as restrições de carga de *s* e de latência de *c* sejam respeitadas, *s* passa a ser o pai de *c*, servindo-o; caso contrário, *s* procurará colocar a réplica num servidor *s'* tão próxima de *c* quanto possível, para que este último passe a ser o seu pai. Esta abordagem tem a desvantagem de nem sempre encontrar o melhor pai para *c*, mesmo que este exista. Por conseguinte, foi desenvolvida uma abordagem inteligente, segundo a qual *c*, ao ser encaminhado para um servidor *s*, avalia também o pai, irmãos e eventuais filhos forma a determinar aquele com menor carga para que possa ser seu pai; caso nenhum dos servidores avaliados respeite as restrições necessárias, *s* procurará colocar a réplica tão longe de *c* quanto possível, no caminho *overlay* (este processo denomina-se *lazy placement*). Todos estes passos têm como objectivo distribuir a carga pela rede de forma a reduzir o número de réplicas necessárias, satisfazendo as restrições impostas.

2.3.2 Criação de Réplicas

Dos diferentes métodos usados para a criação de réplicas, destacam-se os *pull-based caching* e *push replication*[SSPS04], uma vez que são os mais usados a nível global.

De acordo com o *pull-based caching*, o modo de funcionamento das CDN, no que diz respeito à criação de réplicas de objectos, assemelha-se ao que se verifica de forma mais comum nas caches, independentemente dos seus contextos aplicacionais. Por conseguinte, quando um determinado servidor-réplica recebe um pedido, oriundo de um cliente arbitrário, para um objecto cuja réplica, naquele momento, não tem armazenada, trata a ocorrência como uma *falha*. De forma a satisfazer o pedido do cliente, o servidor-réplica procede então à obtenção de uma réplica do referido objecto junto do servidor-fonte, que a cria para o efeito. Consequentemente, a criação de uma réplica é adiada até ao primeiro pedido para obtenção dessa réplica. Note-se que neste caso, este método é apenas utilizado como um mecanismo para a criação de réplica: a decisão de colocação da réplica num determinado servidor-réplica está a cargo do sistema, o qual é responsável pelo encaminhamento dos pedidos dos clientes para os servidores-réplica mais adequados.

O método *push replication* é sobretudo usado em contextos em que os objectos armazenados nas CDN apresentam uma variância baixa. Nestes ramos aplicacionais específicos, entra-se em linha de conta com o conceito de época: esta define intervalos de tempo durante os quais os conteúdos da CDN não são actualizados. Desta forma, a responsabilidade de criação de réplicas e notificação dos servidores-réplica pode estar a cargo do sistema: a cada época, os servidores-fonte *empurram* explicitamente os conteúdos a serem actualizados para junto dos servidores-réplica.

As duas abordagens discutidas serão analisadas, em maior detalhe, na subsecção 2.4.2.2 deste artigo, que diz respeito à direcção de actualização de réplicas numa CDN, um assunto que se prende com a consistência das mesmas.

2.4 Consistência de Réplicas

A consistência das réplicas é um dos aspectos mais importantes no contexto das CDN, uma vez que a fidelidade dos dados acedidos pelos clientes depende das propriedades de consistência garantidas pelo sistema. Garantir a consistência de réplicas no contexto de uma CDN é um problema que diz respeito à selecção se *modelos de consistência*, os quais podem ser implementados com recurso a várias *politicas de consistência* que, por seu turno, podem recorrer a diversos *mecanismos de distribuição de conteúdos*. Um modelo de consistência consiste num contrato celebrado entre a CDN e os seus clientes que dita as propriedades associadas à consistência dos conteúdos distribuídos pela mesma (habitualmente relacionadas com o seu tempo de existência, valor ou número de transacções executadas). Com base no modelo de consistência definido, uma política de consistência tem como propósito definir como, quando e para que réplicas devem ser aplicados os vários mecanismos de distribuição de conteúdos. Note-se, no entanto, que um único modelo de consistência pode ser aplicado com recurso a várias politicas. Por fim, um mecanismo de distribuição de conteúdos é um método ou protocolo segundo o qual os servidores-réplica trocam actualizações entre si, definindo a forma como estas são transferidas, quem dá início à transferência e quando as actualizações ocorrem.

2.4.1 Modelos de Consistência

Os vários modelos de consistência diferem entre si essencialmente pelo rigor que exigem relativamente à consistência de réplicas. Por consistência forte, entende-se que o sistema garante que todas as réplicas de um determinado objecto são idênticas a partir da perspectiva dos clientes. Se uma determinada réplica não é consistente com as restantes, não pode ser acedida pelos clientes até que seja actualizada. Devido aos elevados custos de sincronização de réplicas habitualmente inerentes à consistência forte, esta é raramente utilizada em redes de grande escala. Em oposição, a consistência fraca permite que as réplicas de um determinado objecto difiram, mas assegura que todas as actualizações chegam a todas as réplicas dentro de um determinado espaço de tempo. Uma vez que este modelo é resistente a atrasos na

propagação de actualizações e incorre em custos de sincronização inferiores, é mais adequado a redes de grande escala.

2.4.1.1 Single Master ou Multiple Master

Um modelo de consistência pode ser classificado como *single master* ou *multiple master*[SSPS04] dependendo se as actualizações são originadas, respectivamente, por um único servidor-fonte ou por vários. Os modelos *single master* definem uma única máquina como responsável por conter uma versão actualizada de um dado objecto. Estes modelos são simples e adequam-se bem a aplicações em que a natureza dos objectos implica uma única fonte de actualizações como, por exemplo, redes em que os conteúdos servidos são essencialmente estáticos. Os modelos *multiple master*, por seu turno, permitem que mais do que um servidor possa modificar o estado de um objecto. Estas abordagens são especialmente adequadas a redes em que objectos replicados podem ter o seu estado modificado como resultado de um acesso de um cliente. Todavia, estes modelos podem introduzir problemas como a resolução de conflitos de actualizações. Na prática, existe pouco trabalho realizado nos modelos *multiple master* no contexto das redes CDN.

2.4.1.2 Tipos de Consistência

Os modelos de consistência definem as propriedades de consistência das réplicas numa CDN de acordo, fundamentalmente, com três métricas distintas: tempo, valor e volume de transacções.

Torres-Rojas et al.[TAR99] definiram formalmente um conjunto de modelos que garantem a consistência de um conjunto de réplicas com base em tempo real. Estes modelos têm como propósito definir quão rapidamente os efeitos de uma dada operação são percebidos pelo resto do sistema. Por conseguinte, os modelos de consistência baseados em tempo requerem que, se uma operação for executada num determinado instante de tempo t, esta deve ser visível pelos restantes servidores até ao instante $t + \Delta$. Estes modelos de consistência satisfazem as necessidades de um conjunto de aplicações que necessitam de observar as actualizações experimentadas por objectos alteráveis dinamicamente dentro de determinadas restrições temporais. A consistência baseada em tempo é aplicável a todo o tipo de objectos, podendo ser aplicada segundo diferentes mecanismos de distribuição tal como *polling* (em que um servidor-réplica verifica periodicamente a existência de uma nova actualização) ou invalidação por parte do servidor (em que o servidor invalida uma réplica na posse de um determinado servidor-réplica caso o objecto original tenha sido, entretanto, actualizado). A título de exemplo, V. Cate propôs um sistema, denominado *Alex*, que tem como propósito permitir o acesso de leitura transparente, por parte de clientes e aplicações, a ficheiros contidos em sítios FTP remotos na Internet como se estes estivessem armazenados localmente. De forma a garantir um determinado nível de *performance*, este sistema recorre a caching, adoptando um modelo controlo de consistência de réplicas baseado em tempo real.

A política de consistência garante que as únicas actualizações que podem não ter ainda sido reflectivas numa *cache* são aqueles que aconteceram nos últimos 10% da idade do ficheiro.

Os métodos de consistência baseada em valor têm como objectivo assegurar que a diferença entre o valor de um determinado objecto e os valores das suas réplicas não diferem mais do que um determinado Δ. Por conseguinte, estes métodos podem apenas ser aplicados a objectos que têm uma definição precisa de valor como, por exemplo, um objecto que defina a quantidade de lugares reservados num determinado avião. Este método pode também ser implementado com recurso a mecanismos de *polling* (assumindo que a variação sofrida pelo objecto é majorada para um determinado intervalo de tempo, o qual é usado para a frequência de *polling*) ou invalidação por parte do servidor.

A garantia de consistência com base em volume de transacções é geralmente explorada em bases de dados replicados. Estes modelos definem uma transacção como qualquer operação de leitura ou escrita sobre um determinado objecto, permitindo que réplicas operam em diferentes estados se as transacções fora-de-ordem aderirem a regras definidas por estas políticas (as quais estão habitualmente associadas a relações de dependência entre objectos). A implementação deste tipo política de consistência requer o que os mecanismos de distribuição de conteúdos troquem transacções das várias réplicas entre os servidores-réplica, as quais devem ser acompanhadas de marcadores temporais. Este tipo de consistência é aplicável a um grupo de objectos que, conjuntamente, constituem uma base de dados regularmente actualizada. Krishnakumar et al.[KB94] introduziram o conceito de transacções ignorantes de ordem *N*, que consiste na aplicação de mecanismos de consistência com base em volumes transacção, em que uma transacção pode ser levada a cabo sobre uma réplica enquanto ignorar *N* transacções prévias sobre outras réplicas.

Yu et al.[YV02] propuseram um modelo contínuo de consistência que integra todas as abordagens descritas. De acordo com estes modelo, não são explorados apenas os modelos extremos de consistência forte ou fraca, mas também abordagens intermédias. O espaço entre os modelos de consistência forte e fraca é então explorado ao permitir que diferentes aplicações especifiquem o nível de consistência desejado utilizando unidades a que se dá o nome de *conits*, que definem unidades físicas ou lógicas de consistência. Estes modelos usam um vector tridimensional de forma a quantificar a consistência: este define um *erro numérico* (utilizado para a implementação de consistência baseada em valor), um *erro de ordem* (utilizado para a implementação de esquemas baseados em quantidade de transacções) e um *intervalo de tolerância* (utilizado para consistência baseada em tempo real). Se cada uma destas métricas estiver limitada com o valor 0, a consistência é máxima. De uma forma dual, se não forem definidos limites, a consistência é não existente. O modelo permite que diferentes aplicações definam os seus requisitos de consistência, permitindo a definição de diferentes garantias e limites relativamente a diferentes servidores-réplica. Este aspecto é especialmente adequado para servidores-réplica com largura de banda limitada poderem aplicar princípios relaxados de consistência, enquanto que servidores-réplica com melhor conectividade podem beneficiar de uma consistência mais forte.

2.4.2 Mecanismos de Distribuição de Conteúdos

Os mecanismos de distribuição de conteúdos definem como os servidores-réplica trocam actualizações entre si. Estes diferem em relação a dois aspectos: a forma como a actualização é feita e a direcção segundo a qual esta é despoletada. A decisão acerca destes dois aspectos influência o nível de consistência atingível pelo sistema, bem como o *overhead* de comunicação necessário para que este seja mantido.

2.4.2.1 Formas de Actualização

As actualizações de réplicas podem ser transferidas através de três formas distintas: *state shipping*, *delta shipping* e *function shipping*[SSPS04].

- *State Shipping*: aquando da actualização uma réplica toda ela é enviada. A principal vantagem deste método consiste na sua simplicidade. Todavia, pode incorrer num *overhead* de comunicação significativo, em especial quando as actualizações são feitas sobre objectos de grandes dimensões.
- *Delta Shipping*: apenas as alterações relativamente ao estado anterior de um objecto são enviadas. Consequentemente, a abordagem *delta shipping* surge com o propósito de reduzir o *overhead* de comunicação inerente ao método *state shipping* ao enviar, aquando da actualização de uma réplica, apenas o seu *delta*. Apesar de existir uma redução do *overhead* de comunicação relativamente à abordagem *state shipping*, este método requer que cada servidor-réplica tenha disponível a versão anterior de cada réplica. Assume-se também que as diferenças entre duas versões distintas de um dado objecto podem ser rapidamente serializáveis.
- *Function Shipping*: é enviada a listagem de acções necessárias para evoluir uma réplica do seu estado anterior para o estado actual. É então ultrapassada a barreira imposta pela necessidade de serialização dos deltas de duas versões de um dado objecto. O *overhead* de comunicação é, geralmente, inferior ao das duas abordagens anteriormente descritas, uma vez que a listagem de acções é independente do estado e tamanho de cada objecto. Todavia, a transferência de acções obriga a que os vários servidores-réplica as executem, o que, dependendo das acções em questão, pode ser computacionalmente exigente.

2.4.2.2 Direcção de Actualização

Nesta subsecção serão abordados os princípios básicos subjacentes à direcção da actualização de réplicas numa CDN. A transferência de actualizações pode ser despoletada por um servidor-réplica que necessite de uma nova versão de um determinado objecto e pretenda obtê-lo junto dos seus pares (método *pull*), ou pelo servidor-réplica que detém uma nova versão de um objecto e que o pretende enviar para os seus pares (método *push*)[SSPS04].

Segundo o método *pull*, cada elemento de dados está associado a um atributo *Time To Refresh* (*TTR*), que determina o próximo instante que os dados devem ser revalidados. O valor do TTR pode ser constante ou determinado a partir da taxa média de actualização dos dados, podendo também ser dependente dos requisitos de consistência do sistema. Por conseguinte, dados que requerem taxas de actualização elevadas e/ou requisitos de consistência forte devem ter um TTR baixo, enquanto que dados com uma frequência menor de actualizações podem ter um TTR elevado. A vantagem deste mecanismo reside no facto de que não requer o armazenamento de informação de estado, o que resulta numa elevada tolerância a faltas. Em contrapartida, a aplicação de consistência forte depende de uma estimação cuidada do TTR: valores baixos podem oferecer boa consistência, mas sob pena de transferências desnecessárias quando o documento não é actualizado.

De forma a suprimir o *overhead* desnecessário de comunicação é utilizada, na prática, uma combinação de TTR e de uma verificação da validade de um documento no servidor utilizado. Juntamente com o pedido de actualização de uma réplica, junta-se um campo *se-modificado-desde*, o qual contém a data de modificação da réplica: se o objecto original tiver sido actualizado desde então, é enviada uma réplica do mesmo; caso contrário, é enviada apenas a informação de que a réplica permanece actualizada. Este método permite a aplicação de consistência forte ao reduzir o *overhead* de comunicação sem o eliminar, uma vez que o servidor-fonte deve ser periodicamente contactado a cada pedido de forma a verificar se as réplicas de cada servidor-réplica estão actualizadas.

O método *push* assegura que a comunicação ocorre apenas quando um determinado objecto é actualizado. Consequentemente, o principal argumento deste método reside no facto de que podem ser satisfeitos requisitos de consistência forte sem introduzir o *overhead* de comunicação inerente à abordagem *se-modificado-desde*: uma vez que a comunicação é iniciada por um servidor ciente da actualização feita, é possível determinar exactamente quais as actualizações a disseminar para rede e quando. Todavia, uma limitação subjacente a este método reside no facto de que o servidor-fonte deve manter o registo dos servidores-réplica a serem informados a cada actualização de um determinado objecto. Uma limitação mais importante prende-se ainda com a existência de um ponto único de falha: caso o servidor que detém um estado mais actualizado de uma dada réplica falhe, a consistência do sistema é comprometida uma vez que a actualização não é disseminada até que este recupere.

A distribuição de conteúdos do tipo *push* pode ser associada com *leases*: de acordo com estas abordagens, um dado servidor-réplica regista o seu interesse por um determinado objecto durante um intervalo de tempo (*lease*) que lhe é associado. Enquanto o tempo de *lease* não expirar, o servidor-fonte desse objecto envia todas as actualizações para o servidor-réplica. Os *leases* podem ainda ser distinguido em três grupos: baseados em idade, frequência de verificação de renovação e carga. Nos primeiros, o tempo de *lease* depende da última vez que o objecto foi modificado, estando subjacente a assumpção de que, caso o intervalo de tempo desde a sua última actualização seja *longo*, o período de espera até à próxima actualização deverá ser também *longo*. Ao ser atribuído um tempo de *lease* proporcional à frequência com que cada servidor-réplica verifica se um determinado objecto

foi actualizado, supõe-se que aqueles que pedem a verificação das suas réplicas mais frequentemente devem ter tempos de *lease* superiores uma vez que se supõem *mais interessados* nos objectos. Finalmente, nos *leases* baseados em carga, o servidor-fonte tende a atribuir temos menores quanto maior for a carga a que está sujeito, reduzindo assim o número de servidores para os quais têm de ser enviadas actualizações.

Bhide et al.[BDK+02] propuseram um conjunto de três combinações diferentes das abordagens *push* e *pull*. De acordo com a primeira, *Push-and-Pull* (*PaP*), ambas as estratégias são utilizadas simultaneamente para a troca de actualizações, sendo possível definir os parâmetros inerentes a cada uma delas. A segunda combinação, *Push-or-Pull* (*PoP*), permite que um servidor-fonte escolha adaptativamente entre uma abordagem *push* ou uma abordagem *pull* para cada servidor-réplica, com base nas sua ligações. Esta solução é também mas eficiente do que a *PaP*, uma vez que o servidor-fonte pode determinar o instante em que deve trocar entre *pull* e *push* dependendo dos recursos que tem à sua disposição. A terceira abordagem, *Push, Pull and Pap* (*PoPoPaP*) surge como uma versão melhorada do *PoP*, que escolhe entre *push*, *pull* e *PaP*. Por conseguinte, o servidor-fonte torna-se mais flexível, permitindo uma degradação de serviço controlada (através da troca de abordagens usadas), mantendo consistência forte.

Uma outra forma de combinar estas duas abordagens consiste em adoptar um método *push* que despoleta explicitamente, através de invalidações, ou implicitamente, através de versões, o método *pull*. As invalidações são enviadas do servidor-fonte aos servidores-réplicas quando um determinado objecto é modificado: estes últimos, caso necessitem da versão mais actualizada, procedem à sua obtenção junto do primeiro usando, por conseguinte, uma abordagem *pull*. Esta abordagem é particularmente adequadas a objectos frequentemente actualizados e raramente acedidos. Com o uso de versões, a cada objecto é associado um número de versão, incrementado a cada actualização e guardado num documento pai. Cada servidor-réplica, ao verificar que não tem a versão mais actual do objecto, procede então à sua obtenção junto do servidor-fonte.

Os mecanismos enunciados não escalam para um número elevado de réplicas: em qualquer uma destas, a comunicação entre servidores é feita de forma independente, através de conexões *unicast* independentes. Na secção 2.7 deste artigo, serão abordadas formas mais eficientes para a disseminação de conteúdos.

2.5 Pedidos de Serviços

Existem duas etapas de operações associadas ao serviço de um pedido numa CDN: a primeira prende-se com a localização de um servidor-réplica adequado que contenha uma réplica do objecto pedido, enquanto que o segundo consiste no processo de redireccionamento de um pedido para o servidor seleccionado. Nesta secção serão analisadas ambas as etapas, bem como uma estratégia denominada *anycasting* que as combina numa só.

2.5.1 Localização do Servidor

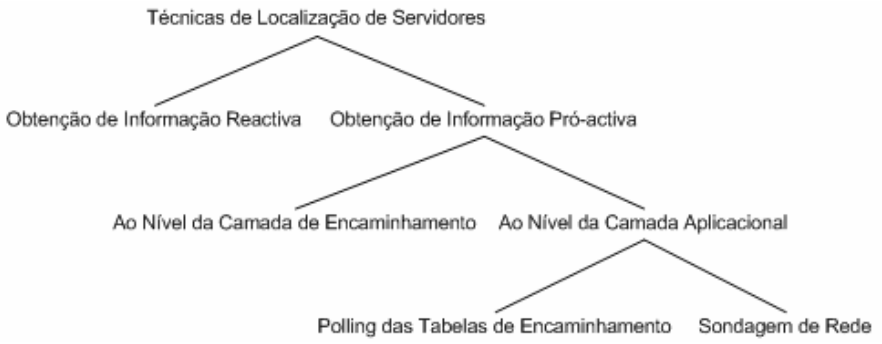


fig. 2: técnicas de localização de servidores.

Como resumido na figura 2[GS95], existem várias formas possíveis de localizar servidores que se encontrem próximos ou sejam adequados, no contexto de uma rede de larga escala. A primeira escolha no processo de localização de um servidor está relacionado com a forma como a informação respeitante à localização dos servidores é obtida. Esta pode ser obtida de forma *reactiva* ou *pró-activa*, isto é, como resposta a estímulos provenientes dos clientes ou por iniciativa própria dos agentes cujo propósito é a obtenção dessa informação. A segunda escolha determina se o processo de obtenção de informação da localização de servidores, partido do pressuposto de que é pró-activo, tem suporte ao nível da *camada de encaminhamento* (o que pode reduzir os custos de obtenção mas não ser viável na prática) ou apenas da *camada aplicacional*. Por fim, são analisadas diferentes opções para a obtenção dessa informação na camada aplicacional.

2.5.1.1 Obtenção de Informação Reactiva ou Pró-activa

Grande parte do trabalho inicial nas técnicas de localização de recursos em redes de larga escala foi desenvolvida por David Boggs. Na sua tese[Bog82] este definiu um mecanismo, *expanding ring broadcast*, que permite que as transmissões em *broadcast* possam atingir áreas concêntricas em torno do servidor de transmissão cujos raios são iterativamente crescentes (de modo a procurar um servidor adequado para servir um dado pedido que se encontre tão próximo da fonte do pedido quanto possível). As abordagens mais recentes ao problema consideram várias formas de encaminhamento *multicast*, permitindo escolher, de entre um grupo *multicast*, qual o servidor que responde mais rapidamente a um pedido. Todavia, de acordo com este tipo de abordagens, cada vez que um cliente necessita de localizar um servidor que ofereça um tipo particular de serviço, uma mensagem tem que ser enviada através de *multicast* para todo o grupo de servidores. Por conseguinte esta abordagem, que assenta sobre um princípio reactivo uma vez que a determinação de um servidor adequado para fornecer um serviço é feito com base na reacção da mensagem enviada pelo cliente, introduz desperdício de recursos de largura de banda.

Por outro lado, ao ser adoptada uma metodologia pró-activa, determinados agentes, tais como *routers* ou aplicações dedicadas, obtêm a informação da rede e dos servidores ao

longo do tempo de forma a que, quando um cliente necessita de um servidor adequado para a prestação de um determinado serviço, estes sejam capazes de determinar um ou vários sem introduzir um *overhead* na comunicação. A referida informação é obtida com recurso a mensagens de sonda enviadas a servidores candidatos ou à recolha de informação relativa à carga com que servidor lida, mantendo as suas localizações numa base de dados para o efeito.

2.5.1.2 Camada de Encaminhamento ou Camada Aplicacional

C. Partridge et al.[PMM93] propuseram um mecanismo de *anycasting*, em particular na camada IP, que procura entregar um pedido proveniente de um cliente a apenas um servidor localizado nas proximidades do mesmo. A abordagem *anycasting* é apelativa ao evitar o consumo de recursos de rede inerentes às mensagens repetidas propagadas pela rede de forma a determinar informação acerca das distâncias dos servidores. Todavia, no modelo proposto por Partridge, assume-se que, dado um grupo de servidores, todos eles oferecem iguais serviços. Esta assumpção não é válida numa CDN, em que os vários servidores-réplica podem conter réplicas de diferentes objectos, pelo que se torna necessária a programação de politicas de restrições ao nível dos *routers* encarregados do encaminhamento de mensagens. Dependendo do contexto da rede CDN, a programação deste tipo de informações ao nível de encaminhamento nem sempre é prático ou mesmo viável.

Contrastando com esta abordagem, a localização de servidores ao nível aplicacional pode incluir propriedades relacionadas com o tipo e qualidade de serviço prestado nos critérios de selecção, após ser determinado um conjunto de servidores próximos do local de onde é feito o pedido. Esta abordagem não necessita de qualquer suporte ao nível de encaminhamento, o que resulta em custos de comunicação superiores. A maior desvantagem desta abordagem reside no facto de que uma base de dados de localizações de servidores requer actualizações periódicas enquanto que, ao nível do encaminhamento, esta pode ser construída incrementalmente através da monitorização do tráfego e de entradas ou saídas de servidores-réplica no grupo.

Foram propostas algumas abordagens que procuram tirar partido dos benefícios de ambas as estratégias. São exemplos desta combinação abordagens que procuram construir uma base de dados do nível aplicacional com base na monitorização de tráfego ao nível do encaminhamento, ou abordagens exclusivas do nível de aplicação que determinam que uma actualização à base de dados deve ser realizada apenas quando o nível de carga de um dado servidor se altera significativamente (neste caso, o servidor em que a mudança de carga se verifica deve reportar a ocorrência, despoletando a actualização).

2.5.1.3 Polling de Tabelas de Encaminhamento ou Sondagem de Rede

Dado um ponto da rede a partir do qual se pretende determinar a localização relativa de servidores, é fazer-se *polling* das tabelas de encaminhamento de vários nós a partir do referido ponto até que se atinja um ou mais destinos determinados (como servidores). Desta

forma, compilando as várias tabelas de encaminhamento recolhidas, é possível formar uma imagem mais abrangente da rede, o que permite uma determinação mais eficiente de qual ou quais os servidores mais adequados para servir um determinado pedido.

Na abordagem de sondagem de rede, agentes designados para o efeito são responsáveis por explorar os caminhos até cada um dos servidores réplica através de sondagens aos mesmos. Quando um determinado cliente solicita uma lista de servidores-réplica nas suas imediações a um destes agentes, este último explora o caminho de volta ao cliente e adiciona essa informação à base de dados de conectividade. A partir desse momento, procura nas bases de dados por servidores adequados.

2.5.2 Encaminhamento de Pedidos

Nesta subsecção serão apresentadas diferentes propostas para encaminhamento de pedidos no contexto de uma CDN, nomeadamente: *multiplexagem no cliente*, *redirecção por HTTP* e *indirecção DNS*. Estes métodos são predominantemente utilizados para encaminhamento de pedidos HTTP, o que não significa que, em determinadas situações, não possam ser aplicados a outros protocolos para troca de informação em redes de computadores.

No contexto da secção de *Pedidos de Serviços*, será abordada também uma estratégia de encaminhamento denominada *anycasting*. A razão subjacente à abordagem do *anycasting* numa *subsecção* independente prende-se com o facto de que esta estratégia conjuga os procedimentos de localização de servidores e encaminhamento de pedidos.

Na secção 2.6 será também abordada a temática do encaminhamento de pedidos em redes *peer-to-peer*. A razão subjacente à separação da estratégia inerente a estas redes prende-se com o facto de este artigo se focar essencialmente em redes com uma coordenação central e com entidades prestadoras de serviços e responsáveis por distribuição de conteúdos bem definidas.

2.5.2.1 Multiplexagem no Cliente

De acordo com esta abordagem, o cliente (ou um *proxy*, que tem como propósito servir o cliente) obtém os endereços de um conjunto de servidores-réplica, escolhendo para qual envia o seu pedido. Existem três mecanismos habitualmente utilizados para a aplicação desta abordagem.

No primeiro mecanismo, o servidor DNS da entidade responsável pela prestação do serviço solicitado pelo cliente retorna um conjunto de endereços IP referentes a servidores na posse de réplicas do objecto pedido. O sistema de resolução de DNS do cliente tem então a responsabilidade de escolher um servidor de entre os retornados. De forma a tomar esta decisão, o sistema de resolução pode sondar os servidores optando com base nos tempos de resposta a estas sondas, ou pode recolher registos de utilização de outros clientes acerca da *performance* em acessos passados a estes servidores[BAZ+97]. A última medida tem como vantagem eliminar comunicações extraordinárias no processo de pedido de um serviço. Todavia, existem várias limitações relativamente a este mecanismo. Em primeiro lugar, esta

proposta requer a utilização de sistemas de resolução de DNS personalizados para o efeito: por conseguinte, caso o sistema de resolução dependa de registos de *performance* de clientes, então o cliente deve também ser modificado (para o envio dos referidos registos). Em segundo lugar, a infra-estrutura DNS apresenta uma forte dependência em relação a *caches* de DNS, pelo que os servidores não podem ser alterados frequentemente sob pena de os sistemas de resolução lidarem com informação desactualizada. Ao reduzir os tempos de *caching*, a frequência de *queries DNS* aumenta, o que pode introduzir uma limitação significativa de *performance* ao sistema.

O segundo mecanismo recorre a *applets Java* para a selecção do servidor-réplica. O URL de objecto referencia, na realidade, uma *applet Java*, a qual inclui conhecimento acerca dos servidores-réplica actuais e o procedimento necessário à escolha dos mesmos. Esta abordagem dispensa alterações ao nível dos clientes, mas envolve uma comunicação TCP extraordinária, relativamente ao primeiro mecanismo, para que seja feito o *download* da *applet*.

O terceiro mecanismo propaga a informação acerca dos conjuntos de servidores-réplica nos cabeçalhos HTTP. É necessário então que tanto os servidores como os clientes (servidores *proxy*, neste caso) sejam alterados de forma a processarem os cabeçalhos extraordinários. Os servidores *proxy* devem também estar preparados para fazer a selecção do servidor-réplica mais adequado a servir cada pedido.

2.5.2.2 Redirecção por HTTP

O protocolo HTTP permite que um servidor *Web* responda a um pedido de um cliente com uma indicação para este último re-submeter o seu pedido a um outro servidor. Este mecanismo pode ser aplicado para a elaboração de um servidor *Web* cujo propósito é receber pedidos de clientes, escolher os servidores-réplica mais adequado para os servir e reencaminhar os clientes para esses servidores.

A principal vantagem inerente a este método reside no facto de que a replicação pode ser gerida com uma granularidade fina, até às páginas *Web* individuais, enquanto que outros métodos postulam frequentemente que o grão é todo um sítio *Web*. A principal desvantagem a redirecção ao nível do protocolo HTTP é o *overhead* que adiciona à comunicação. Não só é introduzida uma mensagem extra, nos dois sentidos da comunicação, como esta mensagem é feita ao nível do protocolo HTTP, o qual usa o *pesado* protocolo TCP ao nível da camada de transporte.

2.5.2.3 Indirecção DNS

Várias implementações de servidores de nomes de domínios (DNS) permitem o mapeamento de um determinado nome de domínio num conjunto de endereços IP. É então possível recorrer esta abordagem para que, no contexto de um dado sítio *Web*, o seu servidor DNS mapeie cada nome de domínio num grupo de endereços IP de forma a escolher, a cada pedido, qual o servidor mais adequado para o servidor, com base em factores como o local

de origem do pedido ou a carga dos servidores-réplica. A diferença relativamente à multiplexagem no cliente utilizando o seu sistema de resolução de nomes reside no facto de que, nesta abordagem, a escolha do servidor-réplica ocorre no sítio *Web* (ou na infra-estrutura DNS) enquanto que na abordagem referida, essa escolha ocorre ao nível do cliente.

A principal desvantagem associada a esta abordagem reside no facto de que o *caching* de respostas DNS pode invalidar uma eventual escolha a ser feita junto do sítio *Web* caso o pedido tivesse sido recebido. Por outro lado, a redução do tempo de validade das respostas DNS armazenadas em *cache* pode resultar num *overhead* significativo na comunicação. De notar que, de uma forma geral, o sistema DNS foi desenhado sobretudo para a adição de novas entradas, não para a alteração frequente do mapeamento existente.

2.5.3 Anycasting

O *anycast*[KW01] consiste num método para endereçamento e encaminhamento em que os dados são direccionados para o destino *melhor* ou *mais próximo*. À semelhança do que acontece com o *multicast*, a relação entre endereços na rede e nós correspondentes é de um-para-muitos, isto é, um endereço *anycast* pode referenciar vários nós da rede. Todavia, enquanto que uma abordagem *multicast* determina que os dados são distribuídos por todos os nós referenciados por um determinado endereço, uma abordagem *anycast* define que os dados são entregues a um receptor: o melhor ou o mais próximo de uma perspectiva de topologia da rede.

Os processos de *anycast* propostos podem ser agrupados em *anycast* ao nível do protocolo IP ou ao nível do protocolo aplicacional.

2.5.3.1 IP Anycasting

Os sistemas *IP Anycasting* assumem que o mesmo endereço IP está atribuído a um conjunto de nós da rede, sendo que cada *router IP* tem, na sua tabela de encaminhamento, uma ligação para o nó mais próximo de si, de acordo com a métrica de distância definida. Por conseguinte, no contexto de uma rede, diferentes *routers IP* têm referências para diferentes nós para o mesmo endereço *IP*.

C. Partridge et al.[PMM93] definiram uma primeira abordagem ao *anycasting* de acordo com o qual o encaminhamento é feito com recurso a protocolos de *unicast*. Todavia, devido às diferenças de funcionamento entre ambos os métodos de reencaminhamento de dados, a implementação de *anycast* com recurso a protocolos *unicast* não é escalável. O *unicast* escala ao agregar caminhos para destinos que partilham o mesmo prefixo numa única entrada numa tabela de encaminhamento. O *anycast*, por outro lado, desafia esta forma de agregação hierárquica: um endereço *anycast*, à semelhança de um endereço *multicast*, representa um grupo de nós que partilham uma determinada característica e coexistem em locais arbitrários da Internet. Por conseguinte, não existe nenhuma razão pela qual se deve esperar que a topologia de um grupo *anycast* seja hierárquica ou compatível com uma topologia *unicast*. Posto isto, o encaminhamento *anycast* recorrendo a protocolos de encaminhamento *unicast* requer que cada endereço global associado a um endereço *anycast*

seja especificado separadamente. Este requisito leva a que as tabelas de encaminhamento cresçam proporcionalmente ao número total dos grupos *anycast* em toda Internet, pelo que não escala.

De forma a contornar esta limitação inerente ao *IP anycasting*, D. Katabi et al.[KW01] propuseram uma outra aproximação e este método sob a forma de uma arquitectura escalável, denominada *Global IP-Anycast* (GIA). Devem então ser definidos os seguintes conceitos de forma a descrever, ainda que de forma breve, esta abordagem:

- Um grupo *anycast* interno a um domínio *D* consiste num grupo em que, pelo menos um dos seus membros, é interno a *D*. Note-se que todos os grupos são internos ao seu *home domain* (o domínio da Internet a que pertence o prefixo do seu endereço *anycast*); no entanto, estes podem ser internos a outros domínios distintos.
- Um grupo *anycast* externo a um domínio *D* é um grupo em que nenhum dos seus membros pertence a *D*.
- Um grupo *anycast* popular a um domínio *D* é um grupo externo a *D* frequentemente acedido pelos seus utilizadores.

A escalabilidade desta arquitectura reside no facto de que são capturadas características especiais do serviço *anycast* no seu protocolo de encaminhamento inter-domínio, o que gera dois tipos distintos de caminhos: caminhos por omissão, definidos com base no prefixo *unicast* do *home domain* do endereço, que é parte de um endereço *anycast*, que não consomem largura de banda e não necessitam de espaço de armazenamento nas tabelas de encaminhamento; caminhos mais curtos, personalizados de acordo com os interesses dos beneficiários do domínio, periodicamente pesquisados pelos *routers* de fronteira e armazenados em *caches* para o efeito.

O encaminhamento em grupos *anycast* recorre ao protocolo tradicional de encaminhamento *unicast*: embora, para cada grupo interno ao domínio, o uso do protocolo *unicast* requeira uma entrada extra na tabela interna de encaminhamento, esta abordagem permanece escalável pois o número de grupos interno pode ser limitado pelo próprio domínio em função da largura de banda e espaço de armazenamento disponíveis. Para encaminhar um grupo *anycast* não-popular, são utilizados caminhos por omissão: o número de grupos não-populares tende a ser significativamente superior ao de grupos populares, pelo que o uso de caminhos por omissão promove a poupança de recursos. Por fim, o encaminhamento de grupos populares é feito com recurso à determinação de caminhos mais curtos para os mesmos.

Embora adequado ao encaminhamento de pedidos e localização de serviços, o *IP anycasting* apresenta as seguintes limitações:

- Endereços *anycast* requerem suporte ao nível de encaminhamento, isto é, em *routers*.
- Consistente com o funcionamento do protocolo IP, o destino é determinado por pacote.

- A escolha do destino de um pedido feito está inteiramente a cargo da rede, não havendo espaço para que o cliente tome quaisquer decisões.

2.5.3.2 *Anycasting* ao Nível Aplicacional

Uma vez que a camada de rede consegue determinar de forma eficiente o caminho mais curto entre dois nós no contexto de uma rede, a utilização de *IP Anycasting* é adequada quando a métrica para selecção de um servidor se baseia numa medida de distância como, por exemplo, o número de saltos. Por outro lado, uma abordagem ao nível aplicacional permite o recurso a várias outras métricas como, por exemplo, a capacidade de processamento de pedidos de um servidor. Nesta subsecção serão abordados sistemas típicos de *anycast* ao nível aplicacional.

De forma a contornar as limitações inerentes ao *IP Anycasting*, Bhattacharjee et al.[BAZ+97] propuseram uma arquitectura de um serviço *anycast* ao nível aplicacional. Segundo esta abordagem, um grupo *anycast* ao nível aplicacional é univocamente identificado por um *Anycast Domain Name* (ADN). O serviço consiste então num conjunto de sistemas de resolução de ADN que asseguram o mapeamento de cada ADN nos endereços IP dos membros do grupo *anycast* referenciado. Estes sistemas contêm também uma base de dados de métricas associadas aos servidores: por conseguinte, dado um critério de selecção, o sistema de resolução *anycast* pode escolher qual o melhor servidor para servir um determinado cliente. Todavia, um sistema deste tipo implica a introdução de alterações nos clientes e servidores, o que pode ser proibitivo no contexto de uma rede como a Internet, em que os números de clientes e servidores são extremamente elevados.

W. Adjie-Winoto et al.[ASBL99] apresentaram o desenho e implementação de um sistema de localização de recursos e serviços para redes dinâmicas e móveis de computadores e outros aparelhos, denominado *Intentional Naming System* (INS). De acordo com este sistema, é definida uma rede *overlay* ao nível aplicacional composta por sistemas de resolução de *Intentional Names* (INRs) que trocam descrições de serviços entre si, construindo caches locais com base nos serviços disponibilizados e encaminhando pedidos dos clientes para os serviços adequados. O sistema INS emprega um mecanismo de *late binding*, o qual integra os processos de resolução de nomes e de encaminhamento de pedidos, o que permite que os clientes continuem a comunicar com serviços mesmo que os mapeamentos de nomes para endereços sejam alterados ao longo da sessão. Este sistema oferece dois tipos distintos de serviços: *intentional anycasting*, um serviço que estabelece uma comunicação entre um cliente e o servidor mais adequado a responder a um pedido (que seja referenciado pelo nome indicado pelo cliente) e *intentional multicasting*, utilizado para distribuir dados a todos os nós que satisfaçam um determinado nome. O sistema apresenta, contudo, algumas limitações: a utilização de *late binding* implica que cada pacote deva conter o nome do serviço, o que se torna dispendioso; os nomes do serviço são definidos de acordo com um par atributo / valor que o define, o que facilita a sua implementação mas retira escalabilidade quando comparado com um sistema hierárquico como o URL.

2.6 Redes Peer-to-peer

Uma rede peer-to-peer consiste numa rede *overlay*, composta por um conjunto de intervenientes, pares, equivalentes a nível de funcionalidade, que partilham capacidade de processamento e largura de banda. Embora existam diferentes tipos de redes peer-to-peer de acordo com o seu propósito, este artigo focar-se-á apenas nas redes peer-to-peer de distribuição de conteúdos.

Uma rede *peer-to-peer* de distribuição de conteúdos permite que os pares publiquem, procurem e obtenham conteúdos uns dos outros. Para além destas funcionalidades básicas, algumas destas redes oferecem também segurança, anonimato, justiça, escalabilidade e garantias de *performance*. É possível classificar uma rede *peer-to-peer* de distribuição de conteúdos de acordo com a sua estrutura da seguinte forma:

- *Redes Não-Estruturadas*: a topologia lógica da rede consiste numa malha não estruturada. Por conseguinte, a disposição de conteúdo numa rede é completamente independente da topologia *overlay*. Neste tipo de redes, os conteúdos necessitam tipicamente de ser localizados, pelo que se recorre frequentemente a mecanismos de pesquisa *brute-force*, como *flooding* ou *random walks*. Estas estruturas são especialmente adequadas a redes transientes.
- *Redes Estruturadas*: este tipo de redes surgiu como uma tentativa de endereçar os problemas de escalabilidade que, inicialmente, afectavam as redes não-estruturadas. Neste tipo de redes, a topologia lógica da rede segue uma estrutura bem definida, habitualmente construída com recurso a técnicas de tabelas de dispersão distribuídas (DHT). Nestes sistemas, existe habitualmente um mapeamento entre conteúdos e localizações sob a forma de tabelas de encaminhamento, para que o encaminhamento de pedidos seja mais eficiente.

Nesta secção serão apresentados os princípios subjacentes a cada um destes tipos de redes.

2.6.1 Redes *Peer-to-peer* Não-Estruturadas

Nesta subsecção serão apresentados os principais métodos de pesquisa numa rede *peer-to-peer* não-estruturada, bem como uma análise destas redes segundo uma perspectiva de centralização.

2.6.1.1 Algoritmos de Pesquisa em Redes *Peer-to-peer* Não-Estruturadas

O algoritmo de pesquisa de objectos é um dos componentes chave de uma rede *peer-to-peer*, uma vez que define uma parte significativa da *performance* global. Os algoritmos de pesquisa de objectos podem ser classificados da seguinte forma: aleatórios ou deterministas, recorrendo ou não a *flooding* e centralizados ou descentralizados. À excepção da pesquisa centralizada da rede *Napster*, a generalidade de redes *peer-to-peer* recorre a processos de pesquisa distribuída do tipo salto-a-salto. Nesta subsecção serão abordados os principais tipos de pesquisa descentralizada[LCC02].

- *Flooding*: consiste no envio, por parte de um interveniente da rede, de uma pedido para a obtenção de um determinado objecto a todos os seus vizinhos. Estes, por seu torno, reenviam o pedido para os seus vizinhos e assim sucessivamente. De forma a limitar o número de mensagens enviadas, é utilizado um atributo nas denominado *Time To Live* (TTL), que determina o número de saltos a ser dado por cada mensagem, sendo decrementado a cada salto (a mensagem é descartada quando TTL chega a zero). O principal problema inerente à utilização do *flooding* consiste na determinação de um TTL. Um valor demasiado baixo par este atributo pode não gerar respostas (isto é, pode não ser encontrado qualquer par que contenha o objecto pedido) enquanto que, por outro lado, um valor demasiado elevado resulta numa grande quantidade de mensagens disseminadas pela rede.
- *Anel de Expansão:* Esta abordagem consiste na utilização de *floodings* sucessivos com TTLs iterativamente aumentados. Um nó faz um pedido com um TTL baixo e aguarda para verificar se foi bem sucedido. Caso não o tenha sido, aumenta o TTL na iteração seguida, enviando novamente o pedido. O procedimento é repetido até que o objecto seja encontrado. Apesar de, em determinados casos, serem necessárias várias iterações do anel de expansão, o anel de expansão reduz significativamente o *overhead* de mensagens quando comparado com o *flooding* com um TTL constante. A redução é mais evidente quando a taxa de replicação do objecto procurado é elevada, embora também se verifique nos casos em que é reduzida. No entanto, verifica-se ainda uma duplicação considerável de mensagens inerente ao *flooding*, método que está por trás do algoritmo do anel de expansão.
- *Random Walks*: De acordo com este algoritmo, um pedido gerado por um nó é enviado apenas para um dos seus vizinhos, o qual é escolhido aleatoriamente. O pedido é então propagado desta forma por toda a rede, isto é, a cada salto, é

escolhido aleatoriamente o próximo nó para o qual o pedido será enviado. Desta forma, o *overhead* de mensagens é significativamente reduzido em comparação com o algoritmo de *flooding*. Todavia, o recurso a este algoritmo pode resultar num atraso significativo pois o número de saltos até que seja encontrado o objecto procurado pode ser elevado. Embora o nó a ser escolhido para reencaminhar o pedido seja escolhido aleatoriamente, as existem dois métodos que definem a probabilidade da selecção: *aleatoriedade uniforme* e *aleatoriedade probabilística*. No primeiro caso, todos os vizinhos de um dado nó têm igual probabilidade de serem escolhidos para o envio do pedido. No segundo, a probabilidade é proporcional ao grau de conexão de cada nó (isto é, ao número de nos a que um determinado nó está ligado). Por conseguinte, a segunda opção resultará, à partida, numa maior taxa de *hits* e num percurso do nó de onde foi originado o pedido até ao nó com o objecto pretendido menor. Todavia, o primeiro método oferece uma maior capacidade de balanceamento de carga. Existe ainda uma variante deste algoritmo denominada *k-Random Walks*. Esta abordagem define que, em vez de ser escolhido um nó para encaminhar um pedido, são escolhidos *k*, aleatoriamente. Por conseguinte, este método consiste num meio-termo entre o *flooding* e o *random walks*.

2.6.1.2 Centralização de uma Rede *Peer-to-peer* Não-Estruturada

Embora, na sua forma mais pura, uma rede *peer-to-peer overlay* deva ser descentralizada, na prática nem sempre é verdade. Por conseguinte, são encontrados vários sistemas com diferentes graus de centralização. Estes podem ser agrupados de acordo com três categorias[DNB03]:

- *Redes Descentralizadas Híbridas*: nestes sistemas, existe um servidor central que medeia a interacção entre pares, mantendo informação acerca dos ficheiros na posse dos vários nós. Embora a troca de ficheiros tenha lugar entre os vários pares da rede, cada nó *i* que se junta à rede tem de notificar activamente o servidor sobre os objectos a partilhar, de forma a que os demais nós necessitem apenas de consultar o servidor central acerca dos conteúdos de *i*. Este tipo de estrutura é simples e facilmente implementável. Todavia, estes sistemas definem um ponto único de falha e são considerados inerentemente não escaláveis, dada a existência de um servidor central e das suas limitações a nível de capacidade de processamento e de resposta a pedidos. O exemplo mais flagrante desta rede consistirá no *Napster*, que surgiu como a primeira grande abordagem às redes *peer-to-peer* para distribuição de conteúdos.
- *Redes Puramente Descentralizadas*: todos os nós desempenham exactamente as mesmas tarefas, de acordo com um padrão arquitectural *peer-to-peer* puro. Não existe qualquer coordenação central. A *Gnutella* constitui um exemplo deste tipo de redes: esta recorre à camada IP como camada de serviço e usa um protocolo do nível aplicacional para a comunicação entre os nós. Após juntar-se à rede, um

nó anuncia a sua presença enviando uma mensagem de *ping* aos seus vizinhos, a qual é propagada pela rede. A pesquisa nesta rede segue um mecanismo de *flooding* com um TTL constante. Este mecanismo resulta numa reduzida escalabilidade da rede: cada nó tem acesso apenas a uma sub-rede, determinada pelo TTL das mensagens que envia, pelo que haverá um limite a partir do qual as suas mensagens nunca chegarão.

- *Redes Parcialmente Centralizadas*: a base destas redes é a mesma das redes puramente descentralizadas. Todavia, nestas redes é introduzido o conceito de *supernós*: nós a que é atribuída uma tarefa de serviço que consiste na indexação de ficheiros partilhados por pares locais. De acordo o protocolo *FastTrack*, usado em aplicações populares como o *Kazaa*, os nós são automaticamente eleitos para o papel de *supernós* caso tenham largura de banda e poder de processamento suficientes. Relativamente aos sistemas centralizados, este tipo de redes apresentam um tempo de descoberta mais reduzido, eliminando a existência de pontos únicos de falha (caso um *supernó* falhe, outro nó é automaticamente promovido a esta categoria). Por outro lado, estas redes são também mais eficientes relativamente à pesquisa de informação do que as redes puramente centralizadas, sobretudo à existência de *supernós* que, na tua totalidade, suportam grande parte da carga da rede.

2.6.2 Redes *Peer-to-peer* Estruturadas

De acordo com esta arquitectura, as redes não apresentam um directório central (logo, constituem redes descentralizadas) mas são definidas de acordo com uma topologia controlada. Por conseguinte, a colocação de objectos não é aleatória, mas específica, facilitando a satisfação de pedidos. Devido à existência de um vasto número destas redes com uma quantidade considerável de propriedades que as distinguem, torna-se difícil classificá-las de acordo com um critério único. Por conseguinte, nesta subsecção serão abordadas, de acordo com as suas características, cinco dessas redes: *Plaxton*, *Tapestry*, *Pastry*, *Chord* e *CAN.*

2.6.2.1 Plaxton[PRR03]

Esta rede é optimizada para suportar uma rede *overlay* para localização de objectos com nomes associados e para o encaminhamento de mensagens para esses nós.

Nesta rede, objectos e nós têm identificadores independentes da sua localização e propriedades semânticas, sob a forma de uma sequência aleatória de bits com uma dimensão fixa. O sistema assume que os nomes são uniformemente distribuídos pelo espaço de nomes, o que é conseguido com recurso a algoritmos de dispersão.

A memoria auxiliar do nó em dividida em duas partes: a tabela de vizinhos e a lista de apontadores do nó. A tabela de vizinhos consiste num mapeamento dos seus vizinhos estratificado em níveis, em que cada nível representa um sufixo partilhado até a uma determinada posição dos identificadores dos nós (isto é, o nó N_i referencia N_j no nível q da

sua lista de vizinhos se os sufixos de ambos forem iguais até a um número de bits proporcional a *q*). A lista de apontadores consiste num conjunto de entradas que mapeiam nós da rede a objectos que estes contenham.

Nesta rede, os objectos são organizados em árvores. Quando um determinado servidor S_1 pretende publicar um objecto O_1, envia uma mensagem para o *nó raiz* de O_1 com o mapeamento (O_1, S_1). Esse mapeamento é guardado nas listas de apontadores dos nós ao longo do caminho de S_1 até à raiz da árvore do objecto O_1. Caso nó no caminho armazena apenas o mapeamento de O_1 para o servidor mais próximo de si.

Ao procurar por um determinado objecto O_1, um cliente C_1 envia uma mensagem para a rede para o objecto em questão, a qual é inicialmente encaminhada até à raiz de O_1. Todavia, a cada passo, caso a mensagem encontre um nó que contenha o mapeamento da localização de O_1, esta é imediatamente redireccionada para um servidor que contenha o objecto. Caso não sejam encontrados servidores ao longo do caminho, a mensagem fica progressivamente mais próxima da raiz de O_1, chegando, de certeza, ao seu mapeamento caso a raiz de O_1 não falhe.

Esta rede não suporta a inserção e remoção dinâmica de nós nem lida com falhas de nós, constituindo uma estrutura de dados estática.

2.6.2.2 Tapestry[ZKJ01]

Tapestry é uma infra-estrutura *overlay* desenhada para a localização de objectos e encaminhamento de pedidos no sistema *OceanStore* (um serviço de armazenamento de dados escalável, desenhado para uma utilização em massa por todo o Mundo). Os mecanismos desta infra-estrutura são baseados na arquitectura *Plaxton*, nomeadamente o processo de atribuição de nomes, estrutura, localização e encaminhamento. Todavia, esta rede tem como objectivo oferecer adaptabilidade e tolerância a faltas.

À semelhança do que acontece com o método de estruturação da infra-estrutura *Plaxton*, cada nó tem um mapa de vizinhos, organizado em níveis de encaminhamento. Todavia, neste esquema, cada nó mantém também um apontador inverso para os nós que o referenciam como vizinho. Estes apontadores são utilizados com o único propósito de gerar mapas de vizinhos para nós a serem integrados na rede.

Tal como na rede *Plaxton*, os objectos são organizados em árvores de acordo com os servidores onde estão mapeados. No entanto, a abordagem *Tapestry* permite que cada objecto tenha múltiplas raízes, as quais são obtidas através da concatenação de valores *salt* (ex: 1, 2, 3, etc.) ao identificador do objecto e computação da sua função de dispersão.

O encaminhamento de pedidos é também semelhante ao da rede *Plaxton*. No entanto, quando existem múltiplas réplicas de um dado objecto, cada nó na rede *Plaxton* guarda apenas o mapeamento da réplica mais próxima de si. Por uma questão de flexibilidade semântica, no caso da *Tapestry* todos os mapeamentos são guardados. A inserção e remoção de objectos são idênticas à verificada na *Plaxton* excepto que, neste caso, têm de ser enviadas mensagens às múltiplas raízes do objecto.

A diferença mais substancial relativamente à abordagem *Plaxton* reside no facto de que a *Tapestry* permite a actualização da topologia. Cada nó actualiza periodicamente a sua lista de vizinhos através de funções de refrescamento (recorrendo ao uso de *pings* ao nível de rede) e existe também um algoritmo de detecção de *hot spots* que sugere localizações para novas réplicas que podem melhorar significativamente a latência no acesso aos mesmos.

2.6.2.3 Pastry[RD01]

O protocolo de localização *Pastry* partilha um grande número de semelhanças com a abordagem *Tapestry*. As principais semelhanças residem ao nível do processo de atribuição de nomes, encaminhamento com base em sufixos (e, neste caso, prefixos), algoritmos semelhantes de inserção/remoção de nós e custos semelhantes de armazenamento.

A cada nó é atribuído um identificador, *nodeId*, de *b* bits, dado pela aplicação de uma função de dispersão ao seu endereço IP ou a uma chave pública criptográfica. Assume-se que estes preenchem o espaço de identificadores uniformemente. Cada nó mantém uma tabela de encaminhamento, um conjunto de vizinhos e um conjunto de folhas: esta consiste numa tabela de *logN* linhas (sendo *N* o número total de nós na rede) e *2b-1* entradas em cada linha. A *n*-ésima linha da tabela contém os *nodeIDs* e endereços IP dos *2b-1* nós mais próximos do nó em análise cujos primeiros *n* dígitos do *nodeID* sejam iguais. A tabela de vizinhos contém os *nodeIDs* e endereços IP dos nós vizinhos. O conjunto de folhas contém os *L/2* nós adjacentes com *nodeID* inferior ao do nó em análise e os *L/2* nós adjacentes com *nodeID* superior (em que $L = 2^{b+1}$).

Dada uma mensagem, o nó verifica primeiro se o seu *nodeID* é abrangido pelo seu conjunto de folhas. Se sim, a mensagem é enviada directamente para este, nomeadamente para o nó do conjunto de folhas com *nodeID* mais próximo do identificador do objecto. Se a chave não é coberta pelo conjunto, é utilizada a tabela de encaminhamento para encaminhar a mensagem para um nó que tenha um prefixo em comum com o identificador da mensagem em pelo menos mais um dígito do que o nó em análise.

Esta rede suporta inserção e remoção de dados, mas não explicitamente objectos móveis. A inserção e remoção de nós é também dinamicamente suportada.

2.6.2.4 Chord[SMK+01]

Chord é um protocolo distribuído de procura desenvolvido no MIT. Suposta a rápida localização de data e a inserção e remoção dinâmica de nós.

Cada nó é associado e um identificador de *m* bits, que consiste na aplicação de uma função de dispersão ao seu endereço IP. Todos os possíveis $N = 2^m$ identificadores são dispostos segundo um círculo bidimensional: cada nó é então mapeado numa posição virtual deste círculo de acordo com o seu identificador. A cada objecto está associado um par chave/valor. Dada uma chave *K* de um objecto, o nó responsável por alojar o valor associado a essa chave pode ser determinado a partir da aplicação da função de dispersão a *K*. Cada

chave *K* é então guardada no nó cujo identificador seja igual à aplicação da função de dispersão a *K* ou, caso este não exista, no primeiro nó cujo identificador suceda a esse valor (ou seja, que seja superior a esse valor, à excepção do valor 2^m-1 cujo sucessor é 0).

De forma a tornar o processo de encaminhamento eficiente, cada nó contem apenas parte da informação de mapeamento. Da perspectiva de cada máquina, o círculo virtual está particionado em *log N + 1* segmentos: o próprio nó e segmentos de dimensão 1, 2, 4, ..., *N/2*. Cada nó mantém então uma tabela com *log N* entradas, em que cada uma contém a informação respeitante a cada segmento: as fronteiras e o seu primeiro nó.

Para um pedido por uma determinada chave *K*, a posição virtual é facilmente calculada através do cálculo da sua função de dispersão. Usando a tabela de mapeamento, o nó procura então o primeiro nó do segmento em que se encontra o objecto procurado, encaminhando para este o pedido. Isto significa que, a cada salto, a distância do pedido ao nó alvo decresce metade.

Para elevada disponibilidade, os dados podem ser replicados com recurso a múltiplas funções de dispersão. É também possível replicar um objecto por *r* máquinas que sucedam ao valor da função de dispersão aplicada à sua chave.

De acordo com esta rede, cada máquina pode entrar ou sair a qualquer altura. A falha de um nó é também detectável e recuperável automaticamente caso cada nó mantenha uma *lista de sucessores* no anel *Chord*.

2.6.2.5 Content-Addressable Network (CAN)[RFH+01]

CAN é uma infra-estrutura distribuída, baseada em funções de dispersão, para rápida procura em redes à escala da Internet.

Na *CAN*, cada máquina é endereçada através do seu endereço IP. Cada objecto é identificado por uma chave *K* e é-lhe atribuído um vector de dimensão *d*, com base na aplicação de uma função de dispersão a *K*, que determina a sua posição num espaço *d*-dimensional.

Um espaço virtual *d*-dimensional é particionado em várias zonas *d*-dimensionais de pequenas dimensões. Cada máquina física corresponde a uma zona e armazena os dados mapeados a essa zona. No espaço *d*-dimensional, dois nós são vizinhos se as suas coordenadas se sobrepuserem em *d-1* das *d* dimensões. Cada máquina conhece as zonas e endereços IP dos seus vizinhos.

A localização de uma máquina que contenha um determinado objecto identificado por uma chave *K* é feita do seguinte modo: é calculado o vector *d*-dimensional da chave para determinar a sua posição virtual; posteriormente, partido de uma qualquer máquina física, é passada uma mensagem de *query* até que seja encontrado o endereço IP da máquina alvo. Num espaço *d*-dimensional, cada nó mantém *2d* vizinhos.

CAN suporta também a inserção e remoção dinâmica de máquinas, detectando e recuperando falhas de nós de forma automática.

2.6.3 CDNs Peer-to-peer

Nos últimos anos, as redes *overlay* têm constituído uma alternativa eficiente ao *IP multicasting* para as comunicações de dados de um servidor para múltiplos clientes. Tradicionalmente, o *IP multicasting* tem constituído o principal método para distribuição de conteúdos a um conjunto de receptores de uma forma escalável. Contudo, determinadas considerações, como a escalabilidade, fiabilidade e controlo de congestionamentos, limitaram a aplicação deste método em larga escala. Mesmo que todos estes problemas fossem endereçados, o *IP multicasting* não considera a largura de banda ao construir a sua árvore de distribuição, o que impede que se tire partido desse aspecto.

Nesta secção serão abordadas quatro estratégias para a disseminação eficiente de ficheiros grandes ou para *streaming* de conteúdos multimédia para uma população de clientes no contexto de uma rede.

2.6.3.1 Overcast[GJJ+00]

Apesar de não ser um sistema *peer-to-peer*, o *overcast* consiste num exemplo importante cujo exemplo influenciou as CDN *peer-to-peer*. Esta estratégia oferece um serviço de *multicast*, com uma única fonte, escalável e fiável recorrendo a um protocolo simples para construção de árvores eficientes de distribuição de conteúdo que se adaptam às alterações da rede. A topologia de uma rede *overcast* consiste numa árvore de servidores de conteúdo auto-organizada. De forma a obter um conteúdo específico, um cliente refere-se a URL contendo o endereço da raiz do grupo, juntamente com o nome do recurso. A raiz selecciona o servidor mais adequado da árvore para servir o pedido do cliente, redireccionando o cliente para o mesmo, o que se torna num procedimento transparente para o utilizador final. Os servidores de conteúdo guardam conteúdos que são disseminados através de *multicast* pela árvore.

A árvore de servidores é construída com o propósito de maximizar a largura de banda da raiz da árvore para todos os seus nós. Para o efeito, esta abordagem leva a árvores de distribuição profundas, em que os nós não observam uma largura de banda pior do que estivessem a obter o conteúdo directamente a partir da raiz. Ao juntar-se à rede *overcast*, um nó associa-se directamente à raiz; de seguida, compara a largura de banda caso obtenha o conteúdo directamente da raiz ou de um dos seus filhos. Se não se verificar um decréscimo da largura da banda segundo um dos seus filhos, associa-se a este. Este processo é executado iterativamente até que se chegue ao fim da árvore ou até que a largura de banda experimente um decréscimo. Periodicamente, cada nó verifica se pode melhorar a sua posição na árvore, comparando a largura de banda disponível com a do seu pai, avô e irmãos. Cada nó mantém também uma lista dos seus antecessores de forma a evitar a formação de ciclos. Informação estatística de cada nó é também enviada de forma periódica para a raiz, para que o sistema possa fazer melhores escolhas ao encaminhar clientes. Para tolerância a faltas, a raiz pode ser replica e as suas réplicas podem ser colocadas linearmente no início da árvore.

2.6.3.2 *Scribe*[CDKR02]

Scribe é um sistema para comunicação de grupos ao nível aplicacional construído sobre a rede *Pastry*. Por conseguinte, a rede *Scribe* consiste num conjunto de nós *Pastry*, pelo que a arquitectura é puramente descentralizada. Qualquer nó *Scribe* pode publicar conteúdos, criando um tópico para o efeito, ou um subscritor de um conteúdo, subscrevendo a um determinado tópico. Cada tópico tem um identificador *topidID* único, o qual resulta da aplicação de uma função de dispersão à concatenação do nome do tópico com o identificador do nó responsável pela sua criação. O nó da rede *Scribe* cujo identificador, *nodeID*, for mais próximo do *topicID* opera como o *rendezvous point* (RP) desse tópico. Quando um tópico é criado, um nó *Scribe* solicita à rede *Pastry* que encaminhe uma mensagem de criação para o nó numericamente mais próximo do *topicID*, isto é, o RP. O RP verifica as credenciais da mensagem, adicionando o tópico à sua lista de tópicos.

A rede *Scribe* cria uma árvore *multicast*, com raiz no RP, para disseminar mensagens *multicast* para o tópico. A árvore é formada pela combinação dos caminhos *Pastry* de cada membro do tópico ao RP. Os membros da árvore de um determinado tópico são denominados *fowarders* em relação ao grupo: estes podem ou não ser membros do grupo. Cada *forwarder* mantém uma lista dos seus filhos que fazem parte da árvore. Quando um nó *i* pretende subscrever a um tópico, envia uma mensagem utilizando o mecanismo de encaminhamento *Pastry*, que encaminha a mensagem até ao RP do tópico em questão. O nó *i* passa então a ser um filho da árvore *multicast* do tópico. Todos os nós entre *i* e RP que não sejam *forwarders* do tópico passam a sê-lo.

O mecanismo de subscrição é escalável a um número elevado de tópicos e a um número elevado de subscritores por tópico. A rede *Pastry* assegura também que as árvores *multicast* tendem a ser bem balanceadas e que a carga de encaminhamento é distribuída pelos nós da mesma, pelo que os nós RP não são sobrecarregados.

2.6.3.3 *Bullet*[KRAV03]

D. Kostic, et al. propuseram um algoritmo escalável e distribuído, denominado *Bullet*, que permite que vários nós, espalhados pela Internet, se auto-organizem numa malha *overlay* de elevada largura de banda. A construção deste algoritmo surge do pressuposto de que a informação deve ser distribuída de uma forma disjunta, para vários pontos estratégicos na rede. Receptores individuais são posteriormente responsáveis por localizarem e receberem os dados a partir de múltiplos pontos em paralelo.

Ao invés de procurar definir a estrutura da rede *overlay* como uma árvore, o que consiste numa replicação do *IP multicasting* seguida pela maioria das aproximações de *multicasting* no nível aplicacional, os autores optaram por uma topologia em malha. A razão subjacente a esta escolha prende-se com os limites da largura de banda de uma árvore *overlay*, bem como da dificuldade em optimizá-la e do elevado *overhead* das funções de *probing*.

A abordagem proposta pelo *Bullet* começa com uma árvore *overlay*, a qual pode ser construída de acordo com o protocolo *overcast* ou outros. O servidor de origem do conteúdo selecciona então um subconjunto aleatório de entre os seus filhos, enviando-lhes uma porção disjunta dos dados, a qual consiste num conjunto sequencial de blocos. O mesmo procedimento é repetido ao longo da árvores, com os nós a dividirem a informação que recebem em porções disjuntes, as quais são enviadas separadamente aos seus filhos. É da responsabilidade dos vários nós a localização das partes de informação que lhes faltam junto dos outros nós. A disseminação de dados uniformemente divididos tem como propósito a eliminação da localização do "último objecto", o qual pode estar disponível apenas num reduzido número de nós. Da perspectiva de cada nó, a largura de banda na obtenção de conteúdos é aumentada, uma vez que estes são obtidos a partir de várias fontes; por outro lado, existe também um aumento de fiabilidade, uma vez que a recepção de dados a partir de múltiplas fontes reduz o risco de potenciais danos causados por falhas em nós.

Este sistema recorre ao *TFRC* como protocolo de camada de transporte, de forma garantir uma comunicação entre todos os nós na rede *overlay*, mesmo em situações de congestionamento, de uma forma compatível com o uso de *TCP*. A construção de conjuntos disjuntos de dados está a cargo de uma abordagem concebida para o efeito, denominada *RanSub*.

2.6.3.4 SplitStream[CDK+03]

SplitStream foi desenhado para a transmissão *overlay* de dados de dimensões elevadas, tal como conteúdos multimédia em *streaming*. Uma vez que o *multicast* baseado numa estrutura em árvore impões cargas elevadas aos nós interiores, estes necessitam de apresentar uma forte disponibilidade e conexões de velocidades altas de forma a que sejam atingidos resultados satisfatórios. De forma a lidar com este problema, Castro et al. propuseram a divisão do conteúdo em *stripes* e disseminação de cada *stripe* numa árvore distinta. Estas árvores devem formar uma floresta disjunta nos nós interiores. Esta abordagem é adequada a ambientes *peer-to-peer* cooperativos, em oposição ao *Overcast*, o qual é desenhado para trabalhar num *overlay* pré-estabelecido. Caso seja utilizado um mecanismo de codificação adequado, torna-se possível aos pares descodificarem a informação mesmo quando existe uma *stripe* em falta (eventualmente com a penalidade de uma qualidade de áudio ou vídeo inferior, no pior dos casos). Como dividir os conteúdos e codificá-los depende da aplicação em questão.

Esta rede é construída no topo de uma abordagem a sistemas *multicast* ao nível aplicacional denominada *Scribe*. Esta infra-estrutura descentralizada oferece um serviço *multicast* eficiente de nível aplicacional, sendo escalável para um elevado número de tópicos, fontes de *multicast* e membros por tópico. A construção de árvores é feita com recurso às propriedades de encaminhamento *Pastry*: as mensagens são encaminhadas para nós cujo identificador partilha prefixos progressivamente maiores com o identificador do objecto. No contexto do *Splitstream*, cada tópico é então definido por um conjunto de nós que partilhem um prefixo ou, pelo menos, um dígito desse prefixo. Ao serem utilizados tópicos cujo

identificador difira no dígito mais significativo, garante-se que cada árvore, respeitante a cada tópico, apresenta um conjunto disjunto de nós interiores.

2.6.4 Codificação de Rede em Sistemas Peer-to-Peer

A computação *peer-to-peer* na Internet assume uma relevância incontornável, representando a maior fatia do tráfego gerado globalmente na rede. Ao contrário do que sucede com os sistemas centralizados, numa topologia *peer-to-peer* os participantes não assumem papéis fixos de clientes ou servidores, acumulando ambas as responsabilidades. Um objectivo importante nas redes *peer-to-peer* é a contribuição que cada utilizador dá ao sistema, fornecendo recursos como largura de banda, espaço de armazenamento ou poder de computação. Logo, com a chegada de nós ao sistema e aumento de pedidos sobre o mesmo, a sua capacidade total aumenta também aumenta. Esta facto não se verifica numa arquitectura cliente-servidor com um número fixo de servidores, em que a adição de mais clientes pode significar uma queda de prestação para todos os utilizadores. A natureza distribuída de uma rede *peer-to-peer* também aumenta a robustez do sistema em caso de falhas ao replicar a informação por múltiplos *peers* e, nos sistemas *peer-to-peer* puros, ao permitir que os utilizadores encontrem a informação que procuram sem o recurso a índices centralizados. No último caso, eliminam-se pontos únicos de falha no sistema.

Existem várias aplicações práticas de sistema *peer-to-peer*, entre as quais se destacam, pela sua popularidade, as comunicações em tempo real (em suporte áudio, vídeo, texto ou qualquer combinação dos três) e a partilha de informação (designado de forma comum por *file sharing*).

De entre as soluções de comunicação em tempo real sobre uma rede *peer-to-peer*, o primeiro sistema a conhecer uma forte utilização global terá sido o IRC (*Internet Relay Chat*). Oficialmente nascido em Agosto de 1988, o IRC foi criado por um académico Finlandês de nome Jarkko Oikarien como uma extensão a um BBS que administrava e tinha como objectivo fornecer um serviço de *instant messaging* (troca de mensagens textuais em tempo real). Este sistema não se baseia numa arquitectura *peer-to-peer* pura, mas numa arquitectura *peer-to-peer* centralizada, pois recorre a servidores para efeitos de listagem e pesquisa de utilizadores e canais de comunicação. Pelo ano de 1989, o IRC já havia sido divulgado um pouco por todo o mundo, vindo a conhecer uma forte utilização com o galopante aumento do número de utilizadores da Internet na segunda metade da década de 1990. Hoje a popularidade do sistema decresceu substancialmente com o surgimento de outras alternativas mais evoluídas, que oferecem um leque bastante mais alargado de funcionalidades. Ainda assim, o número médio diário de utilizadores na totalidade dos servidores IRC é estimado em mais de 1,2 milhões[Rac04]. Hoje, 42% dos utilizadores da Internet declaram utilizar *instant messaging* sendo que, num dia típico, 12% dos utilizadores trocam mensagens recorrendo a estes sistemas entre si[SL04]. De entre as várias alternativas para *instant messaging* com maior aceitação global encontram-se o AOL Instant Messenger (com 53 milhões de utilizadores activos de entre mais de 100 milhões), o MSN Messenger (com 27 milhões de utilizadores activos de entre 155 milhões), o Yahoo! Messenger (com 22 milhões de utilizadores activos), o ICQ (com 4 milhões de utilizadores activos), entre

outros[Rei06]. Recentemente surgiram sistemas, denominados VoIP, com o propósito de oferecerem comunicações por voz com qualidade semelhante às tradicionais comunicações telefónicas, mas a baixo custo. O mais popular destes sistemas será o Skype, lançado em Agosto de 2003 pelas mãos de Niklas Zennstrõm e Janus Friis. Ao contrário do que sucede com os típicos sistemas VoIP, o Skype opera num modelo *peer-to-peer*, pelo que o seu directório de utilizadores está inteiramente descentralizado e distribuído por entre os nós da rede, o que significa que o número de utilizadores pode facilmente escalar a grandes dimensões sem a necessidade de uma infra-estrutura centralizada complexa e dispendiosa. Este sistema permite a conversação por voz (aliada ao *instant messaging* e, mais recentemente, a videoconferência) entre dois clientes Skype ou entre um cliente Skype e um telefone convencional. Hoje, o Skype conta com cerca de 240 milhões de utilizadores registados, 29 milhões dos quais activos, sendo-lhe atribuída responsabilidade por 25% de todo o tráfico VoIP na Internet[XY07]. Existem ainda vários outros sistemas VoIP globalmente populares, como o Google Talk, o VoIPBuster, o Gizmo, entre outros.

Relativamente aos sistemas *peer-to-peer* para transferência de ficheiros, estes tiveram a sua génese no ano de 2002 com o surgimento do Napster, como discutido na secção 2.2.2. Tal como sucede com o IRC no campo de *instant messaging*, o Napster recorre a uma arquitectura *peer-to-peer* centralizada. Sucedendo ao Napster, várias alternativas com diferentes arquitecturas foram emergindo. De entre os sistemas *peer-to-peer* centralizados destacam-se o Audiogalaxy (surgido em Maio de 2001 como sucessor do Napster pela mão de Michael Merhej, com funcionalidades adicionais como o envio de ficheiros de música para grupos ou o *instant messaging*, mas extinto em Junho de 2002 após um processo instaurado pela *Recording Industry Association of America* - RIAA) ou o WinMX (criado em 2001 pela Frontcode Technologies, mas encerrado em Setembro de 2005 após ultimato da RIAA). Dados os problemas legais deste tipo de sistemas, surgiu uma segunda geração de sistemas *peer-to-peer* para *file sharing* que primavam por serem descentralizados. De entre estas redes surgiram a FastTrack (criada em 2001 por Niklas Zennstrõm e Janus Friis, que mais tarde criariam o Skype, tendo atingido o estatuto de rede de *file sharing* mais popular do mundo em 2003 com um número estimado de 2,4 milhões de utilizadores concorrentes e servindo de suporte a sistemas populares como o Kazaa, iMesh ou o Grokster, os quais viriam a ser encerrados após processos instaurados pela RIAA), a eDonkey Network (criada em 2000 pela MetaMachine e vindo-se a tornar, em 2004, no meio mais comum de partilha de ficheiros na Internet, com um número concorrente de utilizadores variável entre 2 e 3 milhões e servindo de base a sistemas como o eDonkey2000 ou o Overnet – encerrados em Setembro de 2006 após processo instaurado pela RIAA – e o ainda resistente eMule), ou a Gnutella (a primeira rede totalmente descentralizada, desenvolvida em 2000 por Justin Frankel e Tom Pepper, tenho vindo a conhecer uma crescente popularidade em 2001 com a queda do Napster e servindo de base a aplicações como o LimeWire – actualmente processado pela RIAA mas ainda em funcionamento -, o Morpheus – encerrado após processo instaurado pela RIAA – ou Gnucleus – ainda em funcionamento).

Posteriormente surgiu o BitTorrent, um sistema que representa mais de metade de todo o tráfego *peer-to-peer* na Internet. O BitTorrent diferenciou-se dos restantes sistemas *peer-to-peer*

para *file-sharing* ao oferecer prestações mais eficientes no *download* de ficheiros (ao permitir a obtenção de um ficheiro através de várias fontes em simultâneo, privilegiando as mais rápidas) ao invés de se focar na procura destes ficheiros. Por conseguinte, não é de estranhar que muitos tenham sido os artigos escritos subordinados a estes temas: desde análises[IUB+04][GCX+05][DL07] a sugestões para aumentar a sua eficiência[SMB02][AHM+03][BHP06], passando inclusivamente pela exploração de deficiências e vulnerabilidades nestes sistemas[FC05][LNKZ05][FIA+07].

É neste contexto que surge o tema da Codificação de Rede, um assunto com amplo destaque no campo da investigação relacionada com a disseminação de conteúdos na Internet, uma vez que tornava mais eficiente a utilização do recurso mais dispendioso e escasso neste meio – a largura de banda. Vários foram os artigos escritos subordinados a este assunto: alguns advogando as suas vantagens[CWJ03][FBW06][MXLX06], outros criticando-o[LUM06]. Todavia, até hoje, é conhecida apenas uma implementação funcional de um sistema de distribuição de conteúdos que faça uso de Codificação de Rede: o Avalanche da Microsoft Research[GR05]. Este sistema partilha profundas semelhanças com o modelo de investigação estabelecido neste trabalho: trata-se de um sistema baseado no BitTorrent cujo objectivo é o de optimizar o algoritmo para escalonamento dos pedaços a serem distribuídos. Enquanto que no BitTorrent cada pedaço em particular tem de ser obtido para que se reconstrua o conteúdo original, no Avalanche basta que seja obtido um determinado número de blocos codificados, tornando o sistema mais robusto a saídas, evitando que um qualquer nó específico se torne num *bottleneck* e fazendo um uso mais eficiente da largura de banda ao evitar que a mesma informação percorra múltiplas vezes ligações que constituam *bottlenecks*. Segundo os investigadores da Microsoft Research, este sistema pode ser 20% a 30% mais rápido do que um sistema de codificação na fonte[RJCE06] e entre 200% a 300% mais rápido do que um sistema de distribuição de informação não codificada[GR05].

No entanto, embora este sistema tenha sido apresentado com a promessa de revolucionar a distribuição de conteúdos de grandes dimensões em larga escala na Internet, encontra-se ainda em fase de pesquisa e desenvolvimento desde 2005.

3 ESTUDO ACERCA DA APLICAÇÃO DE UMA CDN CENTRALIZADA

Nesta secção será abordada a primeira fase do projecto: integração de um sistema de Corporate TV, da Mobbit Systems, com uma rede de distribuição de conteúdos, da Cisco Systems. Em particular, na subsecção 2.1 é feita uma contextualização dos sistemas abordados neste estudo, nomeadamente CDNs e soluções de IPTV. Na subsecção 2.2 é feita uma descrição de ambos os sistemas, com especial ênfase nos aspectos mais importantes para a integração dos mesmos. Na subsecção 2.2 é descrito o planeamento desta fase do trabalho.

3.1 Contextualização

As primeiras CDNs emergiram no ano de 1998[RS02] de modo a endereçarem o facto de que a *World Wide Web* não havia sido concebida para lidar com transmissões de conteúdos volumosos sobre distâncias longas[VP03]. Estes novos sistemas, propostos para maximizar a largura de banda, melhorar a acessibilidade a conteúdos e manter a sua correcção de entre as várias réplicas, gozaram de uma forte aceitação comercial: a fiabilidade e escalabilidade ganhas compensavam o *hardware* não dispendioso que estes sistemas implicavam[PV06]. Desde então, várias foram as CDNs que surgiram, assumindo um papel fundamental na distribuição de conteúdos na World Wide Web: a título de exemplo, no ano de 2004 mais de 3000 companhias recorriam a CDNs comerciais, gastando uma verba mensal superior a 20 milhões de dólares no processo[VP03].

Hoje são inúmeras as CDNs existentes. Atendendo à oferta comercial de CDNs, a Akamai Technologies lidera o mercado de forma destacada, detendo uma quota aproximada de 80% e uma estrutura de mais de 12 mil servidores por mais de 1000 redes espalhadas por 62 países[VP06]. Existem ainda outras CDNs comerciais com ampla aceitação a nível global, tais como a Mirror Image Internet, a Inktomi (da Yahoo!), a LimeLight Network, entre várias outras[VP06].

Por outro lado, proliferam também CDNs académicas que, de uma forma sumária, fornecem um serviço livre para a replicação de conteúdos na *World Wide Web*. Exemplos deste tipo de sistemas são a Coral Content Distribution Network, a Codeen ou a Globule[PS06].

Os benefícios que advêm da utilização de CDNs em sistemas de *Corporate TV* como o Mobbit InSight são claros: através de ferramentas como o *caching*, pré-posicionamento de conteúdos e encaminhamento de pedidos é possível optimizar o consumo de recursos de largura de banda na WAN (habitualmente o *bottleneck* ou o recurso mais escasso do sistema). Todavia, os sistemas de IPTV são mais abrangentes, abarcando também a noção de *broadcast* (difusão de conteúdos em tempo real). O *Broadcast IPTV* compreende duas grandes arquitecturas: a gratuita (habitualmente designada por *InternetTV*) e a paga (designada frequentemente por *PayTV*).

Pelo mês de Junho de 2006, o número de canais *InternetTV* disponíveis ultrapassava já os 1300 por todo o Mundo, transmitindo em taxas compreendidas entre os 24Kbps e os 10Mbps. Este sector tem conhecido um forte crescimento e os grandes canais televisivos de todo o Mundo já transmitem as suas emissões através da Internet. Estes canais gratuitos de IPTV requerem apenas uma ligação à Internet e um dispositivo de visualização, como um computador, um HDTV ou até mesmo um telefone móvel equipado com terceira geração. Neste caso, os provedores de serviço não

têm controlo sobre a entrega final, emitindo numa base de "melhor esforço". Consequentemente, não existe garantia de qualidade, podendo haver pausas ou interrupções no envio do conteúdo (uma vez que este é feito através da rede Internet pública). Técnicas para a optimização do tráfego de vídeo na Internet devem ser implementadas para que seja assegurada uma determinada qualidade de serviço, uma necessidade que ganha maior peso quando se pretende transmitir vídeo com qualidade HDTV (com taxas superiores a 10Mbps). Um sistema de *InternetTV* popular para transmissão de HDTV designa-se por *mariposaHD*: implementado em Dezembro de 2005, este serviço foi criado sobre uma plataforma BitTorrent, fazendo uso da largura de banda dos seus próprios clientes para transmissão de conteúdos[Mar].

Relativamente à oferta de canais IPTV pagos, o mercado tem também conhecido um forte crescimento nos últimos anos, tendo sido verificado um aumento do número de assinantes de 1,9 milhões, em 2004, para 3,7 milhões, em 2005. Estima-se ainda que, pelo ano de 2009, o número total de assinantes deste serviço atinja os 36,9 milhões[MRG06]. De notar que este crescimento é excepcionalmente elevado na Ásia sendo que, no caso específico do Japão, o número de clientes cresceu em 89% entre 2005 e 2006. Por outro lado, cerca de 6% dos lares Europeus usufruem já destes serviços, de acordo com a consultora International Data Group[Gru07]. Para aceder a este tipo de serviços é habitualmente necessário um computador pessoal ou uma *set-top box* ligada a um aparelho televisivo. Os conteúdos vídeo são habitualmente comprimidos usando *codecs* MPEG-2 ou MPEG-4 e enviados através de um *stream* de transporte MPEG. Neste caso, os conteúdos são transmitidos através de uma rede de distribuição de conteúdo fechada, assemelhando-se a uma intranet corporativa. Existe então uma garantia de qualidade e fiabilidade do serviço.

Alguns fabricantes de *software* e *hardware* desenvolveram soluções para os provedores de IPTV paga. Um exemplo desse tipo de soluções é o Microsoft Mediaroom[Mic07]. Esta solução consiste numa plataforma de *software* integrada desenvolvida especificamente para a oferta de vídeo com a qualidade da televisão tradicional, bem como para a oferta de novos serviços em redes de banda larga. A plataforma inclui funcionalidades como a troca instantânea de canais (ICC), visualização de vários canais em simultâneo (PAP e PIP), *video on demand* (VOD) ou a gravação de programas no dispositivo do cliente (DVR). Entre os clientes desta plataforma estão Bell Canada, Bell South, SBC Communications, Verizon Communications, British Telecom, Swisscom, Telecom Itália, entre outros. Existem, toda via, muitas outras soluções disponíveis no mercado para provedores de serviços IPTV pagos, como a Cisco IP/TV, Orca Interactive RiGHTv, Envivio MPEG-4/H.264 IP Television Solutions ou Huawei IPTV Solution, para citar as mais populares[Rod06]. Do lado dos provedores de serviço, contam-se várias centenas por todo o mundo, como a Telefonica (empresa Espanhola de telecomunicações que, em Junho de 2005, contava com 45 mil assinantes), a PCCW (empresa de telecomunicações de Hong-Kong que oferecia, em Fevereiro de 2005, o serviço a cerca de 420 mil assinantes), FastWeb (empresa Italiana de telecomunicações que, em Novembro de 2005, tinha um número de assinantes superior a 160 mil), entre outras[Rod06]. Em Portugal, este tipo de serviços é já oferecido pela Clix (lançado em Abril de 2006), que conheceu recentemente a concorrência da Portugal Telecom (que apresentou o seu serviço em Fevereiro de 2007), tal como por empresas como a Sonaecom, a Cabovisão e a ArTelecom[DE07].

3.2 Descrição dos Sistemas em Análise

De forma a abordar os aspectos necessários à integração dos sistemas torna-se necessário compreender como ambos operam. Por conseguinte, esta secção consistirá numa análise de ambos os sistemas, Mobbit InSight e Cisco ACNS, numa descrição dos objectivos que se pretende atingir com a integração dos mesmos e numa análise comparativa das opções de integração consideradas.

3.2.1 Descrição do Sistema *Mobbit InSight*

Partindo de uma perspectiva comercial, o sistema InSight da Mobbit Systems tem como propósito oferecer um serviço de *Corporate TV* a um conjunto de clientes. Um sistema de *Corporate TV* tem como objectivo a difusão de conteúdos multimédia armazenados, dispostos de acordo com grelhas de programação previamente elaboradas, numa rede de ecrãs arbitrariamente definida, com a dispersão geográfica desejada[Mobb]. Este tipo de sistemas tem como habitual finalidade a apresentação de conteúdos propósitos publicitários, informativos ou de entretenimento em estabelecimentos públicos ou privados. Este tipo de sistemas prolifera em várias grandes superfícies comerciais espalhadas um pouco por todo o país, em redes privadas de lojas ou mesmo na rede de metropolitano de Lisboa.

De uma forma mais geral, o sistema Mobbit InSight consiste num sistema de IPTV: um serviço de televisão digital é prestado através do uso do Protocolo de Internet sobre uma infra-estrutura de rede normalmente baseada em ligações com elevada largura de banda. Todavia, o termo IPTV abrange dois tipos distintos de serviços (embora possam ser oferecidos em conjunto): a difusão de conteúdos em tempo real e a difusão de conteúdos armazenados. Este último conceito, no qual se enquadra o sistema Mobbit InSight, agrupa ainda dois subtipos de serviços: a difusão de conteúdos *on-demand* (isto é, solicitados pelo utilizador final) e o *Corporate TV* (ou seja, os conteúdos são definidos de forma centralizada, pelo que o utilizador não escolhe os conteúdos a serem apresentados). É neste último tipo que se insere o sistema Mobbit InSight.

Toda a gestão de conteúdos e organização dos mesmos em grelhas de programação é assegurada de forma centralizada. Por conseguinte, o processamento a ser executado pelos dispositivos associados a cada ecrã da rede de ecrãs (terminal) é reduzido, resumindo-se à obtenção dos conteúdos a serem apresentados junto de um repositório para o efeito, bem como à sua exibição de acordo com as restrições especificadas. Recorrendo aos padrões arquitecturais de *software*, de uma perspectiva componente e conector (isto é, tendo em conta os nós de processamento do sistema e as ligações entre si), é possível afirmar que o sistema se comporta de acordo com um modelo cliente/servidor[CBB+02].

O sistema, cuja concepção esteve integralmente a cargo da Mobbit Systems, apresenta como sua principal vantagem a flexibilidade ao nível dos terminais, que são os elementos do sistema visíveis da parte do cliente. Estes necessitam apenas de executar um cliente do sistema InSight (de momento disponível para os sistemas operativos Linux® e Mac OS®, mas de futuro também para Microsoft Windows®), tendo a si associados um ou mais suportes de visualização (projectores, ecrãs ou televisores) e áudio (colunas de som), *hardware* que processa o conteúdos

a serem difundidos (placas gráficas e de som) e uma ligação à Internet de forma a que sejam obtidos os conteúdos a exibir. Não são apresentadas restrições à ligação à Internet a ser utilizada, sendo inclusivamente possível a utilização de meios de acesso móvel[Mobc]. Todavia, dado o cariz dos conteúdos a serem disseminados, é recomendável a utilização de uma ligação com uma largura de banda que não levante problemas à obtenção atempada dos mesmos.

Nesta subsecção será feita uma análise à arquitectura do sistema segundo uma perspectiva de afectação, isto é, segundo o mapeamento de módulos de *software* do sistema em *hardware*. Serão descritos os principais conceitos essenciais à compreensão do seu funcionamento, será abordada a forma como os conteúdos são disseminados, bem como o tipo de informação que deve ser manipulada e transferida para o correcto funcionamento do sistema (com impacto também na integração com o sistema Cisco ACNS).

3.2.1.1 Arquitectura do Sistema

Na base do desenvolvimento do sistema InSight estiveram um conjunto de requisitos não funcionais como, por exemplo, a interoperabilidade, a flexibilidade e a usabilidade. Por conseguinte, procurou-se desenvolver um sistema que, da perspectiva do cliente (ou seja, relativamente ao cliente a ser executado em cada terminal), pudesse ser executado sobre vários suportes computacionais e sistemas operativos, sendo necessária apenas uma ligação à Internet para obtenção dos conteúdos a serem exibidos.

A conjugação destes requisitos levou à seguinte definição do sistema[Mobc]:

- *Backoffice*, onde é assegurada a gestão de conteúdos e de grelhas de programação para os vários terminais, pertencentes aos diferentes clientes. É também possível consultar informação estatística, em tempo real, respeitante a todo o sistema. Esta gestão é assegurada através de uma interface *web*, acessível a clientes que tenham permissões para configuração dos conteúdos e programações associadas aos seus terminais.
- Base de Dados, que contém toda a informação, armazenada num suporte persistente, de configuração do sistema.
- *Gateway*, para assegurar a comunicação entre o *Backoffice* e os vários terminais do sistema.
- Terminais, para a recepção e reprodução de conteúdos associados a grelhas de programação, as quais definem a ordem de exibição dos mesmos, apresentando também restrições de nível horário e periódico.

Esta arquitectura define então dois canais distintos de comunicação:

1. Entre utilizadores e *Backoffice*, para que estes possam parametrizar todos os dados e conteúdos a veicular nos ecrãs dos seus terminais (por *https* segundo uma interface *web based*).

2. Entre terminais e o *Gateway*, para que os primeiros possam obter os conteúdos e programações a serem exibidas e possam submeter periodicamente informação estatística (sendo a transferência de dados em ambos os sentidos assegurada pelo protocolo *SSH-2*).

É possível ilustrar o modo de funcionamento deste sistema com um exemplo real, referente à rede implementada na rede de estações do Metropolitano de Lisboa. Neste caso específico é possível definir essencialmente dois tipos de locais: o repositório central e a estação de metro.

O repositório central contém o *backoffice*, a base de dados e o *gateway*, bem como um servidor que contém os conteúdos a serem disseminados. Este repositório refere-se a uma localização na entidade responsável pela gestão do sistema em produção no Metropolitano de Lisboa. O *backoffice* pode ser acedido a partir de qualquer ponto remoto, através da Internet, para gestão do sistema.

A estação de metro consiste numa localização remota, composta por um terminal e um ou mais ecrãs. O seu objectivo é a reprodução de conteúdos disseminados pelo sistema.

3.2.1.2 Conceitos Inerentes ao Sistema

De forma a definir o modo como a gestão de emissões é feita no sistema InSight, é fundamental a prévia definição dos seguintes conceitos[Mobc]:

- Canal: um canal consiste numa rede *overlay*, estabelecida entre o servidor central e um determinado conjunto de clientes. Associado a um canal estão selecções de conteúdos, bem como grelhas que determinam a ordem e o horário em que estes são exibidos. Por conseguinte, todos os terminais pertencentes a um determinado canal exibem os mesmos conteúdos, segundo os mesmos esquemas de programação. Embora este tipo de comportamento possa indiciar que o envio de conteúdos é assegurado pelo sistema segundo um protocolo *multicast* do nível aplicacional, o InSight faz uso exclusivamente de *unicast*, isto é, dissemina os conteúdos para cada terminal de forma independente.
- Bloco: um bloco consiste num repositório de conteúdos para emissão nos canais. Cada bloco tem uma determinada capacidade relativa ao número de conteúdos que pode conter a cada momento. Cada conteúdo é associado a uma entrada do bloco durante um período de tempo arbitrário. Presentemente, existem três tipos distintos de blocos definidos no sistema: bloco genérico (qualquer tipo de conteúdos), blocos *pop-up* e blocos *templates* (o significado destes dois últimos será explicado no contexto desta subsecção). O objectivo dos blocos é o de facilitar a criação de programações para os canais: torna-se possível definir que um determinado canal, a um determinado período do dia, passa o conteúdo associado a uma determinada entrada de um bloco.

Quando essa entrada é actualizada, o conteúdo a ser exibido no canal é alterado sem que seja necessário redefinir a sua grelha.

- Alinhamento: um alinhamento consiste numa sequência de entradas de blocos. A sua função consiste somente em definir uma lista de blocos a serem exibidos e a ordem segundo a qual deve ser assegurada essa exibição.
- Grelha: uma grelha de programação define uma sequência de alinhamentos, estando associados a uma determinada validade (datas de início e de fim) e a um período diário para emissão (horas de início e final da emissão).
- *Frame*: a cada canal estão associadas uma ou mais *frames*. Cada *frame* constitui uma área do ecrã a que é associada uma grelha de conteúdos. Esta área tem a si associadas as suas dimensões (em *pixeis*), bem como as coordenadas do ecrã em que deve ser colocada. Por conseguinte, cada canal pode apresentar um conjunto de *frames* em que cada uma destas está associada a diferentes grelhas de programação, exibindo conteúdos de forma virtualmente independente.
- *Playlist*: uma *playlist* consiste na compilação de toda a informação que um terminal necessita de saber para o seu correcto funcionamento. Existe uma *playlist* por canal, a qual é gerada pelo *gateway* com recurso à informação existente acerca dos clientes, blocos, alinhamentos e grelhas definidos. Uma *playlist* define, para cada canal, as *frames* que lhes estão associadas (com a respectiva informação acerca das suas dimensões e coordenadas no ecrã). Para cada *frame*, define a sua grelha de alinhamentos (com o respectivo período de emissão e validade). Para cada grelha, os seus alinhamentos. Por fim, para cada alinhamento, quais os seus conteúdos, referenciados de forma directa (através do nome dos ficheiros) e não relativa (através de entradas dos blocos), pois os terminais não têm conhecimento dos blocos. De notar que, a par com os conteúdos definidos, é também definido o seu tipo (de acordo com o bloco de que foi retirado).
- Conteúdo *Pop-Up*: um conteúdo do tipo *pop-up* consiste num conteúdo cujo propósito é o de ser exibido de forma sobreposta em relação a outros conteúdos que possam estar a ser exibidos ao mesmo tempo, no mesmo canal. Conceptualmente, consiste num conteúdo apresentado numa camada (*layer*) que se sobrepõe às *frames* definidas. Esta funcionalidade torna-se especialmente útil em casos em que um canal tem um padrão de emissão (segundo o qual existem áreas claramente definidas, sob a forma de *frames*), mas existem excepções, ao longo da emissão, a este formato (quando não faz sentido a utilização de *frames*, devendo um conteúdo ocupar a totalidade do ecrã).
- Conteúdo *Template*: um *template*, no contexto do sistema InSight, consiste numa aplicação no formato *Macromedia Flash*, com a particularidade de suportar a apresentação de conteúdos parametrizáveis. Por conseguinte, uma aplicação *Flash*, à qual está tipicamente associada um determinado conteúdo imutável, pode também apresentar outros conteúdos multimédia parametrizáveis como, por exemplo, blocos

de texto, imagem ou vídeo, os quais são definidos como ficheiros externos. Esta funcionalidade permite que uma determinada aplicação *Flash* exiba, por exemplo, notícias actualizadas (sobre um padrão de imagem ou animação imutável, inerente à própria animação), cuja frequência de actualização consiste na frequência da actualização dos ficheiros externos utilizados como parâmetros da aplicação.

3.2.1.3 Gestão e Manipulação de Informação

Toda a informação essencial à gestão da distribuição de conteúdos no sistema, nomeadamente relativa a clientes, terminais, blocos, alinhamentos, grelhas, *playlists* e *templates*, é parametrizada em ficheiros, segundo o formato XML. Nesta secção serão abordadas as estruturas dos diferentes tipos de ficheiros utilizados[Mobd].

O ficheiro **clientes.xml** contém a informação associada aos clientes do sistema e, para cada um dos clientes, os respectivos canais (e *frames* de cada canal), blocos de conteúdos e terminais associados. O ficheiro **terminais.xml** contém toda a informação associada à configuração de cada terminal. Consequentemente, sempre que um terminal for adicionado, editado ou removido do sistema, este ficheiro deverá ser gerado. O ficheiro **blocos.xml** contém a informação relativa às entradas dos vários blocos. O ficheiro **alinhamentos.xml** deve conter a informação associada aos vários alinhamentos, nomeadamente, para cada alinhamento, os seus conteúdos (como referências a entradas de blocos) e os terminais a que se destinam. Por conseguinte, a cada inserção, edição ou remoção de um alinhamento, o ficheiro deve ser novamente gerado. O ficheiro **grelhas.xml** deve conter toda a informação associada à configuração das várias grelhas de programação. Assim sendo, a inserção, edição ou remoção de uma grelha implica a geração deste ficheiro. De forma a que os conteúdos operem adequadamente, estes devem ter acesso à informação sobre quais os conteúdos a exibir, quando exibi-los e como dispô-los no ecrã. Toda essa informação é compilada em ficheiros denominados *playlists*. Por conseguinte, para cada canal, é gerado o ficheiro **playlist_X.xml**, em que X é o identificador alfanumérico do canal, com toda a informação de que este necessita para o seu correcto funcionamento. Como previamente indicado, os ficheiros disseminados pelo sistema InSight podem ser *templates*, que correspondem a aplicações *Flash* parametrizáveis. De forma a que a aplicação tenha informação acerca dos parâmetros a utilizar, bem como os terminais de forma a obtê-los junto do servidor central, existe um ficheiro com essa informação. O ficheiro em questão denomina-se **template_Y.xml**, em que Y é o identificador alfanumérico do *template*.

A estrutura dos diferentes ficheiros XML necessários para o correcto funcionamento do sistema Mobbit InSight pode ser consultados em anexo.

3.2.2 Descrição do Sistema *Cisco ACNS*

O sistema Cisco ACNS surge como uma solução da Cisco Systems com o propósito de eliminar os desafios impostos às organizações associados à eficiente disseminação eficiente de

conteúdos ricos, tais como vídeo e imagem, através da *World Wide Web*. A abordagem sugerida consiste, de uma forma genérica, na colocação de conteúdos tão próxima quanto possível dos utilizadores finais. Para o efeito, são utilizadas estratégias de *demand-pull caching* e pré-posicionamento. Por *demand-pull caching* entende-se o *caching* de elementos após estes terem sido pedidos pelo cliente previamente, isto é, quando é feito um pedido por um conteúdo, este é guardado em *cache*. Pré-posicionamento consiste em popular uma *cache* com conteúdos antes destes serem solicitados, assumindo-se um conhecimento parcial ou total dos conteúdos a serem servidos antes que surjam pedidos para a obtenção dos mesmos[GJJ+00].

O sistema ACNS da Cisco consiste então numa rede *overlay* para distribuição de conteúdos em contextos organizacionais, a qual é composta por um conjunto de servidores-fonte, denominados *Content Engines*, cujo propósito é a colocação de conteúdos nas proximidades dos utilizadores finais. Toda a gestão da distribuição de conteúdos, nomeadamente quais os conteúdos a distribuir, os canais de distribuição e o posicionamento de réplicas nos vários *Content Engines* é realizada de forma centralizada.

Nesta secção será descrito o sistema Cisco ACNS de uma forma breve, focando sobretudo as funcionalidades e aspectos relevantes ao projecto (uma vez que o sistema oferece um vasto conjunto de opções que vão muito para além do âmbito do objectivo pretendido).

3.2.2.1 Arquitectura do Sistema

A rede CDN definida pelo sistema Cisco ACNS 5.5, em uso neste projecto, é essencialmente definida por três componentes distintos a serem descritos nesta subsecção[Cis03].

- *Content Distribution Manager*: o principal papel do CDM consiste na gestão centralizada de todos os dispositivos que constituem a rede Cisco ACNS, bem como dos conteúdos a serem distribuídos com recurso à mesma. É a partir deste ponto que se determina a inserção, edição (de acordo com um variado leque de parâmetros) e remoção de quaisquer dispositivos da rede. A escolha dos conteúdos a serem disseminados, nomeadamente o ponto a partir do qual estes são obtidos e que conteúdos devem ser obtidos, bem como a escolha dos locais para os quais estes devem ser distribuídos, cabe inteiramente ao CDM. Torna-se então possível definir canais (no contexto do sistema ACNS da Cisco) para distribuição de conteúdos, definir as politicas de envio de informação a si associadas (nomeadamente restrições horárias, de largura de banda, de segurança, entre outras), associar dispositivos de rede a estes canais e monitorizar toda a actividade na rede (consultando o estado de todos os nós, taxas de largura de banda consumida, taxas de replicação de objectos, taxas de *upstream/downstream*, entre outros).
- *Content Engine*: o papel de um *Content Engine* na rede Cisco ACNS é o de servir pedidos de clientes para a obtenção de conteúdos. A sua alocação a canais e os conteúdos que este deve armazenar são função da política definidas pelo CDM.

- *Content Router*: um *Content Router* tem como propósito encaminhar os pedidos de clientes, na rede Cisco ACNS, para o *Content Engine* mais próximo que contenha o conteúdo pedido. O protocolo usado por estes *routers* é o WCCP, proprietário da *Cisco Systems*, que opera ao interceptar pedidos de clientes e encaminhando-os transparentemente. Por conseguinte, o utilizador final não necessita de estar ciente da presença do *Content Router*, nem sequer de toda a rede Cisco ACNS.

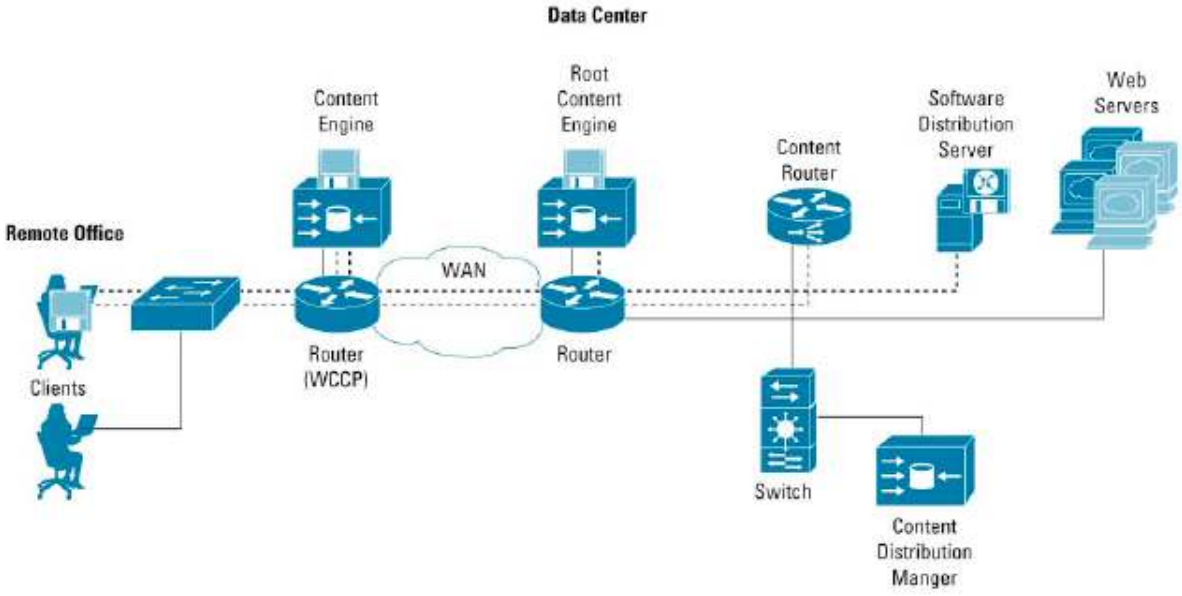


fig. 3: esquematização dos componentes da rede Cisco ACNS[Cis04b].

3.2.2.2 Conceitos Inerentes ao Sistema

De forma a que se entenda o modo como a rede Cisco ACNS opera para a disseminação de conteúdos, nos parâmetros relevantes ao projecto, torna-se necessário ter presente os seguintes conceitos[Cis04b]:

- Local: grupo de *Content Engines* habitualmente encontrados em localizações geográficas próximas e que, na generalidade dos casos, pertencem à mesma LAN. As localizações organizam e agrupam *Content Engines* em redes virtuais para a distribuição de conteúdos através de canais. Estas podem ser configuradas junto do CDM e possuir zero ou mais *Content Engines* destacados para as mesmas.
- Nível: agrupamento hierárquico de localizações definido por relações pai-filho entre estas. Cada localização pode ter zero ou um pai (se tiver zero, considera-se que está na no nível 1, onde se deve encontrar o servidor de onde se obtêm os conteúdos, bem como o CDM) e até 200 filhos. A rede ACNS suporta um total de 5 níveis e 200 *Content Engines* por local.
- Canal: grupo de disseminação de conteúdos para o qual cada *Content Engine* pode ser destacado. Cada *Content Engine* pode ser destacado para servir vários canais simultaneamente. Ao configurar um canal junto do CDM (usando a sua consola ou o

seu interface gráfico), o administrador especifica uma lista de *Content Engines* (que pertencem ao canal e replicam os seus conteúdos) e designa um *Root Content Engine*. Para cada canal, existe apenas uma entidade que publica os conteúdos (*Root Content Engine*), um ou mais receptores desses conteúdos (os restantes *Content Engines* do canal) e, eventualmente, outros *Content Engines* situados entre estes dois últimos (uma situação mais vulgar em redes de maior complexidade ou dispersão geográfica).

- *Root Content Engine*: *Content Engine* destacado para obter os conteúdos, para um determinado canal, junto do seu servidor de origem, disseminando-os pelos restantes *Content Engines* associados ao canal. O *Root Content Engine* deve ser único para cada canal, sendo a localização onde se encontra denominada *Root Location*. Na *Root Location* podem ser apontados mais *Content Engines* para um determinado canal, mas estes apenas funcionarão, como *Root Content Engines* de um canal, caso o *Root Content Engine* original falhe.
- *Edge Content Engine*: *Content Engine* colocado na localização geográfica mais próxima do cliente final de um determinado cliente. Por conseguinte, é o *Edge Content Engine* que serve directamente os pedidos de conteúdos feitos pelos utilizadores finais. De notar que um *Edge Content Engine* pode servir um número arbitrário de utilizadores finais.
- Árvore de Locais: árvore de todos os dispositivos pertencentes a uma rede Cisco ACNS, cuja sua raiz consiste num *Root Content Engine* e as suas folhas em locais, para os quais estão destacados um ou mais *Edge Content Engines*. Caso a árvore tenha mais de dois níveis, os nós intermédios consistem em *Content Engines*. Uma rede ACNS pode consistir em uma ou mais árvores de locais.
- *Manifest File*: ficheiro XML que consiste em objectos, declarados de forma explícita ou com recursos a regras para a obtenção dos mesmos (como a definição da directoria em que estes estão contidos, profundidade de directorias até onde estes devem ser obtidos e restrições a nomes, extensões e dimensões de ficheiros), que devem ser disseminados por um determinado canal. Por conseguinte, é possível atribuir, junto do CDM, um *Manifest File* a cada canal que define a colecção de objectos que este deve distribuir. Com base neste ficheiro, o *Root Content Engine* obtém os conteúdos definidos junto do servidor de origem, distribuindo-os pelos restantes *Content Engines* do canal.

3.2.2.3 Formas de Obtenção de Conteúdos

O conteúdo é o elemento fundamental de qualquer CDN, tal como a Cisco ACNS: este representa toda a informação com que a rede lida. O conteúdo pode ser classificado de acordo com a forma como é adquirido e distribuído. Esta subsecção abordará as diferentes formas de obtenção de conteúdos na rede Cisco ACNS, nomeadamente: conteúdo *on-demand*, conteúdo pré-posicionado e conteúdo *live*.

O conteúdo *on-demand* é adquirido e armazenado pelo *Content Engine* para ser entregue pelo mesmo como resposta a um pedido de um cliente. Quando um cliente solicita, pela primeira vez, um determinado objecto, o *Content Engine* obtém-no junto do servidor de origem de forma a servir o pedido. O conteúdo é também armazenado no *Content Engine*, pelo que pedidos subsequentes para obtenção do mesmo objecto serão directamente servidos por este, ao invés de ser feito um novo pedido ao servidor de origem.

O pré-posicionamento de conteúdo consiste na distribuição de conteúdo para popular *Content Engines* em redes ACNS com gestão central. Por conseguinte, conteúdos que exijam muita largura de banda aquando da sua distribuição podem ser distribuídos durante horários em que o tráfego num determinado troço da rede seja menos intenso. Caso se saiba, de antemão, que determinados conteúdos virão a ser solicitados pelos clientes finais, esta forma de difusão torna-se bastante eficiente, uma vez que, aquando do pedido de um cliente final, o conteúdo já estará armazenado em *Content Engines* próximos de si.

O conteúdo *live* é adquirido através de um *streaming broadcast* a partir de uma determinada fonte. O administrador da rede ACNS configura as políticas associadas à obtenção do *stream*, tal como o seu URL, a sua *bit rate* máxima e políticas de distribuição (tal como prioridades, escalonamentos e limitações de largura de banda).

3.2.2.4 Disseminação de Conteúdos

No contexto do projecto em análise, os clientes finais da rede cisco ACNS serão terminais do sistema Mobbit InSight. Uma vez que estes se destinam a exibir conteúdos multimédia predefinidos, o pré-posicionamento de conteúdos nos *Content Engines* da rede adequa-se. Por conseguinte, será dado particular ênfase a essa funcionalidade do sistema nesta subsecção.

Antes de que a rede ACNS possa adquirir e distribuir um determinado conteúdo, deve ser criado um novo canal, devem ser subscritos *Content Engines* a esse canal e um destes deve ser designado como *Root Content Engine*. O conteúdo a ser adquirido pode estar contido vários servidores de ficheiros ou servidores *Web*. O *Root Content Engine* do canal tem então como propósito obter os conteúdos a distribuir junto desses locais, replicando-os para todos os *Content Engines* do canal. A aquisição do conteúdo pode ser efectuada de acordo com diversos protocolos distintos, nomeadamente HTTP, HTTPS e FTP[Cis04c].

Ao configurar um canal para a aquisição de conteúdo, é necessário especificar qual o método para especificação dos conteúdos a ser utilizado. Este pode consistir na especificação através do interface gráfico do CDM ou através de um ficheiro externo denominado *Manifest File*, o qual deve ser acessível pelo CDM (como, por exemplo, um servidor arbitrário na Internet)[Cis03].

O sistema ACNS permite o pré-posicionamento de conteúdos explicitamente enumerados ou a definição de tarefas de *crawling* sem ser necessária a criação de um *Manifest File*, directamente no interface gráfico do CDM. Uma tarefa de *crawling* consiste na

especificação de um determinado local e de um dado nível de profundidade (a nível de directorias) em que os conteúdos devem ser adquiridos para distribuição pelo sistema. É também possível estabelecer filtros, o que torna este método mais versátil. Todavia, o uso desta forma de especificação de conteúdos é recomendado para casos de demonstração ou casos de utilização simples, em que não sejam necessárias tarefas de aquisição e distribuição avançadas. Isto porque, de forma a alterar os conteúdos a serem distribuídos, é necessária intervenção humana para operar com o interface gráfico do sistema.

O sistema suporta a definição de conteúdos para aquisição e distribuição com base num ficheiro XML de referência denominado *manifest file*. O *manifest file* lista todo o conteúdo a ser usado para popular os *Content Engines* associados a um determinado canal de uma rede Cisco ACNS. Por conseguinte, deve existir um *manifest file* por canal. Este é colocado num determinado servidor de origem, acessível ao CDM e *Root Content Engines*, e identificado pelo seu URL. Após ser validado pelo CDM, este distribui o URL do *manifest file* por todos os *Root Content Engines* que, por sua vez, verificam os conteúdos do mesmo de forma a obterem novo conteúdo que ainda possam não ter. Por conseguinte, depois de cada *Root Content Engine* determinar que conteúdo é novo no *manifest file*, procede à sua obtenção a partir dos locais especificados neste ficheiro. O *manifest file* pode também enumerar os ficheiros a serem obtidos ou definir regras de *crawling*, à semelhança do que sucede com a definição de conteúdos a serem obtidos no interface gráfico do CDM. Por fim, é também possível determinar no CDM a frequência com que deve ser obtido o *manifest file*, de forma a verificar se este foi alterado.

Variadas regras podem ser definidas nos *manifest files*, tais como a dimensão dos ficheiros a serem obtidos, os períodos cronológicos de obtenção dos mesmos, entre outros. Todavia, a sua estrutura básica, para a enumeração de conteúdos é a que se segue[Cis04b]:

```
<?xml version="1.0" encoding="UTF-8"?>
<CdnManifest>
     <item-group>
          <item>
          </item>
      </item-group>
</CdnManifest>
```

3.2.2.5 Formas de Distribuição de Conteúdos

O sistema Cisco ACNS define três formas para a obtenção de conteúdos por parte de clientes finais: intercepção de pedidos por WCCP, *content routing* e encaminhamento por *proxying*. Os dois primeiros métodos requerem a presença de *routers* adequados para fazer o encaminhamento, o que é dispensado pelo terceiro método. Todavia, para além dos métodos apresentados na documentação do sistema, existe ainda um quarto: a obtenção de conteúdos a partir de *Content Engines* recorrendo ao protocolo CIFS. Estes métodos serão descritos nesta subsecção.

No encaminhamento transparente de pedidos por WCCP, os pedidos feitos para um servidor de origem são interceptados por um *router* preparado para o efeito (como, por exemplo, um *Content Router* da Cisco). Este redirecciona transparentemente o pedido para um *Content Engine* que contenha o objecto pedido. Este tipo de reencaminhamento transparente permite a intercepção de qualquer tráfego através de portas que sejam transversais ao *router*. De forma a ser usado um *router* WCCP, o seu endereço IP deve ser adicionado ao *Content Engine*. De notar que este protocolo é para aplicação em servidores *Web*, pelo que só encaminha pedidos HTTP, HTTPS e RTSP[Cis04c].

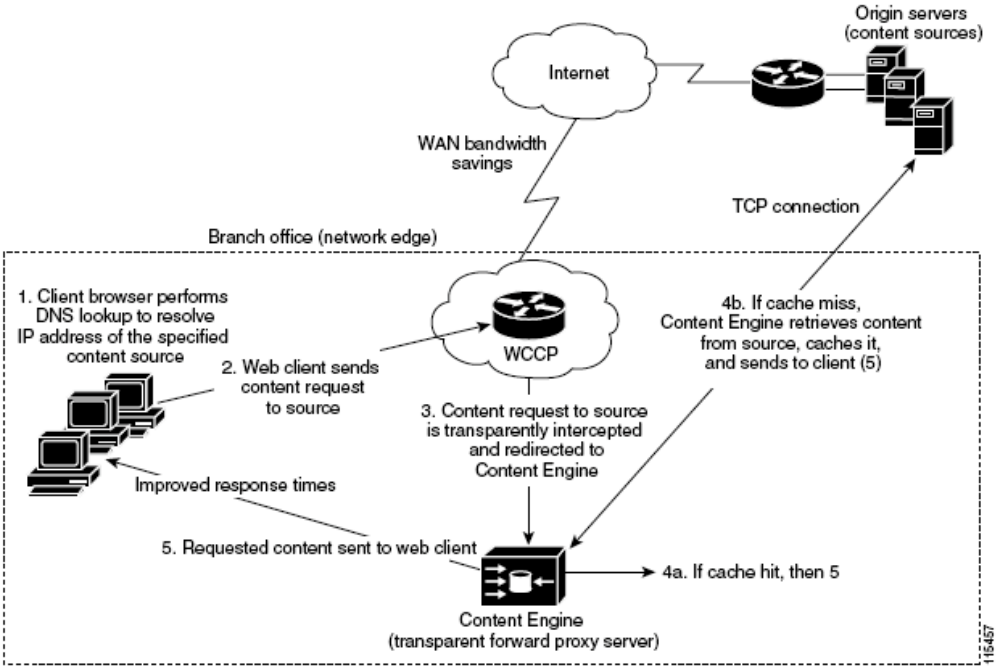


fig. 4: encaminhamento transparente de pedidos usando WCCP[Cis04b].

De acordo com o método *Direct Proxy Routing*, o utilizador final deve estar explicitamente configurado com o endereço IP ou *hostname* de um determinado *Content Engine*, o qual será usado como *proxy*. Por outro lado, não existe a necessidade de ser colocado qualquer *hardware* adicional na rede, tal como *Content Routers*, *routers* com WCCP ou *Layer-4 Switches*. Se o *Content Engine* tiver o conteúdo pedido no seu espaço de armazenamento, este é directamente enviado para o cliente. Caso contrário, o conteúdo é primeiro obtido a partir do servidor original, para ser de seguida servido ao cliente[Cis04c]. O sistema Cisco ACNS suporta também uma funcionalidade denominada PAC, a qual permite que um utilizador

obtenha dinamicamente o endereço IP do *Content Engine* a ser usado como *proxy*, bem como a informação necessária à configuração de portas[Cis04b]. De notar que este método só suporta a utilização de HTTP para efectuar pedidos.

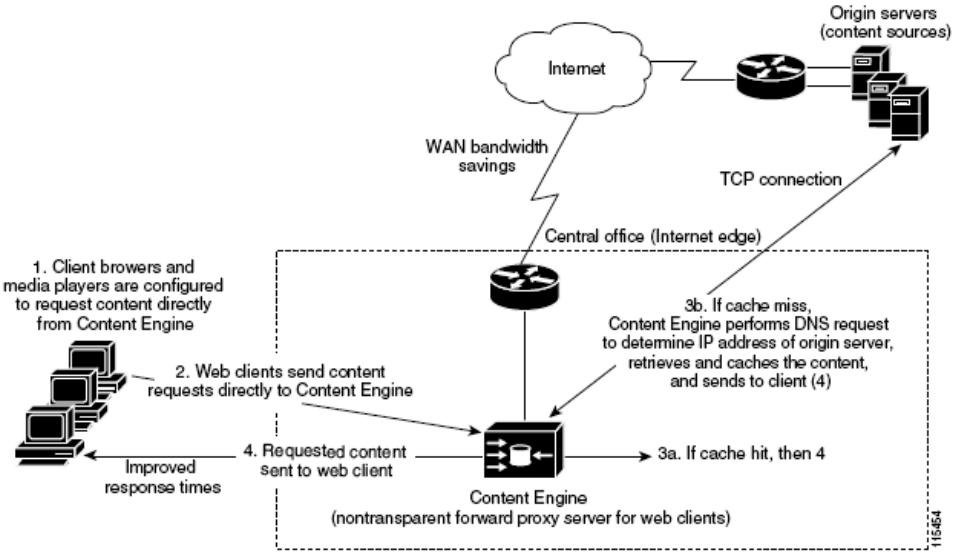


fig. 5: serviço de pedidos usando *direct proxy routing* [Cis04b].

Uma última forma de distribuição de conteúdos com recurso à rede Cisco ACNS, embora não apresentada como tal, é a utilização do protocolo CIFS. De acordo com esta estratégia, o cliente final acede directamente ao espaço de armazenamento do *Edge Content Engine* com recurso a este protocolo, tal como se estivesse a aceder ao seu próprio espaço de armazenamento. Este método é bastante mais simples de utilizar, uma vez que o cliente opera como se os conteúdos fossem sempre locais a si, não necessitando de fazer pedidos através de uma rede: nomeadamente, sempre que necessita de um conteúdo que não tenha no seu espaço de armazenamento, verifica o espaço de armazenamento do *Edge Content Engine*. Para além da simplicidade, este método dispensa o recurso a *hardware* adicional para o encaminhamento de pedidos. Todavia, existe uma grande desvantagem inerente a este método: o funcionamento só é garantidamente correcto caso todos os conteúdos necessários ao cliente estejam já no *Edge Content Engine*. Por conseguinte, é apenas possível utilizar este método caso se saiba, *à priori*, todos os conteúdos necessários ao correcto funcionamento dos utilizadores finais para que os *Content Engines* possam ser antecipadamente populados.

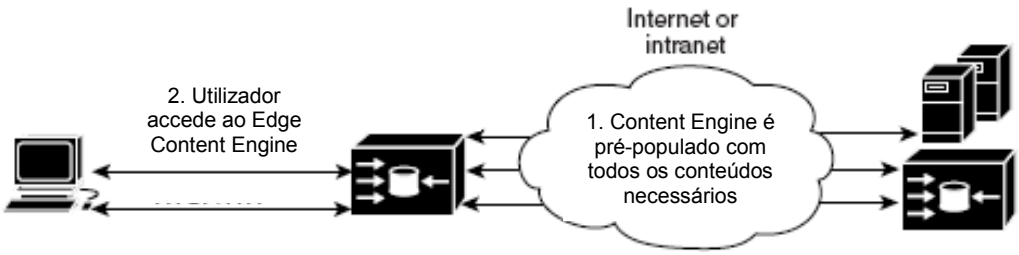


fig. 6: serviço de pedidos usando CIFS[Cisco4b].

3.3 Planeamento

De forma a serem atingidos os objectivos propostos no âmbito do projecto, é necessário fazer uma análise dos requisitos impostos, bem como das várias opções possíveis e dos resultados que destas advêm. Esta secção tem como propósito apresentar o planeamento do projecto.

3.3.1 Abordagens Contempladas para a Integração

A integração da rede Cisco ACNS no sistema Mobbit InSight tem como principal propósito a minimização de largura de banda da WAN consumida: caso não seja utilizada uma solução de *caching* adequada, abre-se a possibilidade de vários terminais num determinado local recorrerem à WAN para a obtenção de um determinado conteúdo, o qual é transmitido em *unicast* para todos os terminais. Esta solução consiste num desperdício de recursos de largura de banda, dado que o mesmo conteúdo será transmitido várias vezes, através da mesma ligação WAN, para os diferentes terminais.

Por outro lado, no caso específico do sistema Mobbit InSight, os conteúdos a serem utilizados pelos clientes finais, isto é, terminais, são inteiramente definidos pelo servidor central do sistema. Por conseguinte, existe total conhecimento, de antemão, de todos os conteúdos que os terminais precisarão de obter para o seu correcto funcionamento.

É possível assumir também que a topologia da rede é estática. Este pressuposto advém do facto de que a solução a que se pretende chegar consiste na optimização dos recursos de largura de banda utilizados para a disseminação de conteúdos a partir de um servidor central, para um conjunto de terminais geograficamente disperso. Todavia, as posições do servidor central e dos terminais são conhecidas à partida, pelo que é possível aplicar soluções com base neste conhecimento.

Levantam-se então três aspectos de relevância capital para a integração de ambos os sistemas:

- Determinar a forma como os conteúdos são disseminados pelo sistema.
- Determinar a forma como os conteúdos são obtidos pelo sistema.

- Definir quais os protocolos de transferência de dados entre *Root Content Engines* e servidor central e entre terminais e *Edge Content Engines*.

Existem essencialmente quatro modos distintos para a replicação de conteúdos no sistema, os quais foram apresentados na secção 2.2.5 deste relatório. Os quatro métodos adequam-se, em termos funcionais, aos objectivos do projecto. A nível de impacto no sistema, o reencaminhamento de pedidos através de WCCP seria o mais adequado: todo o modo de funcionamento do sistema permaneceria transparente para os terminais, que operariam exactamente da mesma forma. As decisões de encaminhamento de pedidos seriam tomadas ao nível dos *routers WCCP*, sem que os terminais necessitassem de ter conhecimento do facto. Todavia, esta abordagem introduz um *overhead* especialmente relevante tendo em consideração que o sistema InSight se trata de um produto comercial: nasce a necessidade se serem colocados *routers* específicos, o que incorre em gastos monetários superiores. Por conseguinte, esta hipótese foi afastada numa primeira análise. Restam então as possibilidades de distribuição de conteúdos através de *Direct Proxy Routing* ou fazendo um *mount* do *Edge Content Engine* e usando o protocolo SMB. Uma vez que os conteúdos a serem disseminados para o sistema são inteiramente conhecidos antecipadamente e dado que nenhuma das abordagens é transparente ao cliente, ambas as opções partem em pé de igualdade. No entanto, existe uma limitação relativamente ao método de *Direct Proxying*, inerente à versão 5.5 do sistema ACNS (a que está a ser usada no projecto): neste caso, só é feito *proxying* dos conteúdos que tenham sido originalmente obtidos por HTTP pelo *Root Content Engine*. Uma vez que foi imposto pelos responsáveis da Mobbit Systems que essa transferência de dados pudesse ser feita de acordo com o protocolo HTTPS, caso necessário (como medida para garantir um maior nível de segurança, isto é, o acesso aos dados mediante autenticação e a cifra dos mesmos para passagem na rede), esta abordagem foi assim descartada. Como consequência, o método mais indicado consiste na distribuição por CIFS, dado que esta abordagem satisfaz os requisitos pedidos, embora retire flexibilidade ao sistema (é fundamental que os conteúdos se encontrem distribuídos pela rede quando os terminais necessitarem de os obter).

Estando o pré-posicionamento de conteúdos na base de todo o funcionamento do sistema, torna-se indispensável definir exactamente quais os objectos necessários para o correcto funcionamento de todos os canais. Para o efeito, a única abordagem que permite a definição de conteúdos de forma automática consiste na geração de *manifest files*. Cada *manifest file* deve listar, para cada canal, exactamente os conteúdos que devem ser distribuídos. Por conseguinte, a solução reside em construir um *manifest file* para cada canal, a qual deve conter todos os conteúdos listados na sua *playlist* e nos *templates* que esta referenciar. De modo a que os *manifest files* sejam consistentes com os conteúdos necessários a cada canal, torna-se necessária a geração periódica destes ficheiros.

Dadas as opções feitas, os protocolos para a transferência de dados entre dispositivos da rede Cisco ACNS e dispositivos do sistema Mobbit InSight consiste na seguinte: entre os *Edge Content Engines* e os terminais, a transferência é assegurada por SMB; entre o servidor onde estão armazenados os conteúdos a serem distribuídos e o *Root Content Engine*, a transferência de dados pode ser assegurada por HTTP, HTTPS e FTP, tendo a escolha dos responsáveis da

empresa recaído sobre o HTTPS devido a eventuais questões de segurança relativas ao acesso a dados e cifra da mesma aquando da sua transmissão.

3.3.2 Solução Proposta

Após uma análise cuidada de ambos os sistemas, bem como dos possíveis pontos de ligação entre ambos, existe já informação necessária para formular uma solução que, simultaneamente, satisfaça os objectivos estabelecidos, incorrendo no menor impacto – técnico e económico – possível, uma vez que se está perante um sistema comercial, já em produção.

Apresenta-se então a solução proposta:

1. À rede do sistema InSight, composta por um servidor central e um número arbitrário de terminais, juntar-se-ão os seguintes componentes do sistema Cisco ACNS:
   - Um *Content Distribution Manager*. Este componente tem como objectivo fazer a gestão de rede Cisco ACNS, indicando quais os conteúdos a serem replicados, o instante da replicação e os canais a serem utilizados.
   - Um *Root Content Engine*. Este componente tem como propósito obter os conteúdos a serem disseminados pela rede Cisco ACNS (neste caso, obtê-los-á junto do servidor central do sistema Mobbit InSight) e distribuí-los pelos *Edge Content Engine*, seguindo as indicações do *CDM*.
   - Um conjunto de *Edge Content Engines*. Estes dispositivos deverão estar localizados junto do utilizador final (neste caso, junto dos terminais do sistema Mobbit InSight). O número de *Edge Content Engines* dependerá da dispersão geográfica dos terminais do sistema Mobbit InSight, bem como da sua quantidade. Cada *Edge Content Engine* poderá estar associado a um número arbitrário de terminais.
   - Um conjunto de *Content Engines* intermédios. Estes dispositivos devem ser entendidos como nós intermédios numa arquitectura em árvore, cuja raiz é o *Root Content Engine* e as folhas os *Edge Content Engines*. A sua presença numa rede Cisco ACNS não é, por conseguinte, obrigatória, sendo úteis em sistemas com uma complexidade a nível de número de nós e localização dos mesmos elevada.
2. Deve ser definido um canal na rede Cisco ACNS para cada canal do sistema Mobbit InSight. Em ambos os contextos, a denominação *canal* tem um significado semelhante: uma rede virtual, composta por uma ou mais fontes e um ou mais receptores, que se destina a disseminar conteúdos, disjunta de outros canais. Por conseguinte, no contexto da rede Cisco ACNS, um canal será formado por um *Root Content Engine* (que contém os conteúdos a serem distribuídos, obtidos através do servidor central do sistema InSight) e por um número arbitrário de *Edge Content Engines*. Cada terminal do sistema InSight deverá então obter os conteúdos junto do *Edge Content Engine* mais próximo de si. Uma vez que a topologia da rede é estática, é possível destacar qual o *Edge Content Engine* responsável por cada terminal, a partir das suas localizações.

3. O servidor central do sistema InSight deverá disponibilizar os conteúdos a serem disseminados num local acessível pelo *Root Content Engine*. A solução mais adequada consiste em disponibilizar os conteúdos que se destinam a ser obtidos pelos terminais por HTTPS (ou, alternativamente, HTTP ou FTP) - para o efeito, deverá ser estabelecido um *link* desde */var/www/html* para a directoria onde se encontram os conteúdos (com as permissões devidamente estipuladas).
4. O *Content Distribution Manager* deve definir, para cada canal da rede Cisco ACNS, quais os conteúdos a serem disseminados. Por conseguinte, deverá existir, para cada canal, um *Manifest File* que contenha todos os ficheiros de que os terminais do sistema Mobbit InSight necessitam para a sua correcta operação, nomeadamente:
   - *Playlist* – ficheiro que define o alinhamento de conteúdos a serem exibidos, bem como a sua ordem e periodicidade.
   - *Conteúdos* – ficheiros vídeo a serem exibidos pelos terminais.
   - *Templates* (e seus descritores) – ficheiros *Adobe Flash* que se destinam a apresentar conteúdos dinâmicos (como notícias e outros ficheiros vídeos), sendo descritos por ficheiros XML que devem também ser disseminados.
   - *Conteúdos referenciados pelos Templates* – conteúdos que devem ser exibidos e são referenciados pelos descritores dos *Templates*. De notar que estes ficheiros são necessários aos terminais e não são referidos nas suas *Playlists*.
5. De cada vez que existe uma alteração numa *playlist* (ou seja, uma alteração na grelha de conteúdos que os terminais devem respeitar), deve ser gerado um novo *Manifest File* para cada canal. O *CDM* recebe então um novo *Manifest File* aquando de cada alteração das *playlists*, dando as instruções adequadas aos *CEs* para a disseminação dos conteúdos acertados.
   - A solução proposta passa pela definição de um processo que está em constante execução, em *background*, no servidor central do sistema Mobbit InSight. Este processo deve verificar periodicamente as *playlists* dos vários canais: caso exista alguma alteração, deve gerar o *Manifest File* respectivo de forma a reflectir a nova grelha de programação, logo os novos conteúdos a serem disseminados.
   - Este aplicação deve fazer o *parsing* das *playlists* de cada canal, extraindo os conteúdos especificados nas mesmas e, com base nessa informação, construindo o *Manifest File* respectivo. Deve ser feito também o *parsing* dos descritores de *templates* referenciados por cada *playlist*. Por último, deve existir o cuidado de referir a fonte exacta de cada conteúdo, para que estes possam ser obtidos pelo *Root Content Engine.*
6. A topologia da rede, no que toca a terminais e *backoffice*, será estática, isto é, não se espera que as suas localizações sejam alteradas. Consequentemente, a rede *overlay* Cisco ACNS também o poderá ser e, adicionalmente, cada terminal sabe também de antemão qual o *Edge Content Engine* junto do qual deve obter os conteúdos de que necessita para o seu correcto funcionamento. Para o efeito, os terminais obtêm os conteúdos junto dos *ECEs* de forma não transparente, recorrendo ao protocolo de

partilha de ficheiros CIFS. Cada terminal faz então um *mount* do espaço de armazenamento do *Content Engine*, recorrendo ao protocolo CIFS.

7. Como acontecia no sistema Mobbit InSight, antes da integração do sistema Cisco ACNS, cada terminal deve consultar periodicamente a sua *playlist* (agora contida no *Edge Content Engine* e não no servidor central InSight). Ao serem adicionados novos conteúdos à *playlist*, estes devem ser copiados do *Edge Content Engine*.
8. O fluxo *upstream* de dados é feito de forma independente da rede Cisco ACNS. A rede ACNS tem como propósito a optimização da distribuição de conteúdos multimédia por vários terminais - conteúdos tipicamente repetidos por entre os vários terminais e com dimensões na ordem das dezenas de *Megabytes*; por conseguinte, o envio de conteúdos estatísticos *upstream* não seria beneficiado caso fosse feito com recurso à rede ACNS.

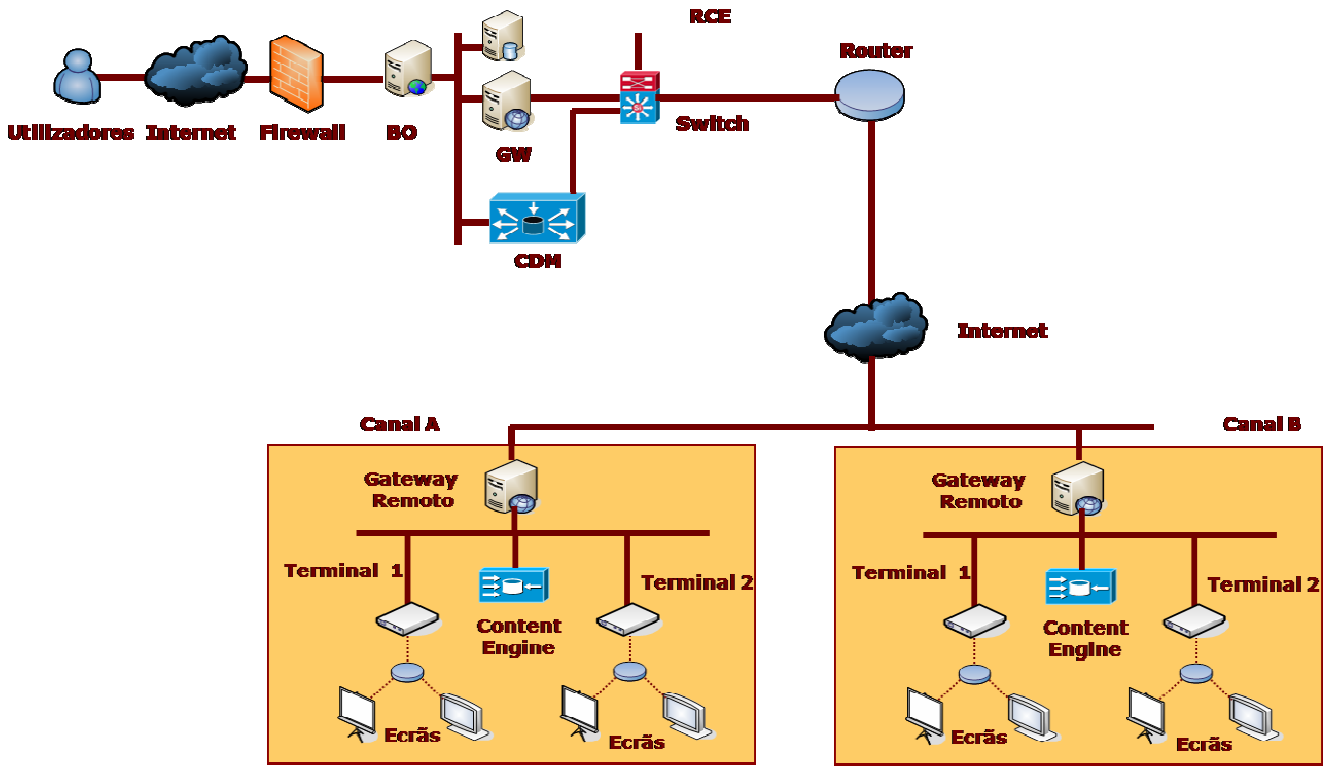


fig. 7: esquematização da solução proposta para a integração dos sistemas.

4 ESTUDO DA APLICAÇÃO DE CODIFICAÇÃO DE REDE A CDNS *PEER-TO-PEER*

Como mencionado na introdução, na experiência anterior foi verficado que a existência de uma topologia estática e pré-definida simplifica radicalmente o desenho de uma CDN, sendo mais simples garantir e determinar a sua eficiência. Por conseguinte, neste capítulo procurou estudar-se o funcionamento das CDNs em sistemas mais dinâmicos, nomeadamente num contexto *peer-to-peer.* Como tal, nesta secção é abordada aplicação de codificação de rede a redes *peer-to-peer* de distribuição de conteúdos. Esta secção consiste em cinco subsecções: descrição do sistema BitTorrent (3.1), descrição dos métodos de Codificação de Rede (3.2), descrição dos simuladores utilizados (3.3), apresentação dos resultados experimentais obtidos (3.4) e discussão de aspectos relevantes relacionados com os resultados obtidos e ilações retiradas (3.5).

4.1 BitTorrent

O paradigma *peer-to-peer* provou ser, nos últimos anos, uma abordagem promissora para o problema da distribuição de conteúdos de elevadas dimensões, a partir de um dado servidor, de forma escalável, a uma grande população de utilizadores. Uma vez que cada utilizador não se comporta somente como um consumidor do conteúdo, mas também como servidor perante os demais utilizadores, a capacidade do sistema servir um conteúdo cresce proporcionalmente com o número de nós, tornando o sistema potencialmente auto-escalável.

O BitTorrent emergiu como uma rede *peer-to-peer* para distribuição de conteúdos, tirando partido da largura de banda de *upload* dos nós participantes para assegurar uma rápida e eficiente distribuição de grandes ficheiros.

Uma das especificidades deste algoritmo consiste na definição de redes independentes para a disseminação de cada conteúdo (*swarms*)[IUB+04]. Cada conjunto de utilizadores que partilham ou pretendem obter um dado conteúdo é agrupado num *swarm*, podendo cada utilizador pertencer, de forma independente, a um número arbitrário de *swarms*.

No BitTorrent, um conteúdo é dividido num grande número de fragmentos, pelo que um utilizador pode começar a servir o conteúdo a partir do momento em que obteve o seu primeiro fragmento. De forma a maximizar a sua utilidade perante os restantes utilizadores, cada nó preocupa-se constantemente em obter os fragmentos mais raros na sua vizinhança. Estas estratégias permitem que o BitTorrent utilize a largura de banda entre os vários nós de forma eficiente, lidando de forma adequada com o fenómeno de *flash crowds*.

Adicionalmente, o sistema emprega uma estratégia que visa maximizar os recursos de largura de banda de uma rede para a distribuição de conteúdo, ao incentivar cada utilizador a usar o máximo possível da sua capacidade de *upload*. Esta estratégia, denominada *tit-for-tat* (TFT)[Coh03], determina que cada utilizador tem mais propensão a enviar blocos de dados a nós junto dos quais tem uma taxa mais elevada de *download*. Este mecanismo é especialmente importante uma vez que existem estudos que determinaram que, em grande parte dos sistemas *peer-to-peer*, uma parte muito significativa dos seus utilizadores são *free riders*[FC05] (ou seja, apenas obtêm dados a partir da rede,

nunca os servindo).

Estas propriedades estão na base do forte sucesso do BitTorrent a nível global, que é responsável por uma grande e crescente fatia do tráfego *peer-to-peer* na Internet.

4.1.1 Terminologia

Os componentes do sistema BitTorrent e respectivas terminologias não estão padronizados pelo que se torna importante a sua apresentação[LUM05]:

- *Swarm*: dá-se o nome de *swarm* a um grupo de utilizadores que simultaneamente enviam ou recebem o mesmo conteúdo.
- *Pedaços e Blocos:* um determinado conteúdo a ser disseminado é dividido em fragmentos de dimensão fixa (habitualmente de 256 KB). Estes fragmentos são conhecidos por *pieces* (pedaços). Os pedaços, por seu turno, são divididos ainda em fragmentos mais pequenos (de 16 KB): os blocos. O bloco é então o grão de informação a ser transmitido numa rede BitTorrent, apesar de o protocolo apenas lidar com pedaços transmitidos. Com efeito, um utilizador apenas pode pedir e servir um pedaço completo, isto é, todos os pedaços que compõem um determinado bloco.
- *Interessados e Sufocados:* diz-se que um utilizador *A* está *interessado* num utilizador *B* quando *B* tem um ou mais pedaços que *A* não tem. Em oposição, um utilizador *A* está *não interessado* num utilizador *B* quando este último tem apenas um subconjunto dos pedaços na posse de *A*. Diz-se que um utilizador *A sufoca* (do inglês, *choke*) um utilizador *B* quando *A* decide não enviar dados a *B*. Em oposição, um utilizador *A des-sufoca* (do inglês, *unchoke*) um utilizador *B* quando *A* decide enviar dados a *B*.
- *Peer Set*: cada utilizador mantém uma lista de outros *utilizadores* que conhece. A este conjunto chama-se o *peer set* de um dado nó. Cada utilizador apenas envia e recebe dados para nós no seu *peer set*. As relações no *peer set* são simétricas, isto é, se um nó A tem o nó B no seu *peer set*, B também tem A no seu *peer set*.
- *Peer Set Activo:* um utilizador apenas pode enviar dados a um subconjunto do seu *peer set*: o subconjunto dos nós que se encontram *não sufocados* pelo utilizador. A este subconjunto dá-se o nome de *Peer Set* Activo.
- *Leecher e Seed:* um utilizador tem dois estados possíveis: o estado *leecher*, enquanto se encontra a obter um conteúdo, mas não tem ainda todos os seus pedaços; o estado *seed*, quando já tem todos os pedaços do conteúdo e apenas os está a servir aos demais utilizadores.
- *Seed Inicial:* a *seed* inicial é o utilizador que é a primeira fonte do conteúdo.
- *Algoritmo Rarest First:* o algoritmo *Rarest First* é a estratégia de selecção de pedaços usada pelo BitTorrent. Segundo este algoritmo, cada utilizador procura obter sempre os pedaços mais raros de entre o seu *peer set*.
- *Algoritmo Choke:* o algoritmo *choke* é a estratégia de selecção de utilizadores utilizada pelo BitTorrent. Segundo este algoritmo, cada utilizador *sufoca* os utilizadores aos

quais não pretende enviar conteúdos. Este algoritmo é também conhecido por *tit-for-tat* (TFT).

- *Tracker:* servidor responsável por manter uma listagem de todos os utilizadores associados que participam na distribuição de um dado conteúdo. Um utilizador junta-se a um *swarm* ao registar-se junto do *tracker*, solicitando-lhe uma lista de outros *peers* para contactar de modo a construir o seu *peer set*. Periodicamente, cada *peer* contacta o *tracker* fornecendo-lhe informação estatística acerca do seu desempenho (estado e quantidades de dados enviadas e recebidas).
- *Torrent:* a cada *swarm* está associado um ficheiro *.torrent*. Este contém um conjunto de metadados relevantes para a disseminação do conteúdo referente ao *swarm* em questão, nomeadamente:
  - URL (endereço IP) do *tracker* associado ao *swarm*.
  - Dimensão dos pedaços do conteúdo.
  - Resultado da aplicação de uma função de dispersão SHA-1 a cada pedaço (de forma a que cada utilizador possa confirmar a recepção em condições de cada pedaço).
  - Lista de ficheiros e directórios a serem partilhados no âmbito deste conteúdo.

4.1.2 Visão Geral do BitTorrent

A participação de um utilizador num determinado *swarm*, no que diz respeito às suas interacções com o sistema e com os restantes utilizadores, processa-se do seguinte modo:

1. O utilizador junta-se a um *swarm* existente ao obter um *.torrent*, habitualmente a partir de um servidor *Web*, que contém os metadados necessários para obter o conteúdo pretendido.
2. Ao juntar-se a um *swarm*, o novo utilizador pede ao *tracker* uma lista de endereços IP de outros nós de forma a construir o seu *peer set* inicial. Esta lista consiste tipicamente em 50 utilizadores escolhidos de forma aleatória de entre todos os nós envolvidos no *swarm*. O *peer set* inicial será posteriormente aumentado por outros nós que se liguem directamente ao utilizador em questão. Estes nós tomarão conhecimento do utilizador em análise ao obterem o seu endereço IP junto do *tracker*.
3. Cada utilizador deve comunicar periodicamente (a cada 30 minutos) o seu estado ao *tracker*, devendo também fazê-lo quando se desliga do *swarm*. Estes mensagens periódicas devem conter a quantidade de *bytes* enviadas e obtidas desde que o utilizador está a participar no *swarm*.
4. Existe um minorante para o *peer set* de cada utilizador, o qual se cifra tipicamente em 20. Assim que o número de elementos do *peer set* de um utilizador fica abaixo desse minorante, o utilizador deve voltar a contactar o *tracker* de forma a obter uma nova lista de utilizadores. Por outro lado, a dimensão do *peer set* é também majorada, habitualmente por 80 elementos. Adicionalmente, dado um utilizador *A*, apenas 40 dos

elementos do *peer set* de *A* podem estar presentes porque *A* os contactou directamente. Os restantes devem ter tido eles próprios a iniciativa de contactar *A*.

5. Cada utilizador tem conhecimento da distribuição de pedaços para cada nó no seu *peer set*. A consistência desta informação é garantida pela troca regular de mensagens. A troca de dados entre os utilizadores é gerida por dois algoritmos nucleares: algoritmo *rarest first* e algoritmo de *choke*. Estes serão descritos em maior detalhe na secção 3.3.3.

4.1.3 Estratégias do BitTorrent para Escolha de Pedaços e Utilizadores

Existem essencialmente duas estratégias responsáveis pelo desempenho do sistema BitTorrent[Coh03]: os algoritmos de escolha de pedaços a obter (algoritmo *rarest first*) e de maximização da largura de banda na rede, o que surge como consequência da maximização da taxa de *download* de cada utilizador (algoritmo de *choke*). Esta secção tem como objectivo expor quais os fundamentos que lhes estão subjacentes.

4.1.3.1 Escolha de Pedaços

A determinação da ordem pela qual são adquiridos os pedaços que compõem um determinado conteúdo a ser disseminado numa rede BitTorrent é de extrema importância para que seja conseguida uma boa prestação. Um algoritmo fraco de escolha de pedaços pode conduzir frequentemente a um problema denominado *último pedaço*: de acordo com este problema, os pedaços que constituem um determinado ficheiro podem ser distribuídos de forma assimétrica, o que resulta na existência de blocos mais comuns e de blocos raros; com efeito, caso os utilizadores que detenham blocos raros decidam sair da rede, passam a existir blocos indisponíveis, o que leva a que nenhum utilizador consiga obter o conteúdo na sua totalidade. Por conseguinte, caso seja utilizado um algoritmo fraco para escolha de pedaços, um utilizador pode ver-se numa situação em que já possui todos os blocos disponíveis nos restantes nós ou em que os pedaços que possui não são de interesse aos mesmos. Esta situação agrava-se quando existem pedaços indisponíveis e, consequentemente, os *peers* deixam de conseguir terminar o *download*.

4.1.3.2.1 Algoritmo Rarest First

O algoritmo *Rarest First* funciona do seguinte modo. Cada nó mantém uma lista que descreve o número de cópias de cada pedaço disponível de entre os nós do seu *peer set*. Esta informação é utilizada de forma a definir o conjunto de pedaços mais raros. Seja *m* o número de cópias do pedaço mais raro no *peer set* de um dado nó: todos os restantes pedaços cujo número de cópias seja igual a *m* são adicionados ao conjunto de pedaços mais raros do nó. Este conjunto é actualizado de cada vez que uma cópia de um pedaço

é adicionada ou removida do seu *peer set*. De acordo com este algoritmo, cada nó selecciona o próximo pedaço que irá obter de forma aleatória, de entre o seu conjunto de pedaços mais raros. Este comportamento tem como objectivo procurar atingir uma distribuição tão homogénea quanto possível de todos os blocos na rede de forma a eliminar o problema do *último pedaço*. Embora cada nó tenha apenas uma visão parcial da rede, correspondente ao seu *peer set*, este consegue atingir uma prestação global satisfatória[LKM06].

O algoritmo descrito, que define o comportamento de base para a definição dos blocos a obter numa rede BitTorrent, conhece três excepções ao seu normal funcionamento, as quais são adoptadas em situações específicas: *strict priority, random first piece* e *endgame mode*[EP05].

4.1.3.2.2 Política Random First

Caso um determinado nó detenha estritamente menos de 4 pedaços do conteúdo a ser disseminado, este escolhe aleatoriamente o próximo pedaço a ser pedido. Esta política, denominada *random first*, visa permitir que um nó obtenha o seu primeiro bloco tão rapidamente quanto possível para que possa permutar blocos com os demais nós do seu *peer set*, o que assume especial importância devido ao algoritmo *choke* (a ser descrito na secção seguinte). De facto, um pedaço escolhido de forma aleatória tem maior probabilidade de ter uma maior taxa de replicação do que um bloco mais raro, o que torna a sua obtenção, em média, mais rápida. Assim que um nó obtém o seu quarto pedaço, abandona a política *random first*, adoptando o algoritmo *rarest first*.

4.1.3.2.3 Política Strict Policy

A política de *strict policy* é empregue pelo BitTorrent ao nível do bloco: quando um bloco pertencente a um dado pedaço é pedido, os restantes blocos do mesmo pedaço são pedidos com a prioridade mais elevada. O objectivo desta política é o de completar o *download* de um pedaço completo o mais rapidamente possível. A sua importância reside no facto de que apenas os pedaços completos são transferíveis, logo torna-se importante minimizar o número de pedaços parcialmente recebidos.

4.1.3.2.4 Política End Game Mode

De modo a atenuar o problema do *último pedaço*, segundo o qual podem existir pedaços mais raros entre os *peers* que dificultem a conclusão do *download*, é adoptada uma política denominada *end game mode*. Esta politica entra em prática assim que um nó tenha pedido todos os pedaços disponíveis, isto é, quando todos os pedaços que compõem o conteúdo a ser distribuído já estejam na sua posse ou já tenham sido pedidos aos seus nós vizinhos. Enquanto neste modo, o nó pede todos os blocos que ainda não recebeu a todos os nós do seu *peer set* que os detenham. De cada vez que

um bloco é recebido, o pedido, pendente junto dos restantes nós, é cancelado. Uma vez que cada pedaço tem um *buffer* reduzido de pedidos, assegura-se que todos os blocos são pedidos a todos os nós efectivamente perto do final do *download*. Por conseguinte, este modo é apenas utilizado numa fase final do *download*, o que resulta num impacto reduzido sobre a prestação global.

4.1.3.2 Escolha de Nós – Algoritmo *Choke*

Um dos aspectos inerentes ao funcionamento de uma rede *peer-to-peer* prende-se com a ausência de uma coordenação central. Por conseguinte, cada nó da rede é responsável por procurar a maximização da sua própria taxa de *download*. Para o efeito, cada nó procura obter dados a partir de nós que se disponibilizam a enviá-los e envia dados para os nós que decidir, o que consiste numa variante do algoritmo *tit-for-tat*.

O sistema de BitTorrent segue uma filosofia de reciprocidade de forma a maximizar os recursos disponíveis na rede e, por conseguinte, as taxas de *download* de cada interveniente. Por conseguinte, cada nó procura enviar dados a quem lhe envia a si. Como consequência, a existência de *free-riders* (nós que participam na rede apenas obtendo dados e nunca enviando, constituindo apenas consomem recursos, não contribuindo) é minimizada, uma vez que estes são penalizados ao não existirem nós que lhes enviem dados.

A este algoritmo, que procura a reciprocidade na troca de pedaços na rede, dá-se o nome de algoritmo de *choke*. Este conhece duas variantes, dependendo se o nó em questão é *leecher* ou *seed*.

4.1.3.3.1 Algoritmo Choke *no Estado* Leecher

A um nível técnico, cada nó num sistema BitTorrent tem, a cada instante, um conjunto com um determinado número (habitualmente quatro) de nós não sufocados (*unchoked*), pelo que o problema se prende com qual os nós a não sufocar. Esta escolha processa-se da seguinte forma:

1. A cada 10 segundos, todos os nós remotos *interessados* no nó em análise são ordenados de acordo com a taxa de *download* do nó local relativamente a estes (ou seja, o nó remoto junto do qual o nó local obteve uma taxa de download mais elevada é classificado em primeiro lugar e, de forma complementar, aquele junto do qual se obteve a taxa de *download* mais baixa é classificado em último lugar). De entre os nós da lista, os primeiros três são seleccionados para não serem sufocados.
2. A cada 30 segundos, um nó adicional é escolhido para não ser sufocado de forma aleatória. A esta medida dá-se o nome de *optimistic unchoke*.

O mecanismo de *optimistic unchoke* tem essencialmente dois propósitos. Em primeiro lugar, o de avaliar a capacidade de *download* de novos nós do *peer set* de um nó. Caso um nó se restrinja unicamente a deixar de sufocar nós de acordo com o primeiro passo, o

conjunto de nós com o qual troca dados é constante e diminuto, desprezando hipóteses de potenciar a sua taxa de *download* junto de nós diferentes. Em segundo lugar, permite fazer o *bootstrap* de novos nós, isto é, oferecer pedaços a nós que ainda não os tenham, conferindo-lhe capacidade de troca. Por conseguinte, quando um novo nó entra no *swarm*, sem qualquer pedaço para troca, tem de esperar que um qualquer nó lhe envie o seu primeiro pedaço, de forma aleatória. Isto resulta num início lento, mas que só dura até que o nó ganhe capacidade de troca.

4.1.3.3.2 Algoritmo Choke *no Estado* Seed

Nas versões iniciais do protocolo BitTorrent, algoritmo de *choke* nos estados de *leecher* e *seed* era idêntico, exceptuando que, no estado de *seed*, o ordenamento de nós para a selecção daqueles a não serem sufocados era feito com base na capacidade de *upload*. Por conseguinte, eram escolhidos 3 nós a cada 10 segundos para não serem sufocados com base na capacidade de um *seed* enviar dados para os mesmos. Esta abordagem maximizava os recursos de largura de banda oferecidos pelo *seed* para a distribuição de um determinado ficheiro. Todavia, favorecia também a capacidade de *download* de nós independentemente da sua contribuição para o *torrent*, o que potenciava a existência de *free-riders*.

As versões mais recentes dos clientes de BitTorrent apresentam uma versão inteiramente nova do algoritmo *choke* no estado de *seed*. À semelhança do que sucedia com a versão anterior e do que sucede com o algoritmo no estado *leecher*, a cada instante existe um determinado número máximo de nós não sufocados (habitualmente quatro). Estes deixam de ser sufocados de acordo com a seguinte política:

1. A cada 10 segundos, os nós remotos não sufocados e interessados são ordenados de acordo com o instante em que deixaram de ser sufocados pela última vez. Os mais recentemente não sufocados são colocados em primeiro lugar.
2. Durante dois períodos consecutivos de 10 segundos, os três primeiros nós são mantidos no estado não sufocado. A estes junta-se um quarto nó, que se encontra sufocado e interessado, escolhido ao acaso.
3. Durante o terceiro período de 30 segundos, os quatro primeiros nós da lista são mantidos não sufocados.

De acordo com esta abordagem, os nós não deixam de ser sufocados de acordo com a sua taxa de *download* relativa ao *seed* em análise, mas relativamente ao instante em que deixaram de ser sufocados pela última vez. Com efeito, a capacidade de *upload* do *seed* passa a ser democraticamente distribuída, de forma alternada, entre os vários nós do seu *peer set*.

O novo algoritmo assume um papel de especial relevância no que diz respeito à reciprocidade e justiça do algoritmo *choke*, ao nunca favorecer *free-riders* em detrimento de nós que contribuem com a sua capacidade de *upload* para o sistema[Coh03].

O BitTorrent foi explicitamente concebido para endereçar o problema dos *free-*

riders, ao recorrer a uma estratégia de reciprocidade *tit-for-tat* de modo a gerar incentivos positivos para que os nós contribuam com recursos para o *swarm*. Todavia, as estratégias de incentivos do BitTorrent podem ser subvertidas, permitindo a prática bem sucedida de *free-riding*[LNKZ05]. Um exemplo desta da situação consiste num cliente modificado de BitTorrent denominado BitTyrant[PIA+07]. Este baseia-se na observação de que, embora uma estratégia *tit-for-tat* desencoraje a prática de *free-riding*, o efeito que domina a prestação do sistema na prática é a contribuição altruísta de um pequeno número de *peers* com elevada capacidade. Por conseguinte, a ideia chave subjacente a este sistema consiste na escolha cuidadosa de *peers* para que se possa tirar partido do seu altruísmo.

4.1.4 Resumo da Prestação do Sistema BitTorrent

O sucesso do BitTorrent, na medida em que é a aplicação para transferência de ficheiros segundo o paradigma *peer-to-peer* mais utilizada globalmente nos últimos anos, deve-se ao facto de que se foca na optimização da distribuição de conteúdos, ao invés da localização dos mesmos de entre uma população de nós. Para o efeito, recorre essencialmente a dois algoritmos que, embora simples, são eficazes.

Segundo testes realizados para determinar o comportamento do sistema em diferentes situações[LUM05], foi possível concluir que:

- O algoritmo *rarest first* resulta numa entropia satisfatória (situação segundo a qual cada nó está interessado na grande maioria dos nós do seu *peer set* a cada instante), prevenindo o reaparecimento de pedaços raros, bem como a ocorrência do problema do último pedaço.
- O algoritmo *choke* é justo, encoraja a reciprocidade, o que aumenta a sua robustez relativamente a *free-riders* na sua última versão.
- Ambos os algoritmos, não sendo óptimos, atingem um nível razoável de eficiência com uma baixa exigência de recursos computacionais.

Porém, embora a entropia introduzida pelo algoritmo *rarest first* seja satisfatória, consiste sempre numa solução subóptima, em que cada nó toma a decisão de obtenção de um pedaço com uma visão limitada do *swarm*. Por outro lado, mesmo que o algoritmo *choke* encoraje a reciprocidade, o sistema BitTorrent permanece vulnerável relativamente a *free-riders*, como referido na secção 3.2.3.

4.2 Codificação de Rede

A Codificação de Rede (termo oriundo do Inglês, *Network Coding*) é um método, proveniente da Teoria da Informação e Teoria da Codificação, que visa atingir o fluxo máximo de informação possível numa rede.

Esta área, surgida no ano 2000[FSC+00], atraiu um crescente interesse ao prometer um impacto significativo no que diz respeito à eficiência da transferência de conteúdos no contexto de uma rede.

De uma forma sumária, a Codificação de Rede surge como um método que permite que os nós intermédios de uma rede não se limitem a encaminhar os fluxos de informação que recebem, mas que também os possam processar. O processamento inerente a esta abordagem consiste na combinação linear de um determinado conjunto de pacotes, com coeficientes aleatórios, para posterior encaminhamento. Por conseguinte, no destino, são recebidas múltiplas combinações lineares oriundos de diferentes caminhos, resultando num sistema linear de equações. Caso o número de equações linearmente independentes seja igual ou exceda o número de pacotes originais, o sistema de equações pode ser invertido, chegando-se aos pacotes iniciais.

Esta abordagem surge em contraste com o encaminhamento tradicional de pacotes, em que cada pacote é reencaminhado indiferentemente pela rede, o que leva a que um nó destinatário tenha de reunir exactamente todos os pacotes originais para construir integralmente o conteúdo disseminado.

Até ao momento, a aplicação de Codificação de Rede a redes de transferência de conteúdos permanece essencialmente no campo da investigação, o que deixa espaço para o debate e para o estudo da real eficiência destes métodos face ao encaminhamento tradicional. Com efeito, esse é o objectivo que se pretende ver atingido com a segunda fase deste trabalho.

4.2.1 Fundamento Teórico da Codificação de Rede

Com recurso à Teoria dos Grafos, uma rede pode ser interpretada como um grafo direccionado e ponderado, em que as suas arestas representam os caminhos segundo os quais pode ser disseminada informação, sendo os seus pesos o fluxo de informação que estas suportam. Posto isto, utilizando o teorema *Max-flow min-cut*, é possível determinar qual a quantidade máxima de informação que pode ser transmitida entre dois nós da rede. Foi demonstrado[LYC03] que este fluxo máximo é atingível, mas que o encaminhamento tradicional de pacotes não é capaz de o atingir.

A noção básica da Codificação de Rede consiste na possibilidade da combinação de dados em nós intermédios da rede. Um receptor que tenha acesso a estes dados pode então deduzir as mensagens originais enviadas. Com recurso a esta abordagem, torna-se então possível atingir o fluxo máximo de informação transmitida entre dois nós de uma rede.

De forma a ilustrar o funcionamento da Codificação de Rede, bem como a sua capacidade de atingir o fluxo máximo de informação disseminado numa rede, face à incapacidade do encaminhamento tradicional atingir o mesmo propósito, considere-se a *butterfly network* representada na figura 8. Nesta rede, representada por um grafo direccionado, consideramos:

- Dois nós fonte, A e B, ambos conhecendo respectivamente os valores x_1 e x_2.
- Dois nós receptores, E e F, cada um querendo conhecer ambos os valores x_1 e x_2.
- Dois nós intermédios, C e D, cujo único propósito é o de encaminhar x_1 e x_2 desde os nós fonte até aos nós receptores.
- Um conjunto de sete arestas, cada qual com a possibilidade de transmitir um dado valor por unidade de tempo.

Usando encaminhamento tradicional, cada nó fonte transmitiria o valor que conhece ao nós a que está ligado: A transmitiria x_1 a E e C, enquanto que B transmitiria x_2 a F e C. Todavia, C

apenas poderia transmitir um valor para D: x_1 ou x_2. Supondo que C encaminha x_1, o nó F ficaria a conhecer ambos os valores x_1 e x_2, enquanto que E ficaria apenas a conhecer x_1, valor que haveria recebido por duas vezes. Por outro lado, caso C enviasse x_2, um problema idêntico se levantaria perante o nó F. Por conseguinte, C necessitaria de transmitir os valores x_1 e x_2 um de cada vez.

Por outro lado, usando um caso simples de codificação de rede (neste caso, a operação *XOR*), C poderia combinar linearmente ambos os valores x_1 e x_2. Por conseguinte, E receberia x_1 directamente a partir de A e $x_1 + x_2$ através do nó D. Conhecendo o valor de x_1 e a combinação linear efectuada, E consegue deduzir o valor de x_2. De forma dual, F consegue deduzir o valor de x_1 conhecendo x_2 e $x_1 + x_2$.

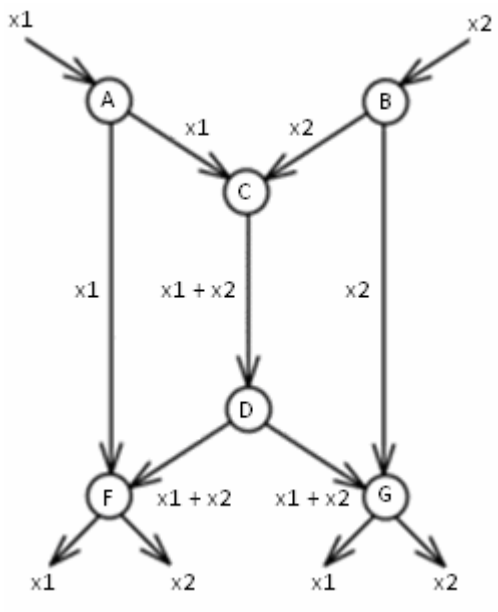


fig. 8[ACLY00] – Esquema de uma *butterfly newtork*

4.2.2 Codificação Linear de Rede

Sem perda de generalidade, é possível modelar uma rede para a distribuição de um determinado conteúdo, com recurso à Teoria dos Grafos, como um grafo acíclico G(V, E), cujas arestas representam caminhos através dos quais podem ser transferidos pacotes (sub-partes) do conteúdo original e em que existe um conjunto de nós fonte $S \subseteq V$ (que detém a integridade do conteúdo original e o distribuem), um conjunto de nós intermédios $I \subseteq V$ (nós com o objectivo de transferir conteúdo) e um conjunto de nós receptores $R \subseteq V$ (que recebem conteúdo). De notar que os conjuntos I e R não são, necessariamente, mutuamente exclusivos, mas S e I, bem como S e R, são-no. Assuma-se também que o conteúdo a ser disseminado consiste num conjunto de p pacotes de dimensão fixa, os quais são a unidade de transferência de dados: por outras palavras, as mensagens trocadas entre nós. Logo, o conteúdo a ser transferido na rede é dado por $M = \{m_1, ..., m_p\}$.

De acordo com a Codificação de Rede, cada nó pertencente ao conjunto I pode, a cada momento, gerar mensagens a disseminar através da combinação daquelas que já detém. A Codificação Linear de Rede consiste num caso particular da Codificação de Rede em que cada

mensagem gerada pelo nó resulta da codificação linear das mensagens previamente recebidas no nó, por coeficientes num campo finito.

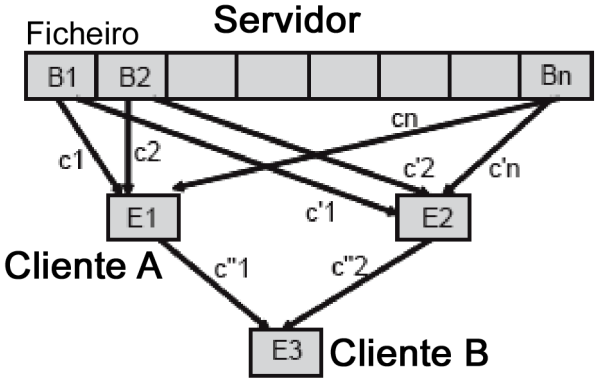
Partindo de um determinado s E S, suponha-se que, num dado instante, este gera uma mensagem $X_{s,k}$ para transferir a outro nó da rede. Esta mensagem é dada pela expressão $X_{s,k} = \Sigma_{(i=1,\ i=p)}\ g_{k,i}\ .\ M_i$, em que g_k representa um vector de coeficientes lineares pertencentes a um campo Galois, GF(2). De notar que, uma vez que as operações são executadas sobre um campo finito, a dimensão de $X_{s,k}$ é a mesma de cada mensagem $m_i \in M$.

Considere-se um nó $r_A \in R$ que recebeu p mensagens distintas provenientes de s, com a condicionante de terem sido utilizados p vectores g_k linearmente independentes, este deparar-se-á com um problema linear do tipo X = G.M. Por conseguinte, caso r_A conheça X e G, poderá descodificar M (aplicando, por exemplo, o Método de Eliminação de Gauss para a resolução de sistemas lineares de equações). Isto significa que, caso r_A receba um conjunto de mensagens linearmente independentes, bem como o vector de codificação responsável pela sua geração, este poderá recuperar integralmente o conteúdo original.

Este raciocínio é também aplicável a nós da rede que recebam um conjunto de mensagens provenientes de nós pertencentes a I. Considere-se então um nó $r_B \in R$, que recebe mensagens a partir de um nó $i \in I$. Uma vez que as mensagens disseminadas por i consistem, em última análise, em combinações lineares das mensagens que compõem o conteúdo original, caso r_B conheça p mensagens linearmente independentes, bem como os seus vectores de codificação, este consegue recuperar o conteúdo original.

Este processo encontra-se ilustrado na figura 9, na qual se apresenta o seguinte caso:

- Um servidor possui um ficheiro que pretende disseminar, o qual é composto por n blocos $B = \{B_1, B_2, ..., B_n\}$.
- Para enviar um bloco E_1 a um cliente A, o servidor combina todos os blocos de B da seguinte forma: escolhe um conjunto aleatório de coeficientes $C = \{c_1, c_2, ..., c_n\}$ e multiplica cada elemento i de B pelo elemento i de C (com $1 \leq i \leq n$), adicionando o resultado das multiplicações.
- De notar que a probabilidade de dois nós independentes escolherem o mesmo bloco de codificações e construir o mesmo bloco depende da dimensão do campo.
- O servidor envia então E_1, juntamente com C, para o cliente A.
- Caso A tenha já um bloco de informação E_2, gerado a partir dos blocos originais e de um vector de codificação C', suponha-se que A quer enviar um bloco de informação E_3 para um cliente B.
- Uma vez que A tem apenas dois blocos na sua posse, dois coeficientes aleatórios c''_1 e c''_2, multiplicando E_1 por c''_1 e E_2 por c''_2 de forma a gerar um bloco E_3.
- O cliente A envia E_3 para o cliente B, juntamente com um vector de codificação C''. De notar que $C'' = c''_1 \cdot c + c''_2 \cdot c'$, isto é, $C'' = \{c''_1 \cdot c_1 + c''_2 \cdot c'_1, c''_1 \cdot c_2 + c''_2 \cdot c'_2, ...\}$.
- Qualquer um dos nós A ou B, caso tenha n blocos de dados linearmente independentes, consegue reconstruir o conteúdo original segundo um processo semelhante à resolução de um sistema linear de equações.



Vector de coeficientes: ($c''_1 c_1 + c''_2 c'_1$, $c''_1 c_2 + c''_2 c'_2$, ...)

fig. 9[GR05] - Exemplo de funcionamento de um sistema de Codificação de Rede

De acordo com [SET03], caso a rede seja composta, no máximo, por 2^8 nós e caso a dimensão do campo no qual são gerados os vectores de codificação seja 2^{16}, é possível afirmar, com uma probabilidade de 0,996, que as p mensagens recebidas por cada nó da rede serão linearmente independentes. No mesmo artigo, defende-se também que um campo de dimensão 2^8 é, na prática, suficiente e que, caso este tenha dimensão 2^{32}, a probabilidade de existirem equações linearmente dependentes é negligenciável comparada com a probabilidade de ocorrerem outras perdas ou problemas numa rede real.

Note-se que, utilizando Codificação Linear de Rede, cada nó tem de transferir, para além de um pacote gerado, um vector de codificação (que consiste em p símbolos extra por pacote). A dimensão do vector de codificação dependerá da dimensão do campo no qual este é definido. Numa situação em que p = 50, o campo é de dimensão 2^8 e cada mensagem é composta por 1400 símbolos (número aproximado de símbolos que um pacote IP pode comportar para a dimensão do campo dada), o volume extra de dados enviado devido ao vector de codificação é de apenas 3%[CWJ03].

4.2.3 Modelo Aplicacional

Nesta secção descreve-se um modelo para uma Rede de Distribuição de Conteúdos *peer-to-peer* que faça uso de Codificação Linear de Rede. O modelo a ser descrito, que segue a abordagem sugerida por Gkantsidis e Rodriguez[GR05] tem por base o sistema BitTorrent, podendo ser visto como uma abordagem com o intuito de melhorar a sua eficiência. É então possível identificar um conjunto de pontos-chave relativos à *framework* do suporte ao sistema cujo comportamento provém do BitTorrent (na secção 3.2 encontra-se descrito, de forma mais compreensiva, o sistema BitTorrent, pelo que os pontos expostos no âmbito desta parágrafo encontram-se aí mais detalhados):

- Assume-se uma população de nós interessados em receber um conteúdo que, originalmente, existe apenas num único servidor. Como é característico das redes *peer-to-peer* cada nó, para além de obter o conteúdo que pretende, também o serve, contribuindo com a sua largura de banda para aumentar a eficiência da rede. O conteúdo original é dividido em k fragmentos, permitindo que cada nó possa servir

fragmentos do conteúdo a partir do momento em que adquire o primeiro fragmento.

- Assume-se que cada nó da rede tem um conhecimento parcial da rede, isto é, conhecem apenas um pequeno subconjunto de todos os seus nós - conjunto que, no sistema BitTorrent, é conhecido por *peer set*. Ainda como no BitTorrent, as relações entre nós são simétricas.
- Existe uma entidade central - conhecida como *tracker* no sistema BitTorrent - junto da qual um nó, aquando da sua entrada na rede, pede uma lista de nós para constituir o seu *peer set*. Caso o número de nós de um dado *peer set* fique abaixo de um dado minorante, este pode contactar o *tracker* de modo a pedir mais nós.
- Cada nó pode assumir um de dois estados: *seed* ou *leecher*. Um *seed* é um nó que já tem o conteúdo na sua totalidade e apenas o serve. Um *leecher* está interessado em obter o conteúdo, servindo-o também como efeito colateral de participar no sistema.
- A cada instante, um nó envia fragmentos do conteúdo original apenas para um conjunto determinado de nós. Esta decisão segue o algoritmo *Choke* do sistema BitTorrent, nas suas vertentes *Seed* e *Leecher*. Por conseguinte, quando um nó é *seed*, procura distribuir a sua capacidade de *upload* democraticamente entre os nós da rede; quando é *leecher*, procura enviar para os nós junto dos quais consegue taxas mais altas de *download*, numa vertente do algoritmo *tit-for-tat*.

Partido da *framework* apresentada, comum ao sistema BitTorrent, passam a discutir-se as especificidades do modelo para uma rede *peer-to-peer* de distribuição de conteúdos que tire partido de métodos de Codificação Linear de Rede.

Sempre que um nó intermédio ou um servidor necessitam de enviar um fragmento de dados para um outro nó, produzem uma combinação linear dos fragmentos que possuem. A combinação linear faz uso de um vector de codificação aleatório, como descrito na secção 3.3.3.

O benefício espectável com a utilização da Codificação Linear de Rede advém da aleatoriedade introduzida de cada vez que um bloco de dados é gerado. Sem recurso a Codificação de Rede, cada nó tem de decidir, a cada instante, que bloco deve obter. Para o efeito, o sistema BitTorrent recorre ao algoritmo *Rarest First*, que visa a obtenção do pedaço mais raro entre a sua vizinhança. Todavia, esta decisão pode não ser óptima, na medida em que é tomada apenas com base em informação local: é possível que um bloco, considerado localmente raro, esteja bem representado noutras áreas da rede. Por outro lado, no sistema BitTorrent, um nó procura sempre adquirir os blocos mais raros de entre o seu *peer set*: caso existe uma forte concorrência no acesso aos referidos blocos junto dos nós que o detêm, geram-se *bottlenecks* na rede, o que deteriora a sua prestação. Com efeito, a abordagem sugerida pela Codificação de Rede elimina a noção de blocos raros e não raros, uma vez que todos os blocos que constituem o conteúdo original são combinados num só. Elimina-se então a necessidade de utilizar um algoritmo para escolha dos pedaços a obter, uma vez que todos são igualmente úteis para um determinado nó (caso sejam linearmente independentes).

Considera-se agora a situação de um nó A, que detém um conjunto de blocos, e de um nó B, que pretende receber blocos de dados. Quando A envia informação para B, este não necessita

de se preocupar com qual a informação a ser disseminada, uma vez que esta é gerada aleatoriamente tendo por base todos os blocos previamente recebidos. Por outro lado, B pode não estar interessado em blocos provenientes de A: só o estará caso estes sejam linearmente independentes da informação que já tem. Por conseguinte, a noção de um nó *interessado* e *não interessado* num outro nó, proveniente do sistema BitTorrent, aplica-se no modelo a ser descrito. Repare-se que, caso A tenha k blocos de dados, sendo que o ficheiro original é composto por n (com k < n), A só *interessará* a B se B tiver menos recebido menos de k blocos a partir de A. Isto porque, tendo k blocos, A nunca poderá gerar mais do que k blocos linearmente independentes entre si. Por conseguinte, enquanto A tiver transmitido menos de k blocos a B, estes blocos denominam-se *inovadores* (uma vez que, provavelmente, são linearmente independentes dos blocos já possuídos por B – esta probabilidade é diferente de 1, uma vez que dois nós independentes podem gerar dois blocos linearmente dependentes, sendo esta probabilidade menor quanto maior for a dimensão do campo finito).

4.3 Descrição dos Simuladores

De forma a determinar comparativamente as prestações do modelo proposto neste trabalho, referente a uma rede *peer-to-peer* de distribuição de conteúdos com recurso a Codificação Linear de Rede, relativamente ao sistema BitTorrent, procedeu-se a simulação de ambos os sistemas em cenários de utilização predeterminados.

Para simular o funcionamento do sistema BitTorrent, recorreu-se a um sistema *open-source*, criado pela Microsoft Research, denominado OctoSim. Por outro lado, a simulação de um sistema com Codificação Linear de Rede foi assegurada através da modificação do simulador OctoSim para que este respeitasse as normas de funcionamento do modelo definido[MR03].

Nas duas subsecções que se seguem descrevem-se ambos os simuladores.

4.3.1 OctoSim: Simulador de BitTorrent

O OctoSim[MR03], desenvolvido em C# por Ashwin Bharambe, Cormac Herley e Venkat Padmanabhan da Microsoft Research, tem como objectivo simular o plano de dados do sistema BitTorrent como uma sucessão de eventos discretos. O seu desenvolvimento surgiu com o propósito de modelar um conjunto de modificações ao protocolo sugeridas pelos autores, as quais não foram consideradas no âmbito neste projecto.

O OctoSim modela a actividade de cada nó, tal como os principais mecanismos do sistema BitTorrent (algoritmos de *rarest first* e *choke*) de forma detalhada. De notar que o simulador permite definir diferentes cenários de simulação, em que é possível parametrizar um conjunto de diferentes aspectos como o tamanho do conteúdo a ser disseminado, a sua divisão em pedaços, o número de nós da rede (*seeds* e *leechers*), a taxa de entrada e saída de nós, as larguras de banda de *upload* e *download* de cada nó e o tempo de simulação. Os dados produzidos no âmbito de cada simulação são também compreensivos, podendo ser distinguidos a nível de eventos e a nível cronológico. No que diz respeito a eventos, podem ser consultados

todos os eventos processados na simulação, nomeadamente a entrada e saída de nós, o início e a finalização do envio de um pedaço de dados e a finalização da recepção de todo o conteúdo (que transforma o *leecher* num *seed*). Relativamente à análise cronológica, é possível consultar, a intervalos parametrizáveis de tempo, os dados relativos a cada nó, que consistem na quantidade de pedaços obtidos e enviados (total e parcialmente), taxas de *download* e *upload*, estado do nó (*seed* ou *leecher*), número de nós no seu *peer set*, número de nós em que está interessado e número de nós não *unchocked*. Uma vez que o presente estudo se foca essencialmente num estudo da prestação de ambos os sistemas, a qual pode ser dada pelo largura de banda útil média empregue para *download* e pelo tempo médio de *download*, o sistema foi modificado para produzir estes dados após cada simulação.

O modelo de rede associa duas ligações a cada nó, cada uma com uma largura de banda parametrizável: a ligação de *download* e a ligação de *upload*. As larguras de banda de cada uma das ligações são independentes entre si, permitindo a modelação de nós com ligações assimétricas à rede. O simulador recorre a estas definições de largura de banda para determinar de forma adequada o atraso em que incorrem as trocas de pedaços entre nós. O cálculo do atraso leva em conta o número de fluxos que partilham as ligações de *upload* e *download* dos nós que trocam um pedaço, o qual está sujeito a variações ao longo do tempo. Uma vez que a determinação destes atrasos é computacionalmente complexa, o número de nós numa determinada simulação deve ser limitado (por exemplo, o máximo de nós recomendados para um computador com processador Intel Pentium 4 2.7GHz, com 1GB de memória RAM, é de 8000).

Posta a complexidade computacional inerente ao modelo definido, os autores do simulador tomaram a decisão de relaxar a sua adaptação do modelo de rede num conjunto de aspectos cujo impacto na prestação global do sistema é menos relevante:

- O atraso na propagação de dados na rede não foi modelado, uma vez que este é apenas relevante quando se está em presença de pacotes pequenos de controlo (como, por exemplo, pedidos de pedaços). Esta simplificação não tem um impacto significativo nos resultados das simulações uma vez que o tempo de *download* é dominado pelo tráfego de dados e o mecanismo de *pipelining* do BitTorrent mascara muita da latência do tráfego de controlo[BHP06]. No contexto do BitTorrent, o *pipelining* consiste em manter vários pedidos de *download* simultaneamente, evitando um atraso entre os pedaços a serem enviados, o que pode influenciar fortemente e de modo negativo as taxas de transferência quando a informação é transferida sobre TCP. Cada pedaço é subdividido em sub-pedaços para transferência, cada um na ordem dos 16KB de tamanho, mantendo sempre um determinado número, tipicamente cinco, de pedidos para *download* em *pipeline*. Sempre que um sub-pedaço é recebido, um novo pedido é enviado[Coh03].
- Os aspectos referentes à dinâmica ao nível dos pacotes em ligações TCP não foi modelado. Assumiu-se então que as várias ligações a um nó partilham igualmente a largura de banda, cada uma com uma fatia da mesma dependendo do número de ligações num determinado instante. De notar que esta simplificação não afecta o modelo de funcionamento do BitTorrent e o facto de que, no âmbito deste projecto, se

pretende comparar a prestação de dois sistemas distintos, assegura-se que as condições a que estes estão sujeitos são idênticas.

- O *bottleneck* de uma ligação entre dois nós é dado pelo *hop* mais lento nessa ligação[SGG02]. Para efeitos de simulação, assumir-se-á que o *bottleneck* na transferência de dados entre dois nós é a largura de banda de *upload* do nó a enviar os dados ou a largura de banda de *download* do nó que os obtém. Os *bottlenecks* interiores de uma rede são, de forma geral, suficientemente rápidos para que os *bottlenecks* nas extremidades da mesma (ou seja, os nós que trocam dados entre si) dominem em situações reais, pelo que se assume que esta abordagem é razoável[MR03].

Foram também feitas duas simplificações no que diz respeito ao comportamento do protocolo BitTorrent propriamente dito:

- O conteúdo a ser disseminado na rede é dividido num número parametrizável de pedaços de dimensão fixa. Todavia, os blocos (fragmentos de pedaços) não foram modelados. Esta simplificação não afecta o comportamento do protocolo uma vez que, quando se inicia a obtenção de um bloco pertencente a um pedaço, são obtidos os restantes blocos do mesmo pedaço de seguida. Por outro lado, a sua modelação introduziria um grau de complexidade não justificável face ao ténue impacto que sem sobre a prestação global do sistema.
- A política de *end game mode*, relativa ao algoritmo *rarest first* e que tem como objectivo acelerar a finalização de um *download* ao permitir a obtenção de diferentes blocos junto de diferentes nós, não foi modelada. Esta decisão é consistente com a análise que se pretende fazer uma vez que esta política não tem impacto na prestação global das características do protocolo e não tem qualquer influência no problema do último bloco (um dos aspectos que se pretende analisar com a comparação de um sistema de encaminhamento tradicional relativamente a um sistema de Codificação de Rede).

4.3.2 OctoSim Modificado: Simulador de BitTorrent com Codificação de Rede

De forma a simular o comportamento de uma rede de distribuição de conteúdos *peer-to-peer* com recurso a Codificação Linear de Rede, procedeu-se à adaptação do simulador OctoSim de modo a respeitar o modelo aplicacional apresentado na secção 3.3.2. Assim, assegurou-se a utilização do mesmo modelo de rede, bem como do mesmo sistema de geração de eventos discretos, permitindo que a comparação dos resultados das simulações efectuadas se centrassem unicamente no que é verdadeiramente relevante: as diferenças de prestações entre o encaminhamento tradicional de dados face ao encaminhamento com recurso a Codificação de Rede.

O principal aspecto da adaptação do OctoSim para utilizar Codificação de Rede consistiu na introdução da capacidade de cada nó gerar um pedaço de dados inovador, representando uma codificação linear dos pedaços na sua posse. Consequentemente, caso um determinado nó A pretenda obter um novo pedaço de dados, solicita-o junto de um dado nó B presente no seu

peer set: todavia, ao contrário do que sucede com o BitTorrent, o nó A não especifica concretamente a B qual o pedaço que pretende uma vez que, à partida, qualquer pedaço gerado por B ser-lhe-á útil. Como efeito colateral, eliminou-se o recurso ao algoritmo *rarest first*, o qual deixava de fazer sentido a partir do momento em que cada nó produz conteúdos inovadores, eliminando-se a noção de bloco raro ou comum. Por outro lado, passaram a surgir a noções de nó *inovador* e *não inovador* (discutidas na secção 3.3.3), que determinam, para um dado nó A, se um qualquer outro nó possa gerar pedaços de dados linearmente independentes, logo úteis, para A. Com efeito, foi então definido um algoritmo que, para cada nó da rede no estado *leecher*, determina se outro qualquer nó da rede lhe é *inovador* ou *não inovador*. A introdução deste algoritmo obriga a que cada nó da rede mantenha um historial de todos os pedaços que obteve junto dos restantes nós, o que aumenta significativamente a complexidade do simulador.

De notar, no entanto, que o algoritmo *choke* foi mantido, uma vez que a sua utilidade permanece independentemente do tipo de estratégia de encaminhamento de dados existente.

Foram, todavia, relaxados dois pontos relativos ao funcionamento de um sistema de Codificação de Rede uma vez que o seu impacto na prestação global do sistema não é significativo, sobretudo face à complexidade que a sua implementação traria ao sistema:

- Cada unidade de informação transferida no contexto de um *torrent* tem, para além de um pedaço do *torrent* (neste caso, um pedaço gerado através da combinação linear de outros pedaços, mas que permanece da mesma dimensão), o vector de codificação utilizado. No entanto, este volume suplementar de informação transmitida é pouco relevante.
- Existe uma probabilidade, diferente de zero, de um dado nó receber dois pedaços de dados linearmente dependentes a partir de nós inovadores e independentes. Todavia, admite-se que este factor não tem um impacto significativo na prestação global do sistema, uma vez que a sua probabilidade de ocorrência é baixa e implica apenas que necessita de ser obtido um número de pedaços para que todos sejam linearmente independentes.

4.4 Resultados Experimentais

O objectivo das simulações a serem feitas consiste na determinação do ganho obtido com a introdução de Codificação Linear de Rede no sistema BitTorrent, identificando-se também as situações em que este é mais assinalável. Com efeito, procedeu-se à simulação dos sistemas BitTorrent tradicional e BitTorrent com Codificação de Rede de modo a analisar as suas prestações em função da variação de dois parâmetros fundamentais: a largura de banda dos nós participantes no *swarm* e a proporção de *seeds* na população de nós participantes.

A análise realizada focou-se na interpretação de dois parâmetros fundamentais: a largura de banda média utilizada pelos nós para *download* e o tempo médio de *download*. O primeiro parâmetro traduz-se na proporção da largura de banda de *download* consegue alocar para a obtenção do conteúdo a ser distribuído. O segundo consiste no tempo médio que cada nó leva para completar a obtenção integral do conteúdo.

As simulações realizadas procuraram espelhar *swarms* com um elevado número de utilizadores (na ordem dos milhares) durante uma janela temporal de algumas horas com o objectivo de analisar os padrões de funcionamento dos sistemas.

4.4.1 Análise em Função da Largura de Banda

Para efeitos de simulações, um *swarm* pode ser caracterizado com recurso a um conjunto diverso de parâmetros, os quais se referem ao comportamento de *seeds* e *leechers*, bem como características do conteúdo a ser distribuído e limitações do *swarm*. De forma a analisar o impacto da largura de banda dos nós participantes no *torrent* na eficiência de ambos os sistemas, todos os parâmetros foram fixados ao longo de um conjunto de simulações à excepção das larguras de banda de *download* e *upload* dos clientes (tabela 1). Os valores seleccionados para largura a probabilidade de saída de *leechers* quando terminam o seu *download* (e se tornam *seeds*) e condição de saída de *seeds* foram baseados em observações efectuadas ao longo de 5 meses num *swarm* referente à distribuição RedHat 9 do sistema operativo Linux[IUB+04]. A capacidade de *upload* das *seeds* fixas foi definida como 1,5Mbps, reflectindo o *throughput* médio de um *torrent* com base em análises de diferentes *swarms*[DGLZ07]. A taxa de entrada de nós foi seleccionada de forma a testar a prestação dos sistemas em situações com uma taxa de entrada de *leechers* elevada, embora muito longe do que sucede com uma típica *flash crowd* (em que centenas ou milhares de *leechers* se juntam ao *swarm* durante um intervalo de escassos segundos[BH05]). Por questões de factibilidade das simulações, a duração das mesmas foi limitado a duas horas. A dimensão de conteúdo foi definida em função dessa limitação, de forma a que uma porção razoável de *leechers* conseguisse terminar o *download* mesmo com ligações relativamente lentas mas, ainda assim, mantendo um conteúdo com um número significativo de blocos para que os diferentes algoritmos de escalonamento de pedaços de ambos os sistemas fossem testados.

A presente análise tem como propósito determinar as eficiências relativas de ambos os sistemas em função da largura de banda dos nós participantes. Por conseguinte, foi definido um

número fixo de *seeds*, com uma largura de banda de *upload* constante de 1,5Mbps (que reflecte o *throughput* médio de um *torrent*[DGLZ07]), sendo variadas as capacidades de *download* e *upload* dos restantes nós participantes. A partir desta análise será possível determinar qual o sistema que faz um uso mais eficiente da capacidade de *download* de cada nó e qual optimiza as capacidades gerais de *upload* (tendo em conta que as capacidades de *upload* das *seeds* iniciais são constantes e a dos *leechers* variáveis).

Tabela 1 – Parâmetros fixos das simulações para análise em função da largura de banda dos nós participantes

Tempo de simulação	7200 segundos (2 horas)
Dimensão do conteúdo	100 MB
Número de pedaços	400 (pedaços de 256 KB)
Número de *seeds* fixas [1]	100
Largura de banda (*upload*) das *seeds* fixas	1,5 Mbps (1536 Kbps) [4]
Número máximo de nós	5000
Taxa de entrada de nós	1 por segundo
Probabilidade de saída de uma nova *seed* [2]	75%[4]
Condição de saída de uma *seed* [3]	Servir 88% do conteúdo (352 pedaços) [4]

[1] – *Seeds* que servem conteúdo durante toda a simulação.

[2] – Probabilidade que cada utilizador da rede saia do *swarm* ao passar do estado de *leecher* para o estado de *seed* (não se aplica às *seeds* iniciais).

[3] – Caso um utilizador passe a *seed* e permaneça no *swarm* (não saia imediatamente), não sai até que esta condição se verifique.

[4] – Valores médios observados num *swarm* durante um período de 5 meses, com base em 20584 sessões únicas de *download*, de um *torrent* referente à disseminação do Linux Redhat 9 (num volume de 1,77GB)[IUB+04]

Os valores para as larguras de banda da população de nós participantes foram escolhidos de modo a espelhar características de ligações reais (tabela 2), abrangendo desde ligações extremamente lentas até excepcionalmente rápidas[KAGL01] [Nun06].

Tabela 2 [KAGL01] [Nun06] – Larguras de banda da população de nós participantes

	Capacidade de *Upload*	**Capacidade de *Download***
GSM Circuit Switched Data (CSD)	10Kbps	10Kbps
High-Speed GSM Circuit Switched Data (HCSD)	43Kbps	14Kbps
Dial-Up	56Kbps	34Kbps
General Packet Radio Device (GPRS)	80Kbps (50%), 60Kbps (50%)	20Kbps (50%), 40Kbps (50%)

Integrated Service Digital Network (ISDN)	128Kbps (50%), 64Kbps (50%)	128Kbps (50%), 65Kbps (50%)
Enhanced General Packet Radio Device (EGPRS)	236Kbps (50%), 177Kbps (50%)	59Kbps (50%), 118Kbps (50%)
Cable	640Kbps	128Kbps
Asymmetric Digital Subscriber Line (ADSL) Lite	1Mbps (1024Kbps)	512Kbps
ADSL	8Mbps (8192Kbps)	1Mbps (1024Kbps)
ADSL 2	12Mbps (12288Kbps)	3,5Mbps (3584Kbps)
ADSL 2+	24Mbps (24576Kbps)	3,5Mbps (3584Kbps)
Very High Digital Subscriber Line (VDSL)	52Mbps (53248Kbps)	12Mbps (12288Kbps)
VDSL 2	100Mbps (102400Kbps)	24Mbps (24576Kbps)
Fiber To The Buildings (FTTB)	25Mbps (33% - 25600Kbps), 50Mbps (34% - 51200Kbps), 75Mbps (33% - 76800Kbps)	75Mbps (33% - 76800Kbps), 50Mbps (34% - 51200Kbps), 25Mbps (33% - 25600Kbps)
Fiber To The Homes (FTTH)	2560Mbps (2621440Kbps)	1228Mbps (1258291Kbps)

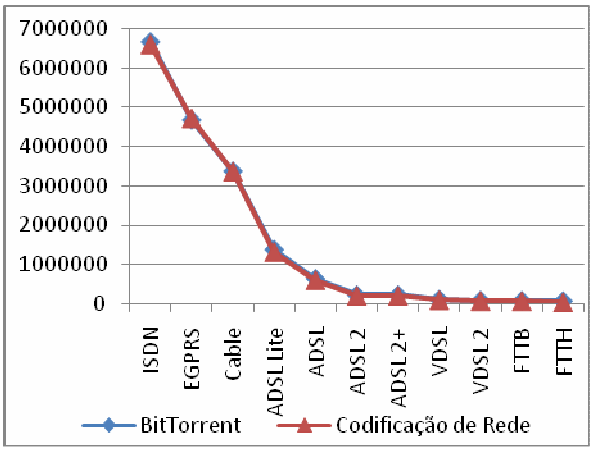


Fig. 10 – Tempo médio de download por nó (em segundos)

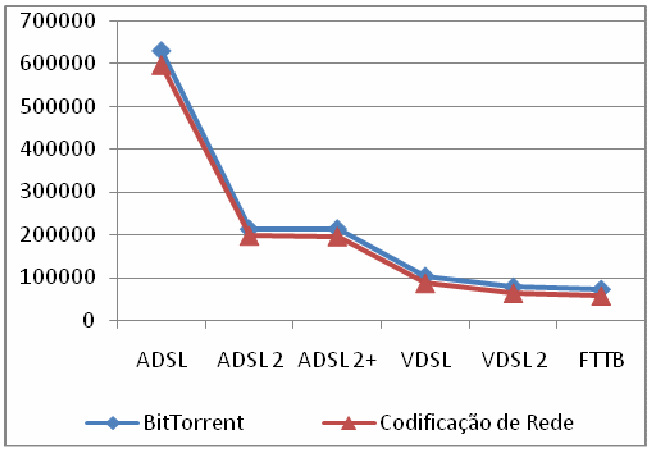


Fig. 11 – Tempo médio de download por nó (em segundos) considerando apenas as ligações entre ADSL e FTTB

Atendendo ao tempo médio de *download*, é possível verificar que a prestação de ambos os sistemas aumenta em a par com o aumento das larguras de banda de *upload* e *download* dos nós participantes. Todavia, existem duas observações importantes que devem ser registadas: em primeiro lugar, de um modo geral, o sistema com Codificação de Rede é mais eficiente do que o BitTorrent, dado que os *downloads* levam, em média, menos tempo a serem concluídos (fig. 10 e 11); em segundo lugar, o ganho introduzido com a Codificação de Rede é tanto maior quanto mais largas forem as bandas de *upload* e *download* dos nós participantes (fig. 12).

Note-se que as figuras 10 e 11 representam os mesmos valores no mesmo cenário de teste: todavia, na figura 6 a análise restringe-se a larguras de banda entre o ADSL e o FTTB para que a diferença de prestações entre ambos os sistemas seja mais facilmente identificável.

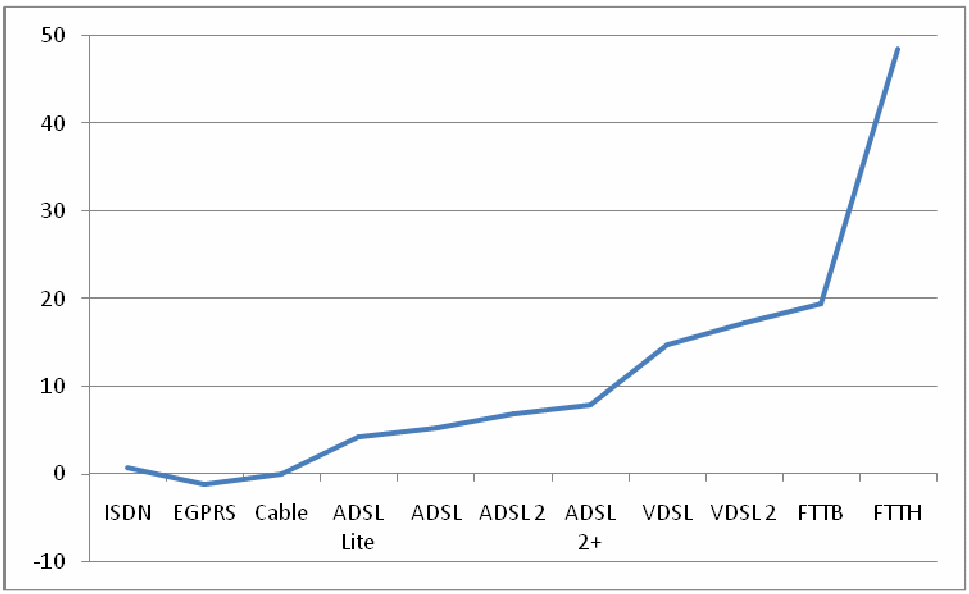


Fig. 12 – Ganho percentual do sistema de Codificação de Rede a nível de tempo médio de *download* por nó em função das Larguras de Banda

A primeira constatação deriva do facto de se usar uma política de escolha e distribuição de pedaços sub-óptima no sistema BitTorrent, situação que não se verifica com o uso de Codificação de Rede em que cada nó gera blocos inovadores para os restantes nós – eliminam-se situações como o problema do *último bloco*. É possível também verificar que existe uma diferença notória de eficiência de ambos os sistemas entre as ligações ADSL e ADSL 2. Esta diferença deve-se essencialmente ao facto de que, até às ligações ADSL, as *seeds* iniciais eram os principais fornecedores de conteúdo, tanto em número como em capacidade de *upload*; no entanto, na ligação ADSL 2, a capacidade de *upload* dos nós participantes triplicou, sendo duas vezes superior à das *seeds* iniciais, tornando estas no *bottleneck* do sistema.

A segunda constatação prende-se com uma utilização mais eficaz da largura de banda da população de nós participantes no *swarm* no sistema de Codificação de Rede: uma vez que cada nó tem a possibilidade de gerar blocos inovadores, existe um maior número de *leechers* com capacidade de fornecer pedaços inovadores a outros nós; o mesmo já não acontece com o sistema BitTorrent, em que podem existir blocos raros, pelo que certos *leechers* podem ter uma menor utilidade no que diz respeito ao *upload* de conteúdos. A estratégia utilizada pelo BitTorrent, embora promova a heterogenização de blocos entre os nós do *swarm*, é sub-óptima, sendo propensa à criação de *bottlenecks* em certos nós da rede: aqueles que contêm pedaços mais *raros*. Por conseguinte, ao existir uma maior proporção de *leechers* que servem activamente conteúdo no sistema de Codificação de Rede, a sua eficiência aumenta a par com o aumento das larguras de bandas dos referidos nós (aproximando-se mesmo dos 50% em, situações extremas, como a utilização de FTTH).

Note-se também que os algoritmos de selecção de pedaços podem influir na diferença de prestações de ambos os sistemas. No sistema de Codificação de Rede, a escolha de pedaços é

óptima, uma vez que qualquer pedaço *inovador* é útil para a reconstrução final do conteúdo. No sistema BitTorrent isto não se verifica: embora se utilize uma estratégia para minimizar a existência de pedaços *raros* (algoritmo *rarest first*), não é possível garantir a sua inexistência, uma vez que este algoritmo se baseia apenas numa visão parcial do *swarm*. Partindo do pressuposto que a existência de pedaços raros no sistema BitTorrent é possível, advém a conclusão lógica de que existe uma dependência mais forte de *leechers* relativamente a *seeds* do que no sistema de Codificação de Rede: se, para um determinado *leecher*, um dado pedaço que pretende obter é raro, os únicos nós que garantidamente terão este pedaço na sua posse serão os *seeds*. Dada esta dependência de *leechers* relativamente a *seeds* no sistema BitTorrent, é possível concluir que a capacidade de *upload* dos *seeds* se pode tornar num *bottleneck* do sistema à medida que a capacidade de *download* dos *leechers* se torna significativamente superior.

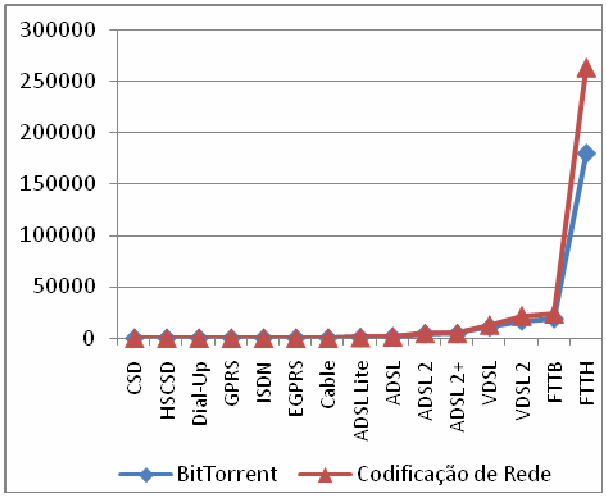


Fig. 13 – Largura de banda média útil para download por nó (em Kbits)

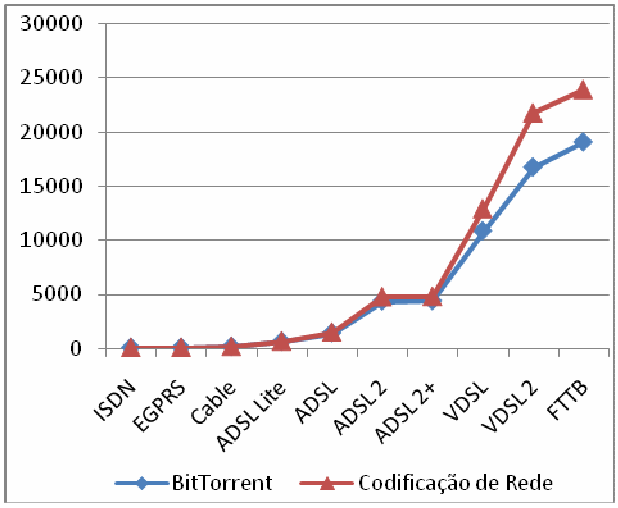


Fig. 14 – Largura de banda média útil para download por nó (em Kbits) considerando apenas as ligações entre ISDN e FTTB

Analisando agora a prestação de ambos os sistemas em função da largura média utilizada para cada nó para efeitos de *download* (fig. 11 e fig. 12), é possível confirmar as observações feitas previamente: a eficiência de ambos os sistemas aumenta com a largura de banda dos nós participantes, tal como o ganho relativo do sistema de Codificação de Rede face ao BitTorrent.

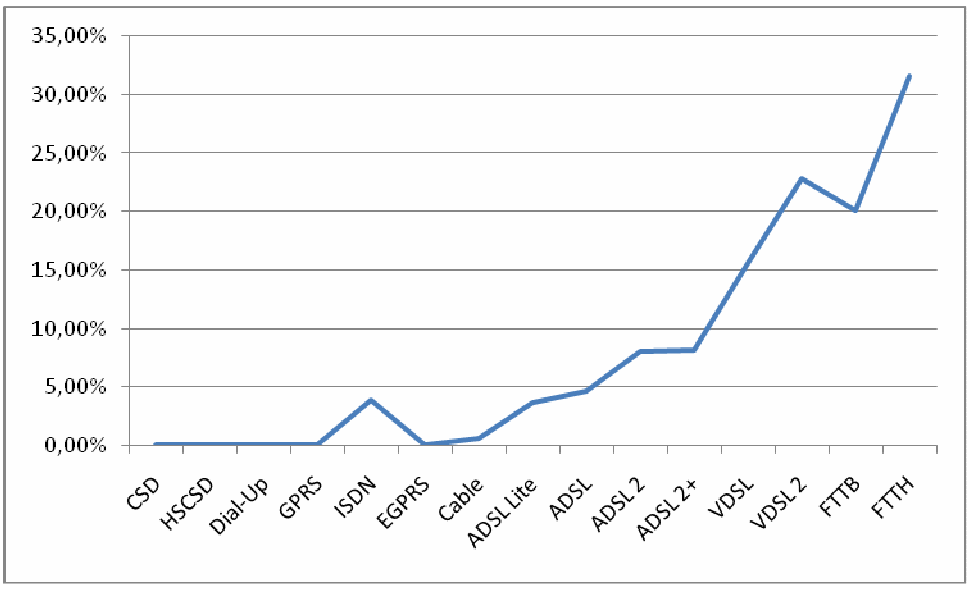


Fig. 15 – Ganho percentual do sistema de Codificação de Rede a nível de largura de banda média utilizada para *download* por nó em função das Larguras de Banda dos mesmos.

Quanto ao ganho introduzido com a utilização de Codificação de Rede relativamente à largura de banda média utilizada para *download* (fig. 13) verificou-se que, de uma forma geral, este aumenta com o aumento da largura de banda dos *leechers*, como é possível constatar pelo gráfico da figura 15. Todavia, existem três excepções a este aumento. Em primeiro lugar nas ligações mais *lentas*, nomeadamente entre CSD e GPRS, e atendendo a uma granularidade de Kilobits por segundo na taxa de *download*, o recurso a métodos de Codificação de Rede não introduz qualquer ganho sobre a utilização do sistema BitTorrent. Em segundo lugar, este aumento não é monótono: a partir de uma ligação ISDN cria-se uma clivagem entre as prestações de ambos os sistemas, num ganho que se quebra nas ligações EGPRS e *Cable*. Estas quebras prendem-se sobretudo com a diminuição da largura de banda de *upload* relativamente à de *download*: na ligação ISDN ambas as larguras de banda eram idênticas, ao passo que numa ligação de EGPRS o rácio entre a largura de banda para *upload* e *download* é de 1:3, valor que evolui para 1:8 numa ligação *Cable*. Já numa ligação ADSL *Lite*, o rácio entre as duas larguras de banda é de 1:2. Todavia, à medida que estes valores aumentam, o ganho do sistema de Codificação de Rede aumenta naturalmente: o sistema de Codificação de Rede consegue tirar partido de uma maior largura de banda dos *leechers* enquanto que, por outro lado, o sistema BitTorrent padece de uma maior dependência das *seeds* iniciais dado o problema dos pedaços raros. Em terceiro lugar, e por último, a quebra de ganho que se verifica na ligação FTTB prende-se com a redução da capacidade de *download*: embora esta ligação apresente um rácio de 1:1 entre as duas larguras de banda dos nós participantes, a velocidade de *download* cai para metade quando comparada com uma ligação VDSL 2. Consequentemente, mesmo que os nós tenham uma grande capacidade de *upload*, não tiram total partido desta dado capacidade de *download* se torna no *bottleneck*, aproximando a eficiência de ambos os sistemas.

4.4.2 Análise em Função do Tempo de *Seed*

A prestação de uma rede *peer-to-peer* de distribuição de conteúdos depende fortemente da proporção de nós participantes que servem activamente conteúdos: quanto maior o número destes nós e quanto maior a sua capacidade de *upload*, maior a eficiência do sistema. Ambos os tipos de nós participantes no sistema BitTorrent servem conteúdos: os *leechers* servem um determinado subconjunto dos pedaços do conteúdo original (aqueles que já obtiveram desde que entraram no *torrent*); os *seeds* servem todos os pedaços do conteúdo e é esta a sua função exclusiva.

A análise prévia permitiu, entre outros aspectos, analisar o impacto da capacidade dos *leechers* servirem conteúdos ao serem feitas várias simulações em que a sua capacidade de *upload* era variada. Esta análise, por seu turno, tem como objectivo analisar o impacto que a proporção de *seeds* na população de nós participantes tem na eficiência global do sistema.

De forma a determinar a eficiência do sistema em função da proporção de *seeds* de entre os nós participantes, foram fixados todos os parâmetros de simulação (tabela 3), sendo variada a condição de saída de um *leecher* tornado *seed*. Relativamente à simulação anterior, a dimensão do conteúdo foi aumentada para 300 MB, que corresponde à dimensão média dos conteúdos a serem distribuídos em *swarms* observados[IUB+04]. Este aumento justifica-se pelo facto de introduzir um maior período de *seed* (os *downloads* levam mais tempo a terminar), pelo que se pode avaliar melhor a importância da saída de *seeds* a meio de *downloads*. Por último, a taxa de entrada de nós foi reduzida para 1 novo *leecher* a cada 5 segundos, para que as simulações fossem tecnicamente factíveis.

Tabela 3 – Parâmetros fixos das simulações para análise em função do tempo de *seed*

Tempo de simulação	7200 segundos (2 horas)
Dimensão do conteúdo	300 MB[4]
Número de pedaços	1200 (pedaços de 256 KB)
Número de *seeds* fixas [1]	1000 (durante 20 minutos)[3]
Largura de banda (*upload*) das *seeds* fixas	1,5 Mbps (1536 Kbps) [4]
Número mínimo de *seeds* no *swarm*	1
Taxa de entrada de nós	1 por cada 5 segundos
Probabilidade de saída de uma *seed* [2]	0%
Largura de banda para *upload*	512 Kbps
Largura de banda para *download*	4 Mbps (4096 Kbps)

[1] – *Seeds* que servem conteúdo durante toda a simulação. O *torrent* foi iniciado com um total de 1000 *seeds* fixas que permaneceram no mesmo durante 20 minutos das 2 horas de simulação. O objectivo passou por heterogenizar a distribuição de pedaços entre os restantes nós participantes (uma vez que, durante esses 20 minutos, manteve-se uma taxa de entrada de 1 nó por segundo, resultando numa gama de nós entre aqueles com todos os pedaços do conteúdo até aqueles com poucos ou nenhum pedaço).

[2] – Probabilidade que cada utilizador da rede saia do *swarm* ao passar do estado de *leecher* para o estado de *seed* (não se aplica às *seeds* iniciais). Esta probabilidade foi anulada, uma vez que o objectivo da simulação passava por variar o tempo de permanência de uma *seed* no *torrent*.

[3]] – As *seeds* iniciais foram mantidas durante um período de 20 minutos, o qual equivale ao momento em que se começam a completar os primeiros *downloads* no sistema com Codificação de Rede. O objectivo heterogenizar a distribuição de pedaços entre os *leechers* para que a análise seja feita não sobre um período de arranque do *torrent*, mas sobre um período de estabilização.

[4] – Valores médios observados num *swarm* durante um período de 5 meses, com base em 20584 sessões únicas de *download*, de um *torrent* referente à disseminação do Linux Redhat 9 (num volume de 1,77GB)[IUB+04]

[5] – Valores associados a uma ligação ADSL, representativa de mais de metade das ligações à Internet por todo o Mundo.[Int07]

Ao longo das várias simulações efectuadas, o tempo de permanência de cada *leecher* tornado *seed* foi variado entre 1 e 90 minutos de forma a avaliar, de forma comparativa, a prestação de ambos os sistemas.

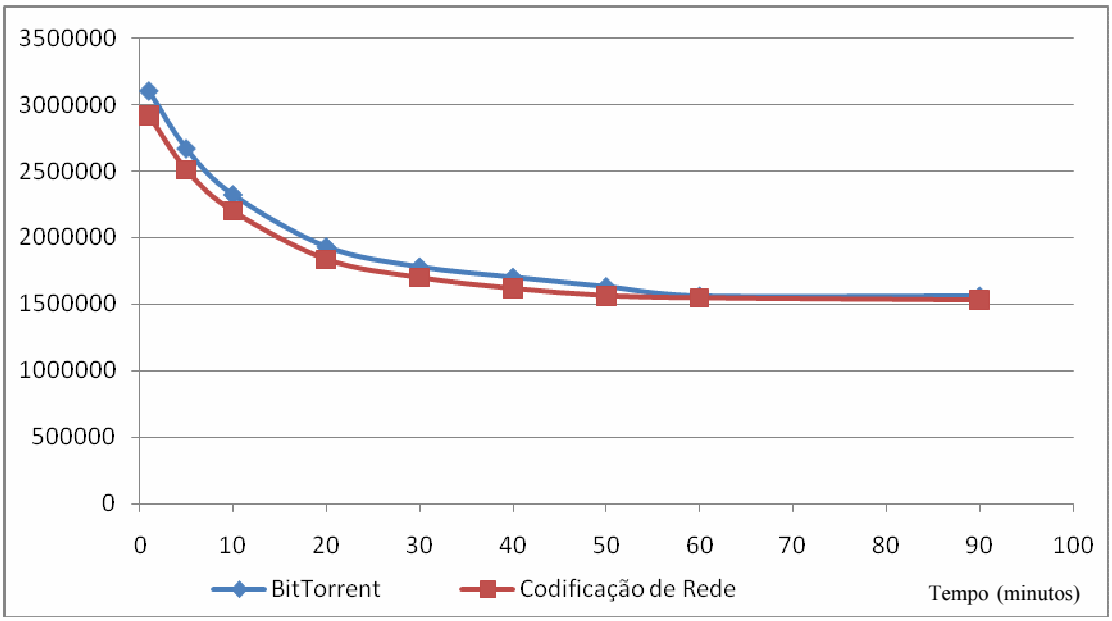


Fig. 16 – Tempo médio de download por nó (em minutos) após os primeiros 10 minutos de simulação

Para esta experiência, as larguras de bandas escolhidas para a população de *leechers* reflectiram as mais usuais numa ligação ADSL. Posto isto, é possível observar que, relativamente ao tempo médio de *download* e de uma forma geral, o sistema de Codificação de Rede apresenta ganhos relativamente ao BitTorrent, confirmando as observações realizadas na primeira experiência. Os ganhos situam-se também na mesma gama de valores (atingem os 6% - fig. 15).

É necessário também notar que, nesta experiência, a densidade de *leechers* é menor do que na experiência prévia, derivada de uma menor taxa de entrada de nós. Tendo em conta o sistema BitTorrent, o tamanho do *peer set* de cada *leecher* não depende do número de *peers* no *swarm* (desde que esse número seja igual ou superior ao número máximo de *peers* num *peer*

set). Por conseguinte, ao ser diminuído o número de *peers* no *swarm* relativamente às simulações anteriores, a perspectiva de cada *leecher* sobre todo o *swarm* é maior, dado que o tamanho do *peer ser* permanece constante, pelo que as decisões de cada *leecher* sobre qual o próximo pedaço a obter aproximam-se da escolha óptima. Por conseguinte, o sistema BitTorrent pode beneficiar, comparativamente às simulações anteriores, de uma taxa de entrada de *leechers* mais baixa.

Analisando a variação dos tempos médios de *download*, para ambos os sistemas, em função do tempo de *seed* verifica-se que este varia de forma inversa ao tempo de permanência das *seeds*. Com efeito, quanto mais tempo as *seeds* permanecem em média no *swarm*, mais eficientes ambos os sistemas se tornam.

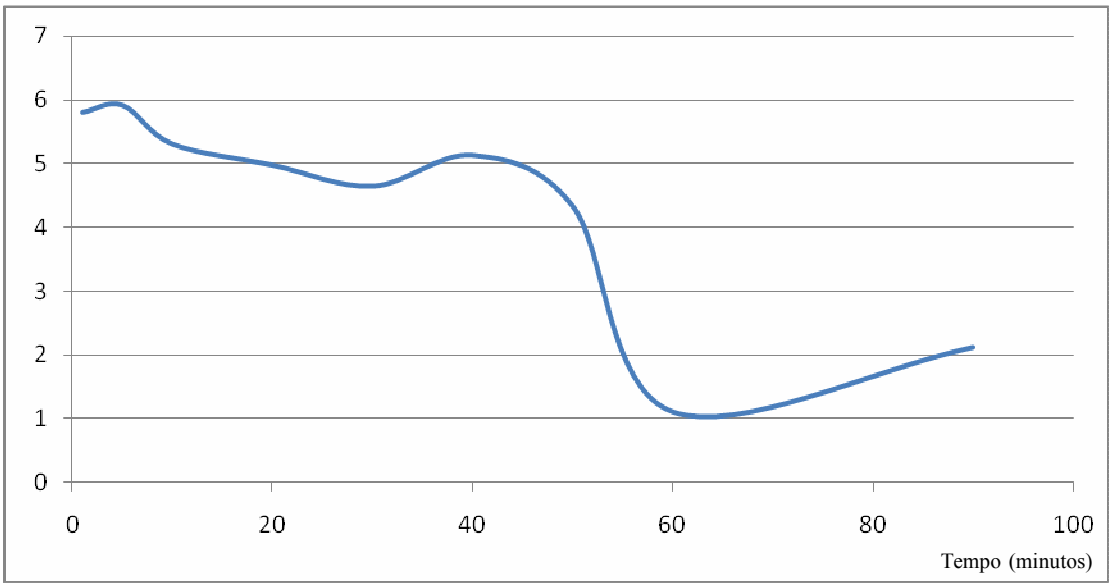


Fig. 17 – Ganho percentual do sistema de Codificação de Rede a nível de tempo médio de *download* por nó (em minutos) após os 10 primeiros minutos de simulação

Atendendo exclusivamente ao ganho do sistema de Codificação de Rede relativamente ao BitTorrent, representado graficamente na figura 17, verifica-se que este é tanto maior quanto menor a proporção de *seeds* no *swarm*. Este resultado corrobora a ilação extraída da experiência prévia de que a prestação do sistema BitTorrent está mais dependente da concentração de *seeds* do que a do sistema de Codificação de Rede. Para tempos de *seeding* muito baixos, verificamos que o ganho aumenta ligeiramente. Este resultado, que pode também ser advir de ruído nas simulações, deve-se provavelmente a uma idiossincrasia das simulações, fruto do patamar mínimo de *seeds* definido: de forma a manter o *swarm vivo*, foi definido um patamar mínimo de uma *seed*, pelo que caso o seu tempo de *seed* fosse ultrapassado, esta permaneceria a servir conteúdo enquanto não surgissem outras *seeds*. Quando o tempo de *seed* é inferior a 5 minutos, para o sistema BitTorrent, o número de *seeds* é, durante uma parte muito significativa da simulação, igual ao patamar mínimo – o sistema BitTorrent apresenta então prestações semelhantes para tempos de *seed* entre 1 e 5 minutos, como se pode verificar pelo gráfico da

figura 16. Por conseguinte, enquanto o sistema de Codificação de Rede experimenta alguma evolução de prestação para tempos de *seed reduzidos*, o mesmo não acontece (ou, acontecendo, é mais atenuado) com o sistema BitTorrent, o que explica a evolução do ganho do sistema de Codificação de Rede inversa ao esperado.

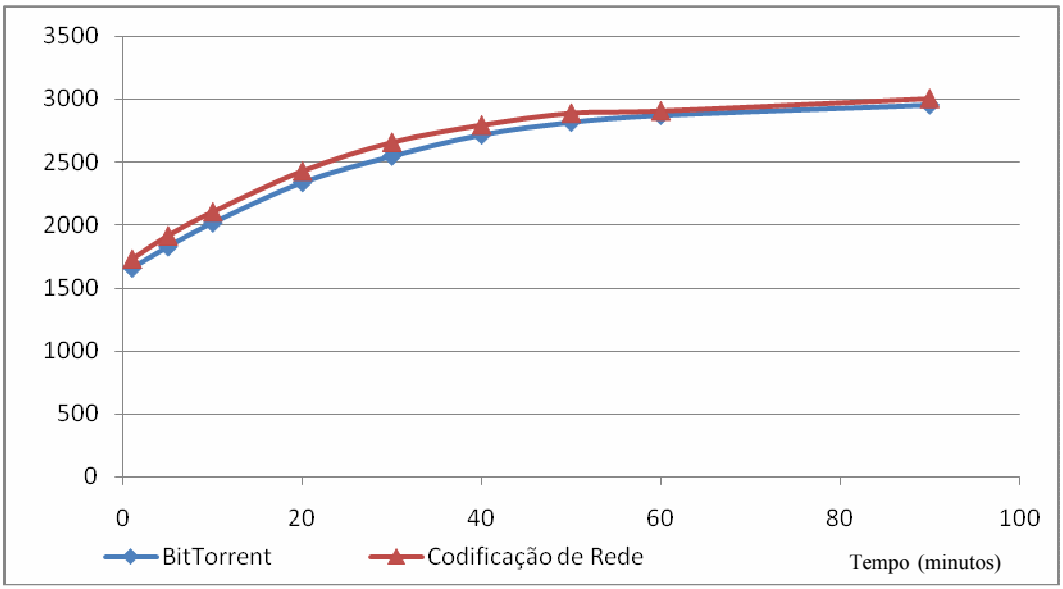


Fig. 18 – Largura de banda média de *download* por nó (em Kbits) em função do tempo de permanêncoa de *seeds*.

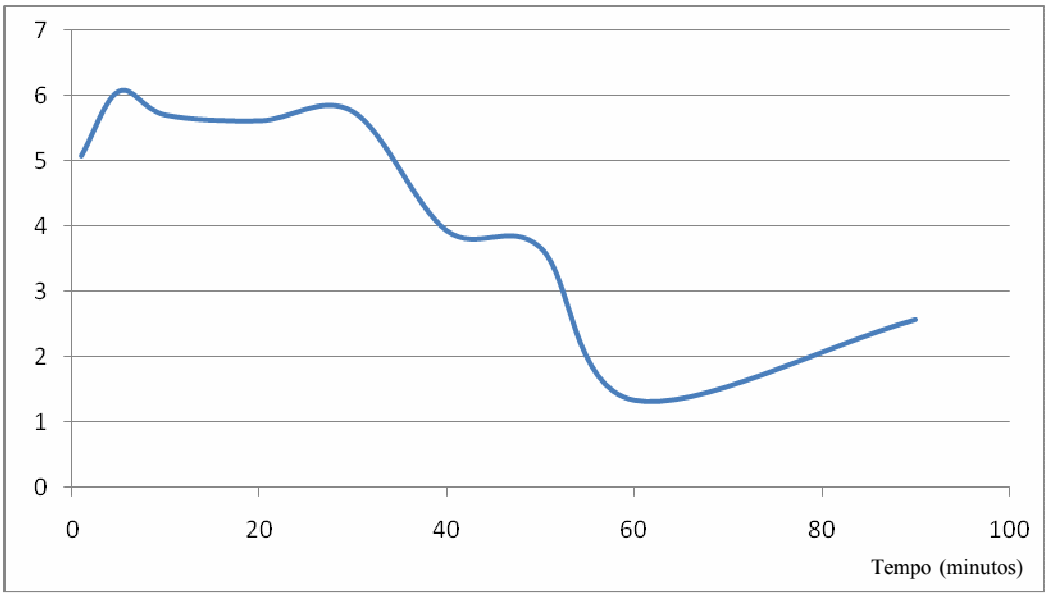


Fig. 19 – Ganho do sistema de Codificação de Rede relativamente ao BitTorrent no que diz respeito à largura de banda média de *download* por nó (em Kbits) em função do tempo de permanêncoa de *seeds*.

Analisando a prestação de ambos os sistemas relativamente à largura de banda utilizada para *download* por cada nó, é possível constatar também uma maior eficiência do sistema de Codificação de Rede que se traduz num ganho que decresce com o aumento do tempo de *seed*.

A excepção que se verifica relativamente a tempos inferiores de *seed* tem a mesma explicação do que para o tempo médio de *download.*

De modo a confirmar experimentalmente a premissa de que o sistema de Codificação de Rede opera melhor do que o BitTorrent em condições em que o número de *seeds* é mais escasso, procedeu-se a uma segunda experiência, retornando aos parâmetros base da primeira simulação (dimensão do conteúdo e taxa de entrada de nós). Foi definido que o parâmetro variável seria o número de *seeds* fixas. Foi também estipulado que cada *leecher* abandonaria o *swarm* ao terminar o *download*, permitindo controlar exactamente o número de *seeds* do *swarm*. Tanto os *leechers* como os *seeds* foram modelados de acordo com uma ligação ADSL.

Tabela 4 – Parâmetros fixos das simulações para análise em função do número de *seeds*.

Tempo de simulação	7200 segundos (2 horas)
Dimensão do conteúdo	100 MB
Número de pedaços	400 (pedaços de 256 KB)
Número de *seeds* fixas [1]	100
Largura de banda (*upload*) das *seeds* fixas	1 Mbps (1024 Kbps)
Taxa de entrada de nós	1 por segundo
Probabilidade de saída de uma *seed* [2]	0%
Largura de banda para *upload*	1Mbps (1024 Kbps)
Largura de banda para *download*	8 Mbps (8192 Kbps)

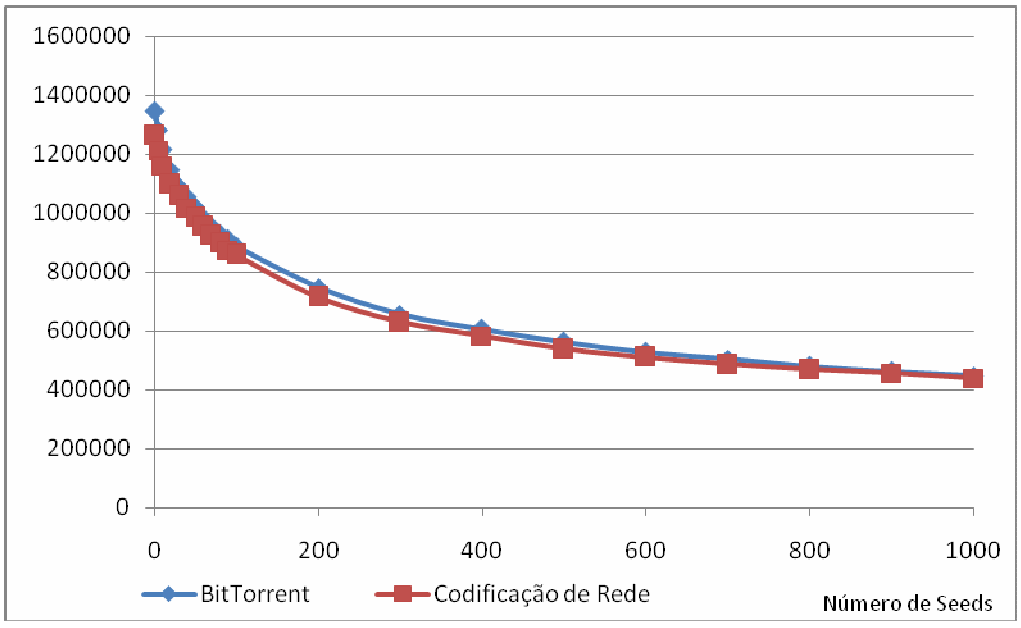


Fig. 20 – Tempo médio de download por nó (em milisegundos) em função do número de *seeds*

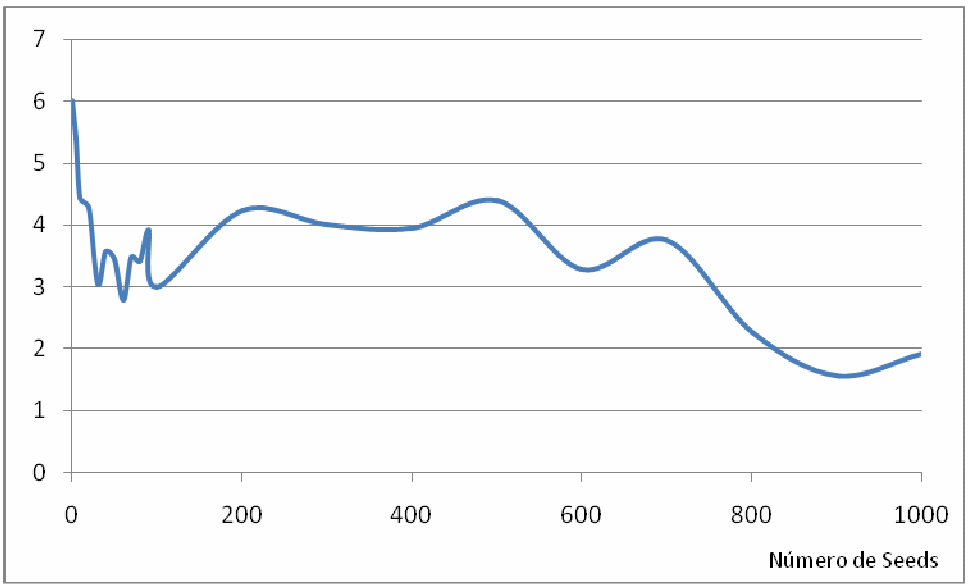


Fig. 21 – Ganho percentual do sistema de Codificação de Rede a nível de tempo médio de *download* por nó em função do número de *seeds*.

Confirmando as ilações tiradas relativamente à simulação em função do tempo de *seed*, verificamos que:

- Ambos os sistemas apresentam prestações (em tempo médio de *download*) incrementalmente melhores com o aumento do número de *seeds* no *swarm*.
- O sistema de Codificação de Rede é globalmente melhor, com ganhos entre 1% e 7%.
- Os ganhos do sistema de Codificação de Rede evidenciam-se para números menores de *seed* no *swarm*. Para números na ordem da centena de *seeds*, embora se verifique uma queda atenuada do referido ganho, este aparenta estabilizar-se.

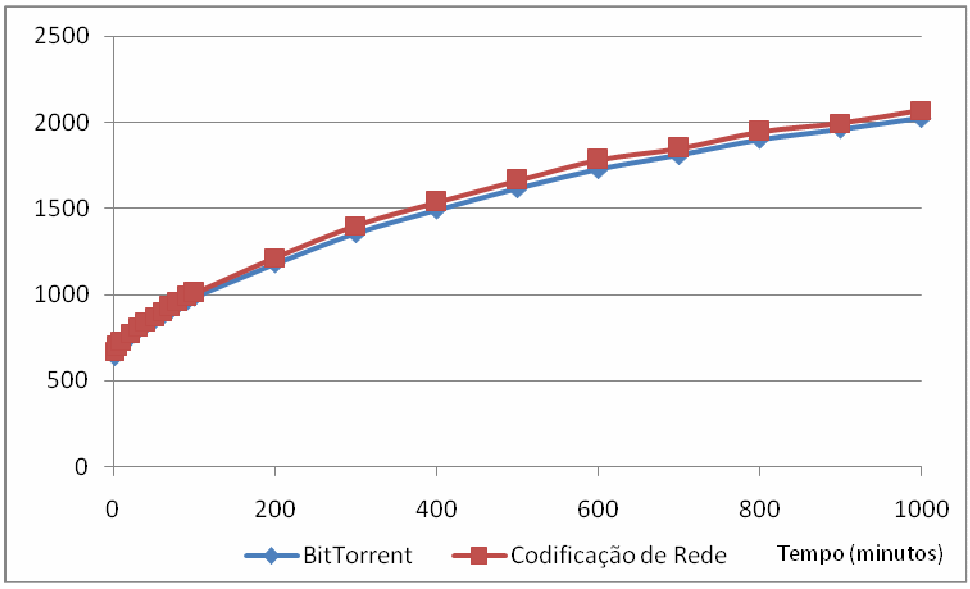


Fig. 22 – Largura de banda média utilizada para *download* (Kbits) em função do número de *seeds*.

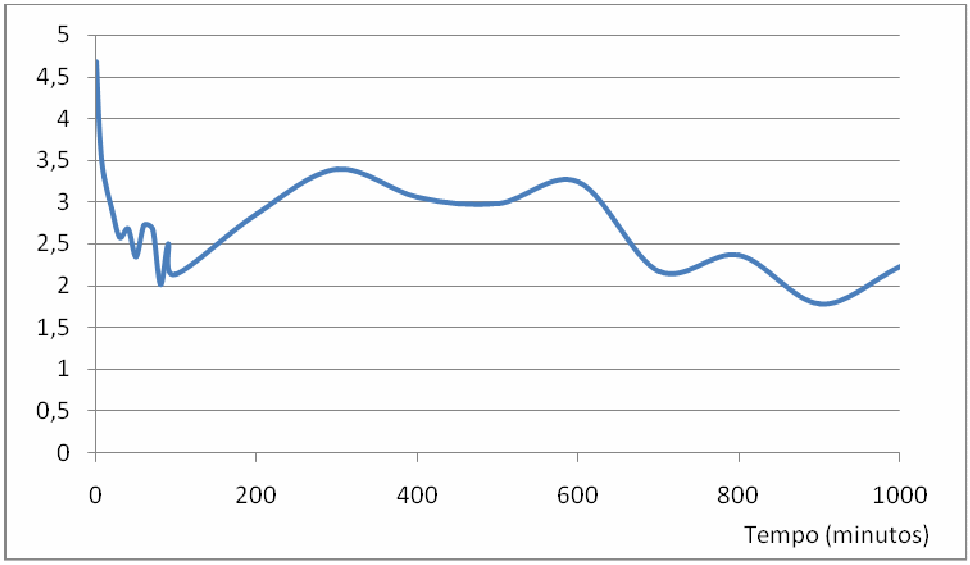


Fig. 23 – Ganho percentual do sistema de Codificação de Rede a nível largura banda média para *download* em função do número de *seeds*.

Atendendo à largura de banda média utilizada por cada nó, o cenário espelha o que se verifica quanto ao tempo médio de *download*.

4.5 Discussão

Em suma, foi possível constatar que a utilização de Codificação de Rede introduz, de uma forma geral, ganhos de eficiência face a um sistema BitTorrent. Porém, estes ganhos não são uniformes, sendo mais relevantes quanto mais elevada for a largura de banda média dos nós participantes no *torrent* e menor for o tempo de permanência de *seeds* no mesmo.

Para melhor analisar a eficiência da Codificação de Rede aplicada numa rede *peer-to-peer* de distribuição de conteúdos como o BitTorrent torna-se ainda necessário entrar em linha de conta com o tempo e complexidade da reconstituição do conteúdo obtido. Num sistema BitTorrent, ao serem obtidos todos os pedaços do conteúdo original, este encontra-se imediatamente reconstituído. Já num sistema com Codificação de Rede, o mesmo não acontece: os pedaços obtidos são combinações lineares de blocos do conteúdo original, pelo que estes devem ser reconstituídos numa operação semelhante à resolução de um sistema de equações lineares. Embora este processo seja local a cada utilizador, não afectando consequentemente aquele que será provavelmente o maior *bottleneck* numa rede de distribuição de conteúdos – os recursos de largura de banda disponibilizados – e sendo irrelevante para os aspectos abrangidos neste trabalho (tempo de *download* e proporção de largura de banda média alocada para a obtenção de conteúdos), implica a introdução de um maior esforço computacional o que, por seu turno, implica o gasto de mais tempo. De acordo com testes efectuados a uma implementação de um sistema *peer-to-peer* de codificação de rede[GK05] numa máquina típica (Pentium IV 2GHz e 512MB de memória RAM), a utilização de CPU aumenta de 20% (mantido ao longo do *download* do conteúdo) para 40%. Por outro lado, tendo em conta uma ligação com 2.2Mbps de capacidade de *download*, o tempo gasto para a reconstituição do conteúdo cifra-se em 6% das 9

horas dispendidas para obtenção de um ficheiro de 2,8GB. Por outro lado, o tempo de descodificação pode ser reduzido utilizando descodificação *on-the-fly* e usando amortização[GR05]: desta forma, a descodificação é feita ao longo do *download*, sendo o esforço necessário para o processo amortizado. Sendo este um aspecto com espaço para melhoramento num sistema de Codificação de Rede, não deve ser utilizado para por em causa os ganhos de eficiência introduzidos pelo mesmo a nível da utilização de recursos da rede.

Um outro aspecto que pode condicionar a prestação de um sistema de Codificação de Rede é a existência da possibilidade de obtenção de blocos linearmente independentes. Num sistema em que não exista qualquer medida para evitar a possibilidade de serem obtidos dois blocos linearmente dependentes, este factor pode influenciar fortemente a prestação do sistema ao ponto de ser menos eficiente do que uma transferência estritamente entre dois nós (como é característico de grande parte dos sistemas *peer-to-peer* para transferência de conteúdos)[WL06]. Existem, no entanto, estratégias que permitem reduzir a probabilidade de obtenção de pedaços linearmente dependentes, tais como o *Randomized Network Coding*[HMS+03] que, de uma forma sumária, cada nó recorre a informação acerca dos vectores de codificação utilizados pelos seus vizinhos para reduzir a possibilidade de gerar pedaços redundantes. Desta forma, a probabilidade de gerar pedaços redundantes é drasticamente reduzida[MXLX06], mas ainda diferente de zero. Torna-se então necessário definir um método para detectar e resolver estas situações, seja este *on-the-fly* (provavelmente a par de um método igualmente *on-the-fly* de reconstituição do conteúdo original) ou no final (aquando da descodificação total do conteúdo) para que sejam obtidos mais blocos, linearmente independentes, que permitam descodificar o conteúdo distribuído.

Existe ainda um aspecto particular que reduz a usabilidade deste sistema face ao BitTorrent: enquanto que ao utilizar BitTorrent um qualquer utilizador pode obter apenas parte do conteúdo, num sistema de Codificação de Rede o conteúdo necessita de ser obtido por inteiro. Esta funcionalidade pode ser útil, por exemplo, em situações em que o conteúdo a ser distribuído é composto por vários sub-conteúdos que possam ser utilizados de forma independente.

Por último, a complexidade do código inerente aos clientes de ambos os tipos de sistemas é também um factor que condiciona os benefícios da Codificação de Rede. Um sistema de Codificação de Rede apresenta, naturalmente, código mais complexo: necessita de contemplar a codificação e descodificação de blocos, operações à partida não-triviais. A introdução de uma maior complexidade no código acarreta consigo uma maior probabilidade de ocorrência de erros, um factor que afecta negativamente a concepção e posterior utilização de um sistema de Codificação de Rede.

5 CONCLUSÕES

5.1 Estudo acerca da aplicação de uma CDN centralizada

A primeira parte do trabalho efectuado no âmbito desta dissertação constituiu na integração de um sistema de *Corporate TV*, da Mobbit Systems, com uma rede proprietária de distribuição de conteúdos, da Cisco Systems. Esta integração visava resolver um problema de uso ineficiente de recursos de largura de banda na WAN: o sistema padecia do facto de que, caso dois terminais geograficamente próximos necessitassem de obter os mesmos conteúdos, estes não cooperariam entre si, mas obteriam os conteúdos separadamente a partir do repositório, situado num local arbitrariamente remoto. Consequentemente, teríamos o mesmo conteúdo a ser transmitido mais do que uma vez sobre a ligação WAN. A abordagem proposta para solucionar esta situação, que passou pela integração do sistema Cisco ACNS, conferiu uma utilização mais eficiente dos recursos de largura de banda, promovendo o *caching* de conteúdos (para que não sejam obtidos mais do que uma vez junto do repositório central de dados) e a cooperação entre servidores do sistema (denominados *cache engines*).

Os ganhos resultantes desta integração são claros, mas não lineares: estes dependem dos conteúdos a serem disseminados, do número e localizações relativas dos terminais e dos alinhamentos de conteúdos definidos para cada terminal.

Embora este projecto tivesse sido essencialmente prático, a fatia mais significativa de tempo empreendido no mesmo prendeu-se com uma análise dos sistemas em questão, não na concepção do módulo para integração dos sistemas propriamente dito.

5.2 Aplicação de Codificação de Rede a CDNs *Peer-to-Peer*

A segunda parte do trabalho consistiu num estudo do ganho introduzido com a aplicação de Codificação Linear de Rede a um sistema *peer-to-peer* para a distribuição de conteúdos, nomeadamente o BitTorrent. Para o efeito, foram efectuadas várias simulações de ambos os sistemas (recorrendo a um simulador de BitTorrent desenvolvido pela Microsoft Research, o OctoSim[MR03], e a uma adaptação deste que contempla a utilização de Codificação de Rede e foi especificamente concebida para esta dissertação), em diferentes cenários de teste previamente estipulados, de forma a determinar as suas prestações comparativas.

Embora a utilização de Codificação de Rede se traduza, de uma forma geral, numa melhoria da prestação do sistema, tanto em tempo de *download* como em largura de banda utilizada para *download*, esta é especialmente relevante em casos específicos: quando as larguras de banda dos nós participantes são mais elevadas e quando a proporção de *seeds* no *swarm* é reduzida.

Devem, no entanto, ser considerados um conjunto de factores para além da eficiência do sistema, desde a possibilidade de geração de informação redundante até à maior complexidade introduzida por este método.

REFERÊNCIAS

[ACLY00] R. Ahlswede, N. Cai, S.R. Li e R.W. Yeung. Network Information Flow. *IEEE Transactions of Information Theory*, Vol. 46, N.º 4, Julho 2000.

[Adl99] S. Adler. The Slashdot Effect, An Analysis of Three Internet Publications. *The Linux Gazette*, N.º 38, Março 1999.

[AHM+03] I. Ari, B. Hong, E.L. Miller, S.A. Brandt e D.D. Long. Managing Flash Crowds on the Internet. Em *Proceedings of the 11th IEEE/ACM International Symposium on MASCOTS*, Outubro 2003.

[ASBL99] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan e J. Lilley. The design and implementation of a intentional naming system. Em *Proceedings of the 17th ACM symposium on Operating systems principles*, pp. 186-201, 1999.

[BAZ+97] S. Bhattacharjee, M.H. Ammar, E.W. Zegura, V. Shah e Z. Fei. Application level anycasting. Em *Proceedings of the 16th annual joint conference of the IEEE Computer and Communication Socities*, Vol. 3, pp. 1388-1396, Abril 1997.

[BDK+02] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham e P. Shenoy. Adaptive push-pull: disseminating dynamic Web data. Em *Proceedings of the 10th international conference on World Wide Web*, pp. 265-274, 2001.

[BH05] A.R. Bharambme e C. Herley. Analyzing and Improving BitTorrent Performance, Microsoft Research Report No. MSR-TR-2005-03, Fevereiro 2005.

[BHP06] A.R. Bharambe, C. Herley e V.N. Padmanabhan. Analyzing and Improving a BitTorrent Network's Performance Mechanisms. Em *Proceedings of the 25th IEEE International Conference on Computer Communications (Infocom'06)*, pp. 1-12, Abril 2006.

[Bog82] D.R. Boggs. Internet broadcasting. *PhD Thesis, Stanford University*, 1982.

[CDK+03] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron e A. Singh. Em *Proceedings of the 19th ACM symposium on Operating systems principles*, pp. 298-313, 2003.

[CDKR02] M. Castro, P. Druschel, A.M. Kermarrec e A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. Em *IEEE Journal on Selected Areas in communications*, 2002.

[Cis03] Cisco ACNS Software Deployment and Configuration Guide, Release 5.1, Cisco Systems Inc., 2003.

[Cis04a] Cisco ACNS Software Version 5.1, Cisco Systems Inc., 2004.

[Cis04b] Cisco ACNS Software Configuration Guide for Centrally Managed Deployments, Release 5.2, Cisco Systems Inc., 2004.

[Cis04c] Cisco ACNS Software Configuration Guide for Locally Managed Deployments, Release

5.2, Cisco Systems Inc., 2004.

[Cis04d] Cisco ACNS Solution for Software Distribution Best Practices, Cisco Systems Inc., 2004.

[Cis04e] Media Publisher and Cisco Application and Content Networking System (ACNS) Software 5.2, Cisco Systems Inc., 2004.

[CBB+02] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord e J. Stafford, *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Longman, 1ª Edição, Setembro 2002.

[CKK01] Y. Chen, R.H. Katz e J.D. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. *Peer-to-Peer Systems: First International Workshop, IPTPS*, 2002

[CO01] K.G. Coffman e A.M. Odlyzko. Internet Growth: Is there a "Moore's Law" for data traffic?. AT&T Labs – Research, Junho 2001.

[Cod01] CIFS Explained. CodeFX software solutions for applications and appliances, 2001.

[Coh03] B. Cohen. Incentives Build Robustness in BitTorrent. Em *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, Junho 2003.

[CWJ03] P.A. Chou, Y. Wu, K. Jain. Practical Network Coding. Em *Proceecings of the 51st Allerton Conference on Communication, Control and Computing*, Outubro 2003.

[DE07] C. Barreto. PT lança amanhã serviço IPTV no qual já investiu 10 milhões de euros. Em *Diário Económico*, Edição de 13 de Junho de 2007.

[DGLZ07] R.J. Dunn, S.D. Gribble, H.M. Levy e J. Zahorjan. The Importance of History in a Media Delivery System. Em *Proceedings of the 6th International Workshop on Peer-to-Peer Systems*, Fevereiro 2007.

[DL07] C. Dale e J. Liu. A Measurement Study of Piece Population in BitTorrent. Em *IEEE Global Communications Conference (Infocom'07)*, Novembro 2007.

[DLGI99] X. Deng, B. Li, M.J. Golin e G.F. Italiano. On the optimal placement of Web proxies on the Internet. Em *Proceedings of the IEEE Global Communications Conference (Infocom'99)*, Vol. 3, pp. 1282-1290, Março 1999.

[DMP+02] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman e B. Weihl. Global Distributed Content Delivery. Em *IEEE Internet Computing*, Setembro 2002.

[DNB03] C.N. Ding, S. Nutanong e R. Buyya. P2P Networks for Content Sharing. *Techical Report, GRIDS-TR-2003-07 – Grid Computing and Distributed Systems Laboratory at the University of Melbourne*, Dezembro 2003.

[EP05] D. Erman e A. Popescu. BitTorrent Request Message Models. Em *Proceedings of The Third Swedish National Computer Networking Workshop*, Março 2005.

[Erm05] D. Erman. BitTorrent Traffic Measurements and Models. *School of Engineering at the Blekinge Institute of Technology in Karlskrona, Sweden, Licentiate Dissertation Series*

No. 1650-2140, 2005:13, Outubro 2005.

[FBW06] C. Fragouli, J.L. Boudec e J. Widmer. Network Coding: An Instant Primer. Em *ACM SIGCOMM Computer Communication Review*, Vol. 36, N.º 1, pp. 63-68, Janeiro 2006.

[FC05] M. Feldman e J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems, Em *ACM SIGecom Exchanges*, Vol. 5, N.º 4, pp. 41-50, Julho 2005.

[Fri06] T.L. Friedman: *The World is Flat: A Brief History of the Twenty-first Century (Updated and Expanded Edition)*, Farrar Straus Giroux, Abril 2006.

[GCX+05] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding e X. Zhang. Measurements, Analysis and Modeling of BitTorrent-like Systems, Em *Proceedings of the ACM/SIGCOMM Internet Measurement Conference*, 2005.

[GJJ+00] D.K. Gifford, J. Janotti, K.L. Johnson, M.F. Kaashoek, J.W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network, Em *Proceedings of the 4th Symposium on Operating System Design and Implementation*, pp. 197-212, Outubro 2000.

[GR05] C. Gkantsidis e P.R. Rodriguez. Network Coding for Large Scale Content Distribution, Em *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 4, pp. 2235-2245, 2005.

[Gru07] IPTV – O mundo da IPTV: Grupo 6, Comunicação de Áudio e Vídeo, 2007.

http://www.img.lx.it.pt/~fp/cav/ano2006_2007/MERC/Trab_6/omundo.html

[GS95] J.D. Guyton e M.F. Schwartz. Locating nearby copies of replicated Internet servers. Em *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 288-298, 1995.

[HIM04] B. Hildreth, G. Iorio e A. Molet. Napster. Copyright Infringement or Sharing of Information. *Academic Paper from the School of Information Studies at the Syracuse University*, Dezembro 2004.

[HMS+03] T. Ho, M. Médard, J. Shi, M. Effros e D.R. Karger. On Randomized Network Coding, Em *Proceedings of The Annual Allerton Conference on Communication Control and Computing*, Vol. 41, Parte 1, pp. 11-20, 2003.

[Int] Internet World Stats – World Internet Usage and Population Statistics, Junho 2007. http://www.internetworldstats.com/stats.htm.

[IUB+04] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra e L. Garces-Erice. Dissecting BitTorrent: Five Month in a Torrent's Lifetime. Em *Proceedings of the 5h Passive and Active Measurement Workshop*, Abril 2004.

[JJJ+00] S. Jasmin, C Jin, Y. Jin, D. Raz, Y. Shavitt e L. Zhang. On the placement of Internet instrumentation. Em *Proceedings of the IEEE the IEEE Global Communications Conference (Infocom'02)*, Março 2002.

[KAGL01] J. Korkhonen, O. Aalto, A. Gurtov e H. Lamanen. Measured Performance of GSM, HSCSD and GPRS. Em *Proceedings of the IEEE International Conference on*

Communications, Vol. 5, pp. 1330-1334, 2001.

[KB94] N. Krishnakumar e A.J. Bernstein. Bounded ignorance: a technique for increasing concurrency in a replicated system. *ACM Transactions on Database Systems (TODS)*, Vol. 19, N.º 4, pp. 586-625, Dezembro 1994.

[KM03] R. Koetter e M. Médard. An Algebraic Approach to Network Coding. Em *IEEE/ACM Transactions on Networking*, Vol. 11, N.º 5, pp. 782-795, Outubro 2003.

[KPS00] P. Krishnan, D. Raz e Y. Shavitt. The cache location problem. Em *IEEE/ACM Transactions on Networking*, Vol. 8, N.º 5, pp. 568-582, Outubro 2000.

[KRAV02] D. Kostic, A. Rodriguez, J. Albrecht e A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. Em *ACM SIGOPS Operating Systems Review*, Vol. 37, N.º 5, pp. 282-297, Dezembro 2003.

[KRR01] J. Kangasharju, J. Roberts e K.W. Ross. Object Replication Strategies in Content Distribution Network. Em *Proceedings of the 6th International Workshop on Web Caching and Content Distribution*, 2001

[KW01] D. Katabi e J. Wroclawski. A framework for scalable global-IP anycast (GIA). Em *Workshop on data communication in Latin America and the Caribbean*, Vol. 31, N.º 2, pp. 186-219, Abril 2001.

[LCC02] Q. Lv, P. Cao, E. Cohen, K. Li e S. Shenker. Search and replication in unstructured peer-to-peer network. Em *Proceedings of the 16th international conference on Supercomputing*, pp. 84-95, 2002.

[LNKZ05] N. Liogkas, R. Nelson, E. Kohler e L. Zhang: Exploiting BitTorrent for fun (but not for profit), Em *Proceedings of the 5th International Workshop on Peer-to-Peer Systems*, Fevereiro 2005.

[LUM06] A. Legout, G. Urvoy-Keller e P. Michiard, Rarest First and Choke Algorithms Are Enough. Em *Proceedings of the 6th ACM SIGCOMM on Internet Measurement*, pp. 203-216, 2006.

[LYC03] S.R. Li, R.W. Yeung e N. Cai. *Linear Network Coding*. *IEEE Transactions on Information Theory*, Vol. 49, N.º 2, pp. 371-381, Fevereiro 2003

[Mar] mariposaHD.tv – The World's First HDTV Show for the Internet

http://www.mariposahd.tv/

[Men01] R. Menta: Bertlesmann Chairman Threatens to Close Napster in 15 weeks. *mp3newswire.net*, Março 2001. http://www.mp3newswire.net

[Mic07] Microsoft Mediaroom, Microsoft, 2007

http://www.microsoft.com/tv/Products.mspx

[Moba] Sistema InSight Versão 2.0 – Datasheet, Mobbit Systems.

[Mobb] Sistema InSight Versão 2.0 – Descrição, Mobbit Systems.

[Mobc] Sistema InSight Versão 2.0 – Especificação, Mobbit Systems.

[Mobd] XML Sistema G-Mobbit 2.0 (G-20), Mobbit Systems.

[MR03] OctoSim – BitTorrent Peer-toPeer Protocol Simulator, Microsoft Research, Dezembro 2003.

http://research.microsoft.com/research/downloads/Details/20d68689-9a8d-44c0-80cd-66dfa4b0504b/Details.aspx.

[MRG06] IPTV Global Forecast – 2006 to 2010, Multimedia Research Group, Inc., Outubro 2006.

http://www.mrgco.com/TOC_GF1006.html

[MXLX06] G. Ma, Y. Xu, M. Lin e Y. Xuan. A Content Distribution System based on Sparse Linear Network Coding. *Department of Computer Science and Technology at the University of Science and Technology of China*, 2006.

[Nun06] M.S. Nunes. Redes de Acesso. *Folhas de Apoio à Disciplina de Redes de Acesso, Instituto Superior Técnico (LECI)*, Setembro 2006.

[Pen03] G. Peng: CDN: Content Distribution Network, *Tecnhical Report TR-125 of the State University of New York,* Janeiro 2003.

[PIA+07] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy e A. Venkataramani. Do incentives build robustness in BitTorrent?. Em *Proceedings of the 4th USENIX Symposium on Networked Systems Design & Implementation*, pp. 1-14, Abril 2007.

[PMM93] C. Partridge, T. Mendez e W. Milliken. Host Anycasting Service. *IETF RFC 1546*, 1993

[PRR97] C.G. Plaxton, R. Rajaraman e A.W. Richa. Accessingnearby copies of replicated objects in a distributed environment. Em *ACM Symposium on Parallel Algorithms and Architectures*, pp. 311-320, 1997.

[PS06] G. Pierre e M. van Steen. Globule: a Collaborative Content Delivery Network. *IEEE Communications Magazine*, Vol. 44, N.º 8, pp. 127-133, Agosto 2006.

[PV06] G. Pallis e A. Vakali. Insight and Perspectives for Content Delivery Networks. *Communications of the ACM*, Vol. 49, N.º 1, pp. 101-106, Janeiro 2006.

[QPV01] L. Qiu, V.N. Padmanabhan e G.M. Voelker. On the placement of Web server replicas. Em *Proceedings of the IEEE the IEEE Global Communications Conference (Infocom'01)*, Vol. 3, pp. 1587-1596, 2001.

[Rac04] S. Racine. Analysis of Internet Relay Chat Usage by DDoS Zombies, *Tese de Mestrado MA-2004-01*, Institut fur Technische Informatik und Kommunikationsnetze da Eidgenossiche Technische Hochshule Zurich, Abril 2004.

[RD01] A. Rowstron e P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. Em *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*, Novembro 2001

[Rei06] J. Reimer. Yahoo Messenger and Windows Live Messenger get together, Em Ars

Technica, Setembro 2006.

http://arstechnica.com/news.ars/post/20060927-7846.html

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp e S. Schenker. A scalable content-addressable network. Em *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 161-172, 2001.

[RJCE06] A. Ramamoorthy, K. Jain, P.A. Chou e M. Effros. Separating Distributed Source Coding from Network Coding, *IEEE Transactions on Information Theory*, Vol. 52, N.º 6, pp. 2785-2795, Junho 2006.

[Rod06] L.M. Rodrigues. IPTV: Conceitos, Padrões e Soluções, *Monografias em Ciência e Computação*, n.º 05/06, ISSN 0103-9741, Fevereiro 2006.

[RS02] M. Rabinovich e O. Spatscheck: *Web Caching and Replication*, Addison Wesley, 1ª edição, 2002.

[SET03] P. Sanders, S. Egner e L. Tolhuizen: Polynomial Time Algorithms for Network Information Flow, Em *Proceedings of the 15th ACM Symposium on Parallel Algorithms and Architectures*, pp. 286-294, 2003.

[SGD+02] S. Saroiu, P.K. Gummadi, R.J. Dunn, S.D. Gribble e H.M. Levy: An Analysis of Internet Content Delivery Systems, Em *Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, 2002.

[SGG02] S. Saroiu, P.K. Gummadi e S.D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. Em *Proceedings of Multimedia Computing and Networking*, 2002.

[SL04] E. Shiu e A. Lenhart. How Americans Use Instant Messaging. Em *Pew Internet & American Life Project*, Setembro 2004.

[SMB02] T. Stading, P. Maniatis e M. Baker. Peer-to-Peer Caching Schemes to Address Flash Crowds. Em *Proceedings of the 1st International Peer To Peer Systems Workshop*, 2002.

[SMK+01] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek e H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. Em *Proceedings of the 2001 conference on Applications, technologies, architectures and protocols for computer communications*, pp. 146-160, 2001.

[SSPS04] S. Sivasubramanian, M. Szymaniak, G. Pierre e M. van Steen. *ACM Computing Surveys*, Vol. 36, N.º 3, pp. 291-334, Setembro 2004.

[TAR99] F. Torres-Rojas, M. Ahamad e M. Raynal. Timed consistency for shared distributed objects. Em *Proceedings of 18th annual ACM symposium on Principles of distributed computing (PODC'99)*, Maio 1999.

[VP03] A. Vakall e G. Pallis. Content Delivery Networks: Status and Trends. Em *IEEE Internet Computing*, Vol. 7, N. º 6, pp. 68-74, Novembro 2003.

[Wet02] R. Wetzel. CDN Business Models – The Drama Continues. Em *Business Communication Review*, Abril 2002.

[WL06] M. Wang e B. Li. How Practical is Network Coding?. Em *Proceedings of the 14th IEEE International Workshop on Quality of Service*, pp. 274-278, Junho 2006

[XY07] H. Xie e Y.R. Yang. A Measurement-Based Study of the Skype Peer-to-Peer VoIP Performance. Em *International Workshop on Peer-to-Peer Systems*, 2007

[YV02] H. Yu e A. Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems (TOCS)*, Vol. 30, N.º 3, pp. 239-282, Agosto 2002.

[ZKJ01] B.Y. Zhao, J. Kubiatowicz e A.D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. *Tecnhical Report CSD-01-1141 of the University of Berkley*, 2001

Página intencionalmente deixada em branco.

INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Redes de Distribuição de Conteúdos

Integração com IPTV e Estudo sobre Codificação de Rede

Rui Dinis Mangueira Bento

ANEXO

Estrutura dos Ficheiros XML de Configuração do Sistema Mobbit InSight 2.0

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Setembro, 2007

ÍNDICE

1. INTRODUÇÃO

Toda a informação essencial à gestão da distribuição de conteúdos no sistema, nomeadamente relativa a clientes, terminais, blocos, alinhamentos, grelhas, *playlists* e *templates*, é parametrizada em ficheiros, segundo o formato XML. Nesta secção serão abordadas as estruturas dos diferentes tipos de ficheiros utilizados.

2. FICHEIROS CLIENTES.XML

O ficheiro **clientes.xml** contem a informação associada aos clientes do sistema e, para cada um dos clientes, os respectivos canais (e *frames* de cada canal), blocos de conteúdos e terminais associados. A sua estrutura é a seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<clientes>
	<cliente>
		<cliente_id></cliente_id>
		<nome></nome>
		<canais>
			<canal_id></canal_id>
			<nome></nome>
			<inicio_downloads></inicio_downloads>
			<fim_downloads></fim_downloads>
			<status></status>
			<frames>
				<frame_id></frame_id>
				<nome></nome>
				<altura></altura>
				<largura></largura>
				<posicao></posicao>
			</frames>
		</canais>
		<blocos>
			<bloco_id></bloco_id>
			<nome></nome>
			<tipo></tipo>
			<carregamentos></carregamentos>
		</blocos>
		<terminais>
			<terminal_id></terminal_id>
			<nome></nome>
		</terminais>
	</cliente>
</clientes>
```

3. FICHEIRO TERMINAIS.XML

O ficheiro **terminais.xml** contém toda a informação associada à configuração de cada terminal. Consequentemente, sempre que um terminal for adicionado, editado ou removido do sistema, este ficheiro deverá ser gerado. A sua estrutura é a que se segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<terminais>
	<terminal>
		<terminal_id></terminal_id>
		<nome></nome>
		<tipo></tipo>
		<gateway_id></gateway_id>
		<nivel_som></nivel_som>
		<estado></estado>
		<intervalo_ligacoes></intervalo_ligacoes>
		<endereco_IP></endereco_IP>
		<gateway_IP></gateway_IP>
	</terminal>
</terminais>
```

4. FICHEIRO BLOCOS.XML

O ficheiro **blocos.xml** contém a informação relativa às entradas dos vários blocos. A sua estrutura é a seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<alinhamentos>
	<alinhamento>
		<alinhamento_id></alinhamento_id>
		<nome></nome>
		<entradas>
			<posicao></posicao>
			<bloco_id></bloco_id>
			<entrada_id></entrada_id>
		</entradas>
		<terminais>
			<terminal_id></terminal_id>
			<nome></nome>
		</terminais>
	</alinhamento>
</alinhamentos>
```

5. FICHEIRO ALINHAMENTOS.XML

O ficheiro **alinhamentos.xml** deve conter a informação associada aos vários alinhamentos, nomeadamente, para cada alinhamento, os seus conteúdos (como referências a entradas de blocos) e os terminais a que se destinam. Por conseguinte, a cada inserção, edição ou remoção de um alinhamento, o ficheiro deve ser novamente gerado. A estrutura do ficheiro é a seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<alinhamentos>
	<alinhamento>
		<alinhamento_id></alinhamento_id>
		<nome></nome>
		<entradas>
			<posicao></posicao>
			<bloco_id></bloco_id>
			<entrada_id></entrada_id>
		</entradas>
		<terminais>
			<terminal_id></terminal_id>
			<nome></nome>
		</terminais>
	</alinhamento>
</alinhamentos>
```

6. FICHEIRO GRELHAS.XML

O ficheiro **grelhas.xml** deve conter toda a informação associada à configuração das várias grelhas de programação. Assim sendo, a inserção, edição ou remoção de uma grelha implica a geração deste ficheiro. A sua estrutura é a que se segue:

```
<?xml version="1.0" encoding="UTF-8"?>
<grelhas>
	<grelha>
		<grelha_id></grelha_id>
		<nome></nome>
		<frame_id></frame_id>
		<data_inicio></data_inicio>
		<data_fim></data_fim>
		<hora_inicio></hora_inicio>
		<hora_fim></hora_fim>
		<entradas>
			<posicao></posicao>
			<alinhamento_id></alinhamento_id>
			<nome></nome>
		</entradas>
		<terminais>
			<terminal_id></terminal_id>
			<nome></nome>
		</terminais>
	</grelha>
</grelhas>
```

7. FICHEIRO PLAYLIST.XML

De forma a que os conteúdos operem adequadamente, estes devem ter acesso à informação sobre quais os conteúdos a exibir, quando exibi-los e como dispô-los no ecrã. Toda essa informação é compilada em ficheiros denominados *playlists*. Por conseguinte, para cada canal, é gerado o ficheiro **playlist_X.xml**, em que X é o identificador alfanumérico do canal, com toda a informação de que este necessita para o seu correcto funcionamento. A estrutura do ficheiro é definida da seguinte forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist>
	<frame>
		<grelha>
			<alinhamento>
				<entrada></entrada>
			</alinhamento>
		</grelha>
	</frame>
</playlist>
```

8. FICHEIRO TEMPLATE.XML

Como previamente indicado, os ficheiros disseminados pelo sistema InSight® podem ser *templates*, que correspondem a aplicações *Flash* parametrizáveis. De forma a que a aplicação tenha informação acerca dos parâmetros a utilizar, bem como os terminais de forma a obtê-los junto do servidor central, existe um ficheiro com essa informação. O ficheiro em questão denomina-se **template_Y.xml**, em que Y consiste no identificador alfanumérico do *template*. A sua estrutura é a seguinte:

```
<?xml version="1.0" encoding="UTF-8"?>
<template>
      <ficheiro></ficheiro>
</template>
```