DEPARTAMENTO DE CIÊNCIAS ECONÓMICAS
EMPRESARIAS E TECNOLÓGICAS

Dissertação de Mestrado em
Engenharia e Tecnologias Informáticas

Uma Arquitetura de Transferência de Ficheiros Otimizada a Redes Sociais

Autor: Marco André do Vale Pires Brotas
Orientador: Professor Doutor Mário Marques da Silva

Lisboa, Maio de 2013

ÍNDICE

1 - INTRODUÇÃO

1.1 Objetivos

Pretende-se com esta dissertação de mestrado, propor uma arquitetura de rede computacional que integre as redes sociais da *web* com o protocolo de transferência de ficheiros P2P (Peer-to-peer) BitTorrent. O objetivo consiste em unir o melhor destas duas tecnologias, a componente social e a capacidade avançada de transferir ficheiros entre os vários computadores dos utilizadores, fornecendo uma forma mais eficiente de transferência de ficheiros aos utilizadores de *sites* redes sociais e um maior número de adesão à arquitetura BitTorrent.

Pretende-se demonstrar que esta inovação que resulta da unificação destas duas tecnologias representa uma mais-valia, proporcionando as seguintes vantagens:

- Introduzir formas intuitivas que facilitem a transferência e gestão de ficheiros entre os vários utilizadores em P2P;
- Obter o máximo proveito da largura de banda na transferência de ficheiros;
- Aumentar a base de utilizadores que tiram proveito de protocolos P2P ou BitTorrent. No segundo caso, aumentando o número de sementes[1];
- Alcançar uma maior integração social dos ficheiros a serem partilhados (comentários, classificações, qualidade, etc…);
- Facilitar a partilha de ficheiros de grande porte entre os utilizadores de redes sociais, sem ser necessário recorrer a outras ferramentas[2];
- Reduzir custos de equipamentos necessários para efetuar a gestão do sistema, o qual resulta do facto dos computadores dos utilizadores funcionarem simultaneamente como servidores de arquivo.

Todas estas vantagens resultam do facto de esta dissertação apresentar a resolução de alguns desafios arquitetónicos das estruturas de redes sociais, favorecendo também o desempenho do protocolo BitTorrent.

Com o crescente aumento do número de utilizadores dos *sites* de redes sociais, correntemente estimado em mais de 1 bilião [Tweetsmarter 2012] [InternetWorldStats 2012], relativo à utilização do sistema P2P BitTorrent, correntemente estimado em cerca de 150 milhões [BitTorrent 2012], perspetiva-se que a junção dos dois proporcione um aumento significativo do número de utilizadores da rede BitTorrent e da partilha de ficheiros neste protocolo.

A utilização massiva da arquitetura proposta poderá conduzir a um melhor aproveitamento da largura de banda disponível, um aumento do tempo de vida médio dos ficheiros na rede, bem como uma possível melhoria no desempenho geral do sistema P2P.

[1] Semente é o termo usado para representar o computador de um utilizador que está disponível para fornecer partes dos ficheiros a transferir.

[2] Muitos utilizadores da Internet não recorrem a ferramentas de transferência de ficheiros baseado em P2P devido à sua complexidade, falta de intuitividade ou simplesmente porque não as conhecem ou não querem correr riscos.

1.2 Trabalho Anterior

O trabalho produzido nesta dissertação pode ser considerado uma continuação ou adição ao protocolo BitTorrent [Cohen 2003] e à plataforma corrente da maioria dos *sites* de redes sociais.

O protocolo BitTorrent é resultante de uma arquitetura semi-centralizada P2P que pretende facilitar ou proporcionar uma forma mais eficiente de transferência de ficheiros entre os utilizadores. No entanto, foram identificadas as seguintes limitações que pretendem ser ultrapassadas:

- Falta de intuitividade de utilização para quem tem pouca experiencia em informática.
- Falta de integração social. Por exemplo, muitas vezes são partilhados ficheiros maliciosos ou falsos. Situação que poderia ser consideravelmente reduzida se estes tivessem uma maior integração social. Alguns *sites* de partilha de ficheiros de identificação ".torrent" tentam resolver este problema, mas continua a existir bastante mais espaço para melhoramentos.
- Em alguns casos, devido à falta de *peers* para ficheiros mais raros, pode acontecer que certas partes ou um todo desapareça da rede permanentemente.
- Gestão dos ficheiros carregados. Neste momento existe pouco controlo sobre os ficheiros que foram carregados na rede BitTorrent.

Os *sites* de redes sociais já passaram por muitas alterações e inovações ao longo dos anos, muitos melhoramentos foram implementados e existem até bastantes *sites* com funcionalidades exclusivas. No entanto continua a haver uma limitação que obriga os utilizadores a recorrer a ferramentas adicionais para fazer o trabalho, a partilha de ficheiros. A verdade é que a partilha de ficheiros em *sites* de redes sociais, se estiver disponível, continua a seguir a forma tradicional de transferência Cliente-Servidor. Esta é uma

importante limitação, visto que é muito frequente que certos utilizadores tenham a necessidade de partilhar ficheiros com os seus contactos, e esta transferência é efetuada de forma pouco eficiente.

1.3 Tecnologias Envolvidas

Esta seção efetua uma breve descrição das tecnologias, cuja integração é o objetivo desta dissertação.

1.3.1 BitTorrent

BitTorrent é o protocolo de P2P semi-centralizado[3] mais usado hoje em dia. Permite maximizar a velocidade de transferência, juntando as partes do ficheiro que se pretende adquirir de vários computadores e procedendo ao descarregamento das mesmas em simultâneo.

Este processo aumenta a popularidade de ficheiros de grande porte, devido à velocidade de transferência ser geralmente bastante mais elevada do que a de outros protocolos.

Para uma melhor compreensão do funcionamento do protocolo BitTorrent e a justificação do facto de a partilha de ficheiros em P2P ser a mais usada atualmente, as próximas subsecções efetuam a descrição da sua evolução.

1.3.1.1 Forma Tradicional de Descarga Cliente-Servidor

A forma tradicional de transferência de dados consiste numa arquitetura Cliente-Servidor. Esta baseia-se na criação de uma ligação entre um computador cliente (que pretende adquirir um recurso), um computador servidor (que contem e está pronto a servir o recurso em causa) e uma ou mais transações entre eles. Existem algumas variantes dependendo do protocolo que está a ser usado, como *FTP* (*File Transfer Protocol*), *HTTP* (*HyperText Transfer Protocol*), etc...

Após a troca de dados correspondentes ao protocolo de estabelecimento de uma ligação de transporte baseada no protocolo TCP (*Transmission Control Protocol*), entre o cliente e o servidor, dá-se início à transferência do ficheiro utilizando o protocolo FTP (*File Transfer Protocol*) que funciona ao nível da camada aplicacional do modelo TCP/IP [Marques da Silva 2012] [Kozierok 2005]. Os bits percorrem o seu caminho do servidor até ao cliente, passando por vários nós intermédios (routers), até a transferência terminar. Em certos casos, dependendo do servidor e do *software* que está a ser usado, também é permitido pausar/continuar a transferência.

No entanto, existem um número de variáveis que afetam a velocidade de transferência:

- A quantidade de trafego no servidor;
- O tráfego existente nos nós intermédios, entre o servidor e o cliente.

Naturalmente que ficheiros de grandes dimensões e com elevada popularidade serão alvo de uma procura mais elevada, o que resultará numa velocidade de transferência mais reduzida. A ilustração 1 ilustra esta arquitetura, com um exemplo da descrita limitação.

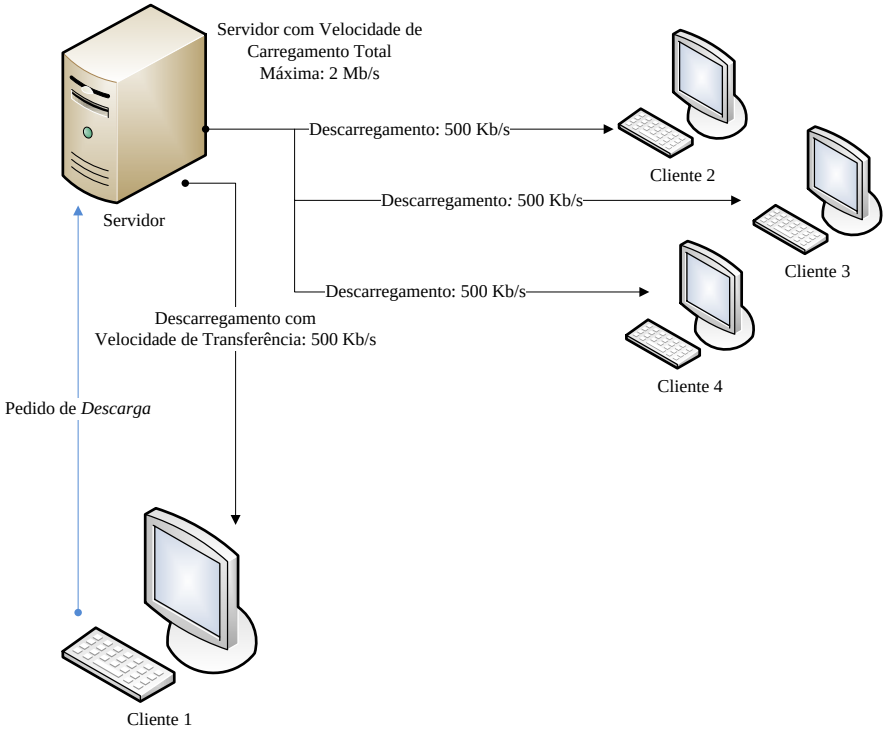

Ilustração 1 - Exemplo de uma limitação de transferência do tipo Cliente-Servidor.

1.3.1.2 Partilha de ficheiros P2P com uma arquitetura descentralizada

A partilha de ficheiro em P2P [Melville et al. 2002] [Acosta and Chandra 2005] [Pouwelse et al. 2004] é diferente da tradicional. Neste caso é usado uma aplicação específica para localizar outros computadores que contêm o ficheiro pretendido. Visto que estes são computadores de utilizadores normais, em vez de servidores, são chamados de *peers*. O processo é o seguinte:

- Um utilizador faz uma pesquisa de um determinado ficheiro por nome, associado à utilização de uma determinada aplicação P2P;
- A aplicação executa a pesquisa noutros computadores com a mesma;
- Os computadores pesquisados, por sua vez, também efetuam uma pesquisa em computadores terceiros.
- Este ciclo é executado até ser alcançado um ponto especificado pela aplicação.

- Quando o ficheiro é finalmente encontrado, a sua transferência pode então iniciar.

Com vista a utilizar a largura de banda disponível duma forma mais eficiente, a transferência é distribuída pelos múltiplos computadores encontrados com o ficheiro em causa. Este princípio encontra-se demostrado na ilustração 2.

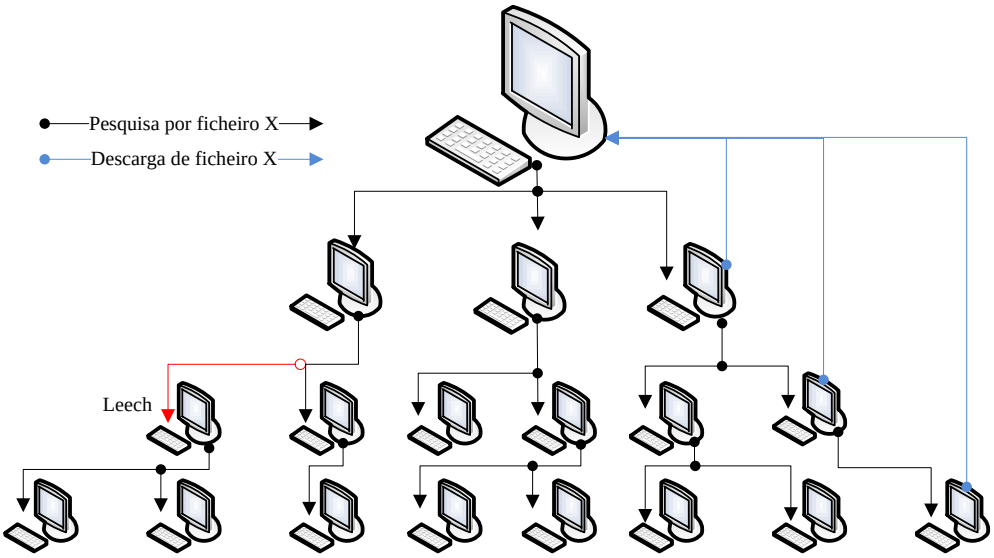

Ilustração 2 – Transferência de um ficheiro numa arquitetura não centralizada de comunicação estruturada indireta

[3] Um sistema P2P semi-centralizado considera a utilização de peers para fazerem as transferências de dados entre si, sendo estas semi-autónomas, tirando também proveito de um ou mais servidores centrais com funcionalidades administrativas com o intuito de produzir um sistema mais eficiente.

Este exemplo demonstra algumas limitações, como no caso de um utilizador servir alguns ficheiros durante apenas a sua descarga, fechando imediatamente aplicação quando termina. Isto pode causar entropia no serviço a outros utilizadores que estavam a usar este computador como semente para as suas transferências. A este tipo de ação dá-se o nome de *leeching* e limita o número de computadores a que a aplicação consegue alcançar nas pesquisas.

1.3.1.3 Partilha de ficheiros P2P com uma arquitetura semi-centralizada

Este caso é também uma transferência P2P, visto que os computadores dos utilizadores são os servidores de arquivo (*peers*), mas com a diferença de usar um servidor central com tarefas específicas para o bom funcionamento da rede. Um utilizador faz um pedido de um determinado ficheiro através do nome, a aplicação executa a pesquisa no servidor central, procurando não só pelo possível ficheiro, como também por quais os nós que o têm e que estão atualmente ativos. Assim que esta informação é obtida, o servidor redireciona para o cliente, para que este possa estabelecer ligações com os nós que têm partes ou a totalidade do ficheiro em causa.
A carga da transferência é distribuída por todos os nós envolvidos, de forma a maximizar o uso da largura de banda, exemplo demostrado na ilustração 3.

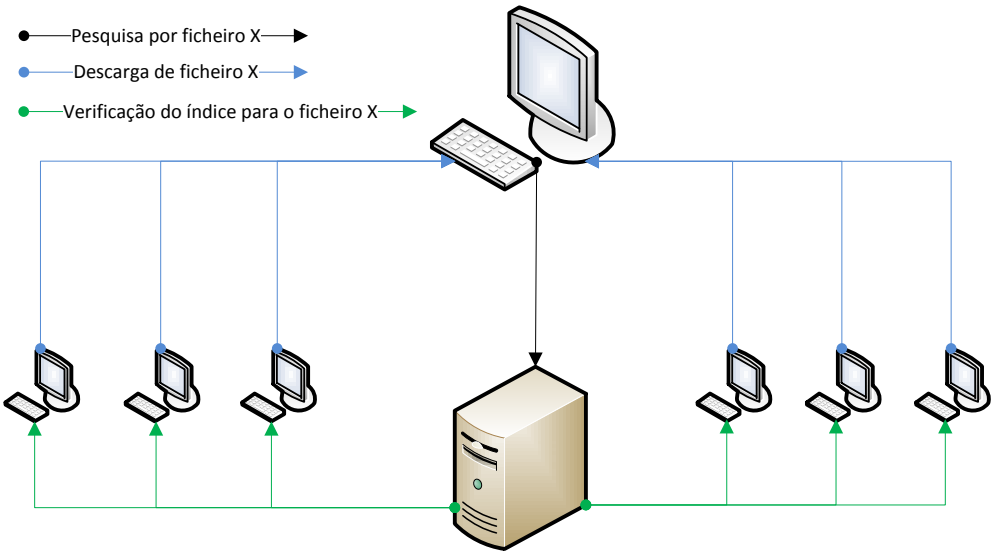

Ilustração 3 - Transferência de um ficheiro numa arquitetura semi-centralizada com um servidor de índice central único

Este exemplo demonstra uma fraqueza. No caso da ligação com o servidor central, ou do próprio, falhar, o funcionamento da arquitetura ficará comprometido. Não será possível executar novas procuras, até o servidor estar novamente disponível. Aqueles utilizadores que já estão a efetuar uma descarga serão menos afetados, porque já estabeleceram ligações com os outros nós necessários, ficando praticamente independentes do servidor.

Uma solução para este problema, é evoluir a arquitetura para o uso de múltiplos servidores centrais, sendo esta a solução adotada por Bram Cohen ao criar a arquitetura BitTorrent.

1.3.2 *Sites* de Redes Sociais

Hoje em dia, os *sites* de redes socias [Ellison 2007] [Ellison et al. 2007] já são uma comodidade quase indispensável para muitos utilizadores. Apesar de terem ganho fama com os utilizadores regulares, correntemente já muitas empresas tiram proveito deste serviço para fornecerem um novo nível de contacto com os seus clientes.

Os *sites* de redes sociais consistem num serviço web de partilha de informação. Este serviço pode ser utilizado pelo facto de, teoricamente, todos os utilizadores poderem interagir com o *site*. Geralmente muitos destes *sites* têm tendência a focalizar um determinado grupo de utilizadores, por exemplo, redes sociais de animais (apesar do meio de interação do animal ser geralmente um humano).

A informação que é partilhada varia bastante de *site* para *site*, mas todos mantêm uma estrutura relativamente consistente, uma página do utilizador no qual partilha a informação que pretende, uma lista de contactos geralmente caracterizados como amigos e uma ou mais formas de comunicação entre os contactos.

Os *sites* de redes sociais são parte importante da arquitetura que é proposta nesta dissertação. A estruturação da plataforma para as relações sociais, é o seu ponto fulcral.

1.4 Inovação Proposta

Face às limitações de transferência de ficheiros das plataformas das redes sociais e à potencialidade de execução desta tarefa do protocolo P2P BitTorrent, esta dissertação apresenta uma arquitetura que consiste na junção destes dois sistemas, o que representa um valor acrescentado para os utilizadores de redes sociais.

Propõe-se então juntar a capacidade de transferência de dados do protocolo BitTorrent, com uma plataforma de comunicação dos *sites* de redes sociais.

A unificação do serviço prestado por estes sistemas potencia funcionalidades inovadoras para os *sites* de redes sociais e um crescimento considerável na utilização do protocolo BitTorrent.

1.5 Estrutura do Documento

O presente documento encontra-se estruturado em 6 capítulos, com os seguintes conteúdos:

- 1º Capítulo: **Introdução** – Apresentação dos objetivos do trabalho, descrição do trabalho anterior, e breve descrição das tecnologias envolvidas;
- 2º Capítulo: **Descrição das Tecnologias Envolvidas** – Este capítulo é dedicado à descrição mais aprofundada das tecnologias envolvidas na arquitetura proposta nesta dissertação;
- 3º Capítulo: **Descrição da Arquitetura Proposta** – Neste capítulo é feita uma descrição aprofundada da arquitetura proposta;
- 4º Capítulo: **Conclusões** – Considerações finais sobre a dissertação;

2 - Descrição das Tecnologias Envolvidas

2.1 Partilha de Ficheiros em P2P

Os protocolos de transferência de ficheiros em P2P são uma parte fundamental da arquitetura proposta. Estes são constituídos por múltiplas arquiteturas que consistem na comunicação e transferência direta ou indireta de dados entre um conjunto de computadores numa rede.

2.1.1 Arquitetura física e lógica

Podemos considerar dois tipos de arquiteturas dentro do tipo P2P, físicas e lógicas. A arquitetura física refere-se às ligações físicas entre os diferentes nós de uma rede (routers e computadores). A arquitetura lógica pode ser considerada uma abstração da primeira em que são apenas contabilizados os nós que contêm a aplicação P2P e as conexões lógicas entre eles. Por exemplo, uma rede pode ser constituída por 10 nós a utilizar o sistema P2P (parte lógica), mas na verdade existem 20 computadores ou outros equipamentos pelo qual os dados estão a passar (parte física). Esta dissertação foca-se nas lógicas, pois é nessa categoria que a proposta se encaixa.

2.1.2 Arquiteturas dinâmicas e estáticas

Vale a pena chamar a atenção a um detalhe sobre as arquiteturas P2P. Estas tanto podem ser estáticas como dinâmicas. Apesar de tipicamente serem dinâmicas, caso em que existem nós a sair e a entrar na rede ou a estabelecerem e quebrarem de forma constante. Não existe razão para um sistema P2P não ser constituído de nós estáticos e estáveis.

2.1.3 Nós ativos e potenciais ativos

Dentro das arquiteturas lógicas, existem também dois tipos de nós. Os ativos, aqueles que estão de facto a participar nas catividades do sistema, no caso de um sistema de transferência de ficheiros, aqueles que estão prontos a descarregar/carregar. Os potenciais ativos, sendo este o número total teórico de possíveis nós ativos em simultâneo no sistema, embora seja muito improvável que aconteça conforme o número de nós na rede aumenta, é importante desenvolver o *software* com este número em mente.

Em uma rede P2P podemos ter por exemplo um número de nós potenciais ativos 100, sendo estes onde a aplicação foi instalada. E uma quantidade de 40 ativos, sendo estes o número de participantes que estão correntemente a tirar proveito das funcionalidades da aplicação em simultâneo.

2.1.4 Utilizadores e nós

É importante salientar que numa rede P2P existe diferença entre utilizadores e nós. Pois é possível que um nó esteja a tomar parte da rede com as suas funcionalidades mas desprovido de utilizadores. Como também é possível que um nó esteja a ser alvo de uso de múltiplos utilizadores.

2.1.5 Noção de igualdade

Numa arquitetura P2P pura existe também a noção de igualdade entre os nós. Deveriam ser considerados todos iguais dentro da rede. No sentido em que nenhum nó tem mais autoridade que um qualquer outro, têm todos o mesmo estatuto e todos participam de igual forma na rede. No entanto, na prática não funciona bem assim. A verdade é que é muito difícil atingir esta teórica igualdade, visto que é muito improvável que todos os computadores na rede tenham a mesma largura de banda, processamento, etc.

O que acontece, por exemplo, é que computadores com maior capacidade de processamento fazem mais tarefas computacionais do que aqueles com menos.

2.1.6 Tipos de arquitetura lógica

Podemos verificar, que existem 2 tipos principais de arquiteturas P2P lógicas, as não centralizadas e as semi-centralizadas. Nas primeiras, também conhecidas como P2P puras, todos os nós dentro da rede são considerados iguais, não existe nó de controlo, sendo estes totalmente autónomos. No segundo tipo, a abordagem é relativamente diferente, são semi-autónomos, na medida em que utilizam um ou mais nós servidores centrais para controlo do serviço com funcionalidades autoritárias administrativas. Este último é também conhecido como arquitetura hibrida.
Seguidamente serão demonstradas 7 arquiteturas de P2P possíveis, dentro das 2 acima referidas, e as suas características predominantes:

2.1.6.1 Arquiteturas não centralizadas

São aquelas que não contêm nó central de controlo. Qualquer nó é considerado igual a todos os outros. Assumindo que têm o mesmo número de funcionalidades dentro da mesma rede. Seguidamente serão descritas 3 destas:

- **Comunicação Direta**

Todos os nós são considerados iguais e autónomos, isto é, independentes, com o mesmo número de funcionalidades, nenhum tem mais autoridade que os outros e quaisquer dados ou computação é distribuída de igual modo entre eles.
Esta arquitetura é referida como comunicação direta, devido ao facto de todos os nós comunicarem diretamente entre eles. Todos têm noção da existência de todos, sendo esta a característica principal. É muito pouco escalável, visto que quanto maior se torna

a rede, mais difícil será para todos os nós estarem cientes uns dos outros. Aconselhável apenas para pequenas estruturas. Exemplificado na ilustração 4.

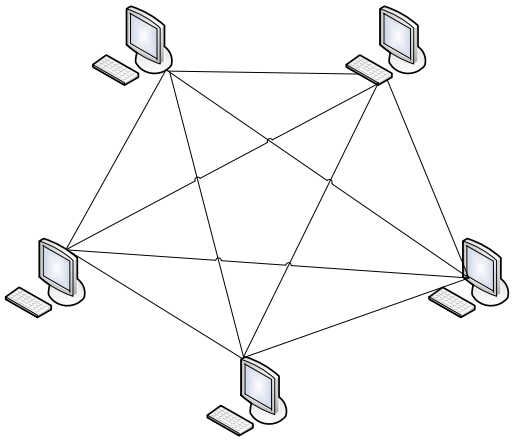

Ilustração 4 - Arquitetura Comunicação Directa

- **Comunicação Estruturada Indireta**

Tal como a descrita acima, todos os nós são considerados autónomos. Nenhum dos nós tem controlo sobre a rede e toda a computação pode ser distribuída por todos.
A diferença deste para o anterior é que não é necessário que todos os nós tenham comunicação direta entre eles. Em vez disso a comunicação pode ser feita através de outro nó intermediário. Ultrapassando assim o problema da escalabilidade.
Esta diferença permite criar vários tipos de topologias, como árvore hierárquica, estrela ou anel. Exemplo de árvore hierárquica na ilustração 5.
Esta arquitetura tem um grande potencial de crescimento, mas sem a gestão adequada pode causar grandes congestionamentos na rede.

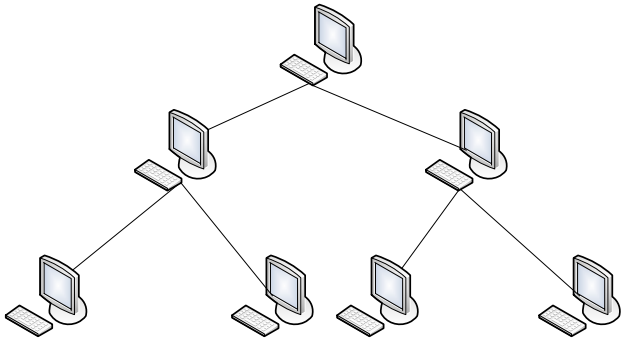

Ilustração 5 - Arquitetura Comunicação Estruturada Indireta

- **Comunicação não Estruturada Indireta**

Tal como ambas as arquiteturas mencionadas anteriormente, todos os nós são considerados iguais e autónomos. Nenhum nó tem controlo sobre a rede e toda a computação necessária pode ser distribuída por qualquer um.

Assim como na Comunicação Estruturada Indireta, não é necessário que todos os nós comuniquem diretamente entre eles. A diferença desta para a anterior é que não existe qualquer tipo de estruturação, resultando numa grande variedade na quantidade de nós agrupados em determinadas partes da rede. Exemplificado na ilustração 6.

Este tipo de arquitetura retira a preocupação de qualquer tipo de gestão na configuração da rede, no entanto cria uma maior preocupação no sistema de procura. Pois é possível que a qualquer altura haja mudanças na rede, sem que os outros nós se apercebam disso.

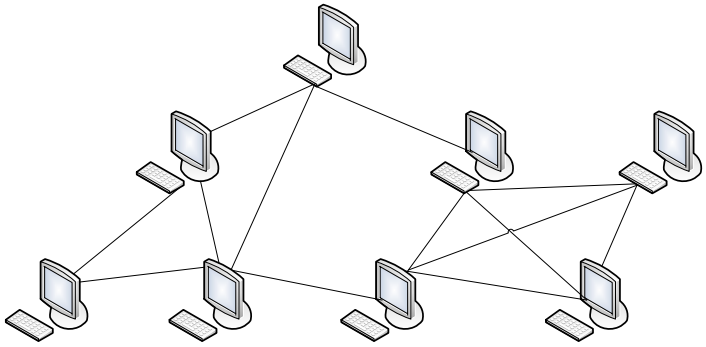

Ilustração 6 - Arquitetura Comunicação não Estruturada Indireta

2.1.6.2 Arquiteturas Semi-Centralizadas

São consideradas Semi-Centralizadas porque contêm pelo menos um nó central de controlo. O propósito deste nó pode variar, desde manter um controlo restrito sobre toda uma rede, a ser simplesmente um ponto de referência para os outros nós. Seguidamente serão descritas 4 destas arquiteturas:

- **Servidor de Índices Centralizado Singular**

Neste tipo de arquitetura existe um único nó que tipicamente serve como ponto de referência para todos os outros nós dentro da rede. Esses são vistos como autónomos e é através deles que todo o tipo de dados ou computação é espalhada.
Todos os nós conseguem comunicar diretamente entre eles, no entanto, este fim é tipicamente alcançado com o uso do servidor central para adquirir a localização do nó com o qual pretendem comunicar.
A vantagem de usar um nó central como servidor de índice é que a necessidade de usar algoritmos de descoberta é reduzida ou mesma removida. Quando um nó se liga à rede, informa o nó central da sua localização. Isto significa que estabelecer ligações entre nós pode ser um processo simples e direto.
Um problema desta arquitetura é que no caso do nó central falhar, os outros nós ficam sem meios de se encontrarem. É possível no entanto, adicionar funcionalidades de

redundância, como usar algoritmos de descoberta de nós, no caso do nó central falhar. Exemplificado na ilustração 7.

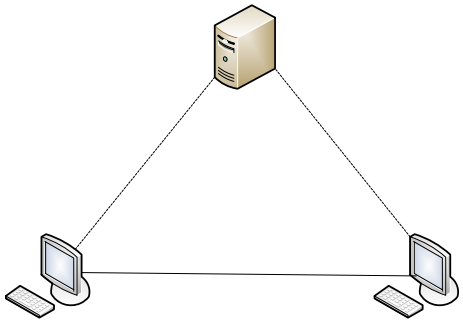

Ilustração 7 - Arquitetura Servidor de Índices Centralizado Singular

- **Semi-Centralizada Computacional Sem Autonomia**

Existem duas arquiteturas Computacionais, sem e com autonomia. Ambas são semelhantes à arquitetura discutida acima. Ambas usam um nó central como ponto focal para os outros nós.

A principal diferença, é que nesta arquitetura os nós restantes nós da rede, não possuem autonomia sendo controlados pelo nó central. Adicionalmente a comunicação entre eles é também feita através deste servidor.

É discutível se esta arquitetura deve ou não ser considerada P2P, visto que os nós não têm qualquer autonomia e usam um servidor central para executar todas as suas funcionalidades de comunicação. Exemplificado na ilustração 8.

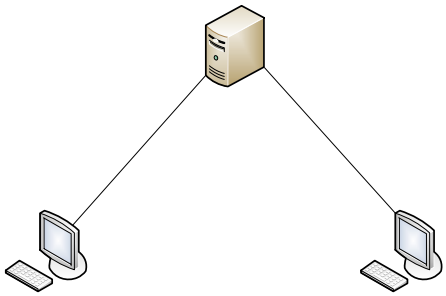

Ilustração 8 - Arquitetura Semi-Centralizada Computacional Sem Autonomia

- **Semi-Centralizada Computacional Com Autonomia**

Esta arquitetura é idêntica à anterior, com a diferença que os nós mantêm algum nível de autonomia entre eles. Como estabelecer comunicação com outros nós sem o auxílio do nó central de controlo, tal como descrito na arquitetura servidor de índices centralizado singular. Exemplificado na ilustração 9.

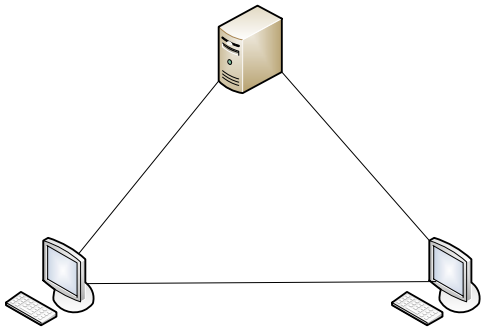

Ilustração 9 - Arquitetura Semi-Centralizada Computacional Com Autonomia

- **Múltiplos Nós Servidores**

Todas as arquiteturas Semi-Centralizadas descritas até agora têm sido baseadas na ideia de usar apenas um único servidor ou nó central de controlo. No entanto não existem restrições para não serem usados múltiplos nós centrais. Uma das grandes vantagens desta arquitetura é que ao desaparecer o ponto focal, ou este ser dividido em múltiplos nós, aumentamos a confiabilidade do sistema. Ou mesmo no caso de os outros nós estarem fisicamente a grandes distancias uns dos outros, usar múltiplos servidores aumenta a qualidade do serviço.

A comunicação neste tipo de arquitetura tem parecenças com a do servidor de índices centralizado único. A grande diferença esta na capacidade de os servidores também poderem comunicar entre eles. Isto significa que estes têm a possibilidade de associar as suas bases de dados para resolver problemas como determinar a localização de nós. Exemplificado na ilustração 10.

Um dos problemas desta arquitetura é a forma como os nós comunicam com os servidores. Se um determinado número de nós se conectar a apenas um servidor, esta ligação torna-se um potencial ponto de falha. No entanto se permitirmos que todos os nós se liguem a todos os servidores, poderemos estar a limitar as vantagens desta arquitetura. Uma forma de resolver este problema é implementar algoritmos apropriados à decisão de comunicação com os servidores.

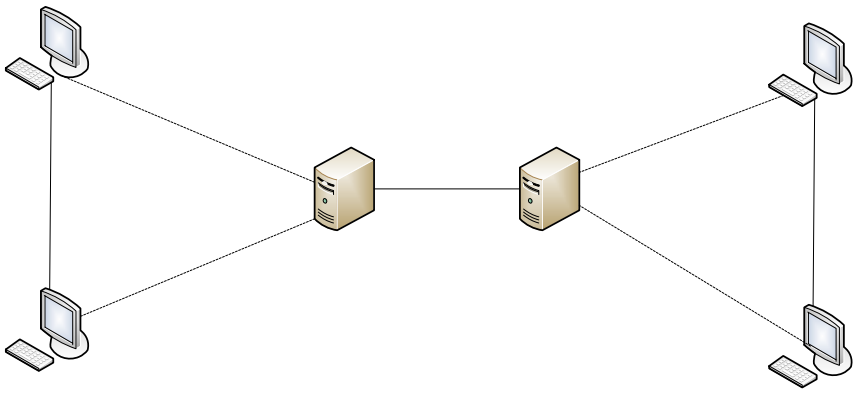

Ilustração 10 - Arquitetura Múltiplos Nós Servidores

2.2 Partilha de Ficheiros em BitTorrent

BitTorrent é um protocolo P2P, criado por Bram Cohen, que tira proveito de uma arquitetura semi-centralizada com múltiplos nós servidores para fazer partilha de ficheiros. Este é o tipo de arquitetura P2P focada nesta dissertação.

2.2.1 Como Funciona

Sendo um protocolo P2P, e seguindo o exemplo acima mencionado, a carga de um determinado ficheiro é dividida por múltiplos nós que contêm partes ou a totalidade do mesmo. Desta forma é tirado um melhor proveito da largura de banda.

O facto de a carga ser feita pelos nós utilizadores, permite que o número dos mesmos envolvidos na descarga de um determinado ficheiro em simultâneo se torne potencialmente ilimitado.

Um utilizador faz a procura de um determinado ficheiro num servidor regular que contem uma lista de pequenos ficheiros com a extensão ".torrent" (existem múltiplos servidores deste género, mas é importante chamar a atenção que estes não fazem parte da arquitetura integral BitTorrent). Estes ficheiros contêm a informação necessária para que os utilizadores possam proceder à descarga do ficheiro principal. Esta informação é o seu tamanho, o nome, a informação de *hashing* e o URL (Uniform Resource Locator) de um servidor de rastreamento ou *tracker*. Este último é um servidor responsável por ajudar a efetuar o encontro entre os nós ativos, para que a descarga/carga de ficheiros possa começar.

Os servidores de rastreamento comunicam a partir de um protocolo muito simples numa camada em cima de HTTP (Hypertext Transfer Protocol), no qual um nó que pretende efetuar uma descarga, envia informação sobre o ficheiro que pretende descarregar e a porta em que está à escuta. O servidor de rastreamento responde com uma lista de informação sobre outros nós que também estão ativos e contêm o ficheiro em causa. Os interessados usam então esta informação para criar uma ligação entre

eles, e o processo de carga/descarga pode começar. Os requerimentos de largura de banda dos servidores regulares e dos servidores de rastreamento são muito baixos.

2.2.2 Interface

Correntemente a interface da maioria das aplicações que tiram proveito do sistema BitTorrent é relativamente simples. Geralmente um utilizador acede a um *site* com uma lista de ficheiros ".torrent", procura e clica no link do ficheiro que pretende descarregar. O utilizador guarda e abre este ficheiro ".torrent" na aplicação BitTorrent que por sua vez faz a gestão necessária para que a descarga do ficheiro principal possa acontecer. Esta facilidade no uso foi uma das principais razões para a adoção do sistema BitTorrent pela maioria.

2.2.3 Situação Típica de Descarga/Carga de um Ficheiro

Numa situação típica, o número de descarregadores incompletos, (sendo estes os que não têm um determinado ficheiro na totalidade), aumenta rapidamente após o ficheiro ter sido publicado. Eventualmente atinge um máximo e começa a descer abruptamente. O número de descarregadores completos aumenta lentamente, atinge um máximo após o anterior, mas depois também começa a descer rapidamente.
O máximo de descarregadores incompletos baixa conforme o número de completos aumenta. O máximo de descarregadores completos baixa conforme estes param de carregar o ficheiro.
Este padrão também é refletido em novos utilizadores que se juntam à partilha do ficheiro após a corrida inicial.

2.2.4 Problemas

Existem alguns problemas de desenvolvimento associados a este tipo de arquitetura. Logística e robustez são duas áreas bastante complicadas. O simples descobrir de qual dos nós tem o quê e para onde devem ser enviados é bastante difícil de atingir sem incorrer a grandes despesas computacionais.

Existem também grandes variações no número de utilizadores a tirar proveito da ferramenta em simultâneo. Estes raramente ficam ligados mais do que uma ou duas horas e frequentemente apenas alguns minutos. O que gera um aumento significativo na dificuldade da gestão do sistema.

Em causa está também o problema da igualdade. Teoricamente numa situação perfeita, a quantidade total de descarga de um determinado nó deveria ser igual à sua quantidade de carga. Esta aparenta ser a melhor estratégia para a satisfação dos utilizadores visto a capacidade de descarga se tornar proporcional à carga feita. No entanto na prática esta igualdade é mais complicada de atingir do que aparenta. É difícil por exemplo fazer com que a capacidade de descarga de alguns utilizadores a determinada altura não desça para 0, resultante dos baixos níveis de carga. Normalmente os utilizadores fecham de imediato a aplicação assim que terminam a descarga do ficheiro que pretendem adquirir. Apesar destes casos, é considerada uma questão de educação deixar a aplicação a carregar para outros utilizadores durante algum tempo mesmo após satisfeitas as suas necessidades de descarga.

2.2.5 Distribuição de Peers

Todos os problemas logísticos das descargas de ficheiros são tratados através de interações entre *peers* (ou nós da rede). Alguma informação sobre as cargas ou descargas da informação é enviada para os servidores de rastreamento apenas para propósitos estatísticos. As responsabilidades destes servidores estão estritamente limitadas a ajudar no encontro entre *peers*.

Apesar dos servidores de rastreamento serem a única forma que os *peers* têm de se encontrar e serem o único ponto de coordenação, o algoritmo padrão usado retorna uma lista de nós aleatória. A escolha deste tipo de algoritmo está relacionada com questões de robustez.

Para que o BitTorrent consiga manter um controlo sobre quem tem o quê, os ficheiros são cortados em pequenos pedaços de um tamanho fixo, tipicamente um quarto de um *megabyte*. Os nós, após comunicarem com o servidor de rastreamento, informam-se mutuamente sobre quais as partes do ficheiro que possuem. Para verificar a integridade dos dados, é usado o algoritmo SHA1 em todos os pedaços e seu *hash* incluído no ficheiro ".torrent". Assim sendo, um peer só dá a indicação que tem um determinado pedaço depois de verificar o seu *hash*.

Os *peers* fazem continuamente a descarga dos pedaços que ainda lhes faltam de todos os outros a que estão conectados. Logicamente não podem fazer descarga de nós ao qual não estão ligados, visto não terem sido disponibilizados pelo servidor de rastreamento. Também acontecem situações em que um determinado *peer* não tem partes em que o outro esteja interessado, ou apesar de ter essas partes, não está correntemente a permitir a sua descarga (a este tipo de Acão dá-se o nome de engasgo ou engasgar outros *peers*).

2.2.6 Lista de Pedidos em Espera

Quando é feita uma transferência de dados em TCP, como o BitTorrent faz, é muito importante ter sempre vários pedidos em espera. Isto para evitar atrasados entre o envio de pedaços do ficheiro. Se assim não fosse a velocidade de transferência seria extremamente afetada. Este protocolo lida com este problema a partir os pedaços de ficheiro em sub-pedaços na lista de espera, tipicamente com 16 *kilobytes* de tamanho e mantendo simultaneamente cerca de 5 pedaços em espera. Cada vez um sub-pedaço chega ao destino, um novo pedido é feito.

2.2.7 Seleção de pedaços

A seleção de pedaços para descarregar em ordem correta é também importante para boa performance. Um algoritmo descuidado pode resultar em *peers* ligados que não têm os pedaços necessários da lista de espera, comprometendo a performance da descarga do ficheiro. Seguidamente serão descritas várias técnicas usadas na seleção de pedaços.

2.2.7.1 Prioridade Estrita

É a primeira regra de seleção de pedaços do BitTorrent, assim que um sub-pedaço é pedido, os restantes sub-pedaços daquele pedaço em particular serão os próximos a entrar em fila de espera, antes de passar para sub-pedaços de outro pedaço. Esta regra faz um bom trabalho a completar pedaços o mais rápido possível.

2.2.7.2 Raridade Prioritária

Quando é feita a seleção da ordem dos pedaços a descarregar, a escolha é feita através de um algoritmo que dá prioridade aos pedaços mais raros entre a lista de *peers* a que o nó em causa está conectado. Esta técnica faz um bom trabalho ao tentar garantir que os *peers* têm pedaços em que todos aqueles a que estão ligados têm interesse. Consequentemente todos os pedaços que são considerados mais comuns são deixados para o fim, portanto a probabilidade de um *peer* que está correntemente a carregar pedaços não ter nada de interesse mais tarde, é reduzida. É importante salientar que uma descarga só pode ser marcada como completa após a obtenção de todos os pedaços do ficheiro em causa.

Esta regra tem também a vantagem de no caso de um ficheiro ter originalmente apenas uma semente e esta ter baixa capacidade de carga, a performance vai ser bastante melhor se diferentes *peers* obterem pedaços diferentes deste ficheiro, visto que

descargas redundantes desperdiçam a oportunidade de reduzir a raridade de mais pedaços do ficheiro.

Em alguns casos acontece também a situação em que a semente original de um determinado ficheiro para de o servir, deixando apenas os *peers* a fazer o seu carregamento. Isto pode resultar no desaparecimento por completo de alguns pedaços da rede. Uma vez mais regra de raridade prioritária lida bem com estes casos, replicando rapidamente os pedaços mais raros primeiro.

2.2.7.3 Primeiro Pedaço Aleatório

Uma exceção à regra da raridade prioritária é quando a descarga de um ficheiro é iniciada. Naquele momento, o *peer* não tem nada para carregar, portanto é importante conseguir um pedaço completo o mais rápido possível. Os pedaços mais raros estão em menos *peers*, logo a descarga do primeiro pedaço seria mais lenta. Por esta razão, os primeiros pedaços escolhidos são aleatórios até um pedaço estar completo. Assim que isto acontece a estratégia volta a ser a raridade prioritária.

2.2.7.4 Finalização da Descarga

Em certos casos, um pedaço pode ser pedido de um *peer* com velocidade de carga muito baixa. Isto não é um problema no meio da descarga de um ficheiro, mas pode causar um atraso significativo na altura final do processo. Para impedir que isto aconteça, assim que um *peer* chega à fase em que já não tem falta de mais pedaços, apenas sub-pedaços, este envia pedidos destes sub-pedaços também aos outros *peers*. Assim que estes vão chegando, são enviados também cancelamentos dos pedidos já obtidos, para evitar desperdícios de largura de banda em transferências redundantes. Na prática não é desperdiçada muita largura de banda visto que este período final é muito curto devido ao fim do ficheiro ser descarregado com alta velocidade.

2.2.8 Algoritmos de Engasgo

Como já foi anteriormente referido, BitTorrent não tem nenhuma central para alocação de recursos. Cada *peer* é responsável por tentar maximizar as suas velocidades de transferência. Para alcançar este objetivo, os *peers* descarregam de quem conseguem, mas apenas carregam consoante um algoritmo de *tit-for-tat*. Para cooperar os *peers* carregam dados, para não cooperar, estes usam técnicas de engasgo nos outros *peers*. Engasgo, tal como já foi anteriormente referido, é o nome que se dá à ação de temporariamente recusar a descarga de dados. A ligação não é perdida entre os dois pontos, o peer que está a carregar, simplesmente desabilita esta possibilidade temporariamente, podendo ser retomada posteriormente sem causar distúrbio na ligação.

Os algoritmos de engasgo não fazem parte integral da arquitetura BitTorrent, mas são uma parte essencial para a boa performance do sistema. Um bom algoritmo de engasgo deveria ser capaz de utilizar todos os recursos disponíveis, fornecer velocidades de descarga consistentes para todos e ser razoavelmente resistente a *leeches*.

2.2.8.1 Eficiência de Pareto

Um sistema é considerado pareto eficiente quando duas contrapartes alcançaram o máximo de benefício em trocas mútuas. Trocas extras entre as duas contrapartes seriam inúteis na medida em que nenhuma teria benefícios adicionais.

Em termos de ciência computacional, alcançar uma eficiência pareto consiste em algoritmos que envolvem pares de contrapartes que tentam melhorar o seu lote juntas.

Especificamente, no caso de dois *peers* independentes, estarem a receber pouca retribuição para a quantidade de carga que estão a fornecer, estes podem passar a fazer trocas entre si para melhorar a sua velocidade de descarga.
Os algoritmos de engasgo BitTorrent tentam alcançar esta eficiência usando uma versão aprofundada do algoritmo *tit-for-tat* já acima referido. Este consiste em *peers* retribuírem favores, ou neste caso, carregar dados. Estes retribuem com mais carga a aqueles que também estão disponíveis para o fazer, com o objetivo de a qualquer momento haver várias ligações cativas a transferir nas duas direções. Ocasionalmente também são usadas ligações alternativas para testar se melhores velocidades de transferência são possíveis ao usá-las.

2.2.8.2 O algoritmo de engasgo BitTorrent

A um nível mais técnico, por omissão, o número de *peers* que cada utilizador de BitTorrent desengasga é quatro, portanto o desafio torna-se escolher quais os *peers* a desengasgar. Esta abordagem permite à funcionalidade de controlo de congestão do modelo TCP conseguir eficazmente saturar a capacidade de carga da ligação.
As decisões de escolha sobre quais os *peers* a desengasgar ou não, são baseadas na velocidade de descarga corrente. Surpreendentemente verificou-se que fazer o cálculo que permite obter esta informação de forma significativa é um desafio inesperadamente difícil. A implementação corrente usa um ciclo de cálculo da média da velocidade de transferência durante janelas de 20 segundos.
Para evitar situações em que recursos são desperdiçados ao rapidamente engasgar e desengasgar *peers*, estes recalculam quem querem engasgar ou desengasgar a cada 10 segundos.

2.2.8.3 Desengasgo otimístico

Esta é uma técnica usada para resolver um problema criado pelo algoritmo de escolha de *peers*. Pois se forem feitos carregamentos apenas para os *peers* que fornecem maior capacidade de descarga, não existe método de descobrir se ligações que não estão a ser usadas fornecem melhores velocidades de transferência do que as que estão a ser usadas. Para resolver isto, um utilizador BitTorrent tem continuamente um único desengasgo otimístico, este é escolhido independentemente da velocidade de descarga fornecida. A escolha é feita a cada terceiro ciclo de decisão de *peers* a engasgar ou desengasgar, ou seja, a cada 30 segundos. Segundo Bram, 30 segundos é tempo suficiente para que a velocidade de carga e descarga atinjam a sua capacidade máxima.

2.2.8.4 Anti Desprezo

Ocasionalmente acontece o caso em que um *peer* fica engasgado por todos os *peers* do qual esteve anteriormente a descarregar. Nestes casos este *peer* vai continuar a receber velocidades baixas de descarga até o desengasgo otimístico encontrar outros *peers* melhores. Para tentar mitigar este problema, quando acontece o caso em que durante 1 minuto não foi possível descarregar um único pedaço de um determinado *peer*, é assumido que este está a ser desprezado por esse *peer* e portanto para de carregar para o mesmo, com a exceção do desengasgo otimístico. Esta técnica frequentemente resulta em mais do que um desengasgo otimístico em simultâneo (em exceção à regra acima referida), o que faz com que a velocidade de descarga recupere bastante mais rapidamente quando estas situações ocorrem.

2.2.9 Fase de Carga

Assim que um *peer* acaba a sua descarga, deixa de ter velocidades de descarga para decidir quais os *peers* para o qual pretende carregar. Assim, nesta fase a implementação corrente muda o algoritmo de escolha de peers para aqueles que têm

maior velocidade de carga. Esta técnica faz um trabalho decente ao tirar proveito da largura de banda de carga e ao preferir peers para o qual ninguém está a carregar.

2.3 *Sites* de Redes Sociais

As redes sociais na web são a outra face da arquitetura proposta nesta dissertação, sendo esta igualmente importante para a inovação proposta, vamos agora apresentar uma possível definição e algumas das características mais comuns.

Os *sites* de redes sociais podem ser considerados serviços web que disponibilizam as seguintes funcionalidades base, aos seus utilizadores:

- Construir um perfil público ou semipúblico com informações pessoais;
- Gerir uma lista de outros utilizadores com o qual pretendem manter contacto;
- Visitar e utilizar funcionalidades de comunicação com a sua lista de contactos e outros.

Todos os *sites* de redes sociais têm um propósito em comum, partilhar informação. A natureza dessa informação pode variar de *site* para *site*. Alguns têm um foco maior na criação de relações entre os utilizadores, outros têm objetivos mais específicos de resolução de problemas.

No entanto aquilo que torna os *sites* de redes sociais únicos, não é a capacidade de interagir com utilizadores desconhecidos, mas sim fornecer a capacidade de publicar e manter visível a informação que desejarem com a sua rede social. Apesar de isto poder resultar em relações entre utilizadores que não aconteceriam de outra forma, este geralmente não é o objetivo, e mesmo estes contactos partilham normalmente alguma ligação passada fora da *web*. Em grande parte dos *sites* de redes sociais, aquilo que os utilizadores procuram é a possibilidade de comunicar de uma forma mais conveniente com pessoas que já faziam parte da sua vida social fora da *web*.

Ao longo dos tempos, este tipo de *sites* foi evoluindo bastante, apresentando novas funcionalidades técnicas para proporcionar uma melhor experiencia ao utilizador. No

entanto a sua característica principal, e aquela que proporciona o desenvolvimento de todas as outras, é a capacidade de manter um perfil pessoal e uma lista de contactos que também estes são utilizadores do sistema.

Os perfis são páginas onde o utilizador publica informação pessoal que pretende partilhar com todos ou um grupo de utilizadores do sistema. Geralmente este perfil é criado na altura em que o utilizador se regista no sistema. São apresentadas algumas questões específicas das quais a maioria é opcional, como a idade, interesses, localização, etc. No entanto recusar o fornecimento destas informações pode colocar em causa o propósito do perfil. A maioria dos *sites* também encoraja a publicação de uma fotografia de apresentação. Alguns *sites* permitem também a alterações visuais da interface de utilizador. Ou mesmo a adição de módulos com aplicações multimédia.

A visibilidade destes perfis e da informação evidenciada depende do *site* e das permissões fornecidas pelos seus utilizadores. É necessário ter atenção que ao colocar certos tipos de informação num perfil público (perfil a que todos os utilizadores do sistema têm acesso) pode proporcionar um risco real para a sua vida pessoal. Visto que a gestão da visibilidade varia bastante de *site* para *site*, este detalhe pode fazer uma grande diferença nos benefícios de uso a longo termo.

Apos o registo, em alguns *sites*, os utilizadores são solicitados a identificar outros utilizadores do sistema com quem partilham algum tipo de relação. O nome dado a estas relações pode mudar dependendo do *site*, mesmo contendo o mesmo propósito. Geralmente existem dois tipos base de proposta de relação entre utilizadores no sistema, bidirecional e unidirecional. A primeira consiste no envio de uma proposta de relação, normalmente atribuída a amizade, que necessita ser aceite pela outra parte para ser estabelecida. A segunda, tal como o nome indica, não é necessária a aceitação da outra parte, este tipo de proposta de relação está normalmente associada a seguidores ou fans de um determinado utilizador do sistema.

Um dos componentes mais importantes dos *sites* de redes sociais é a demonstração pública das listas de conexões entre os utilizadores. Estas normalmente contêm ligações de acesso

ao perfil de cada utilizador, o que permite um acesso simples e rápido de navegação pela rede. É comum a lista estar visível a todos os utilizadores que têm acesso a um determinado perfil, mas nem todos os *sites* seguem esta regra, em certos casos é possível esconde-la de outros utilizadores.

Outra das funcionalidades comuns destes *sites* é a capacidade de publicar mensagens no perfil dos utilizadores que mantêm algum tipo de relação no sistema. Adicionalmente também existem *sites* que permitem executar a mesma ação mas em privado, o que significa que apenas o utilizador associado ao perfil consegue visualizar a informação transmitida.

Alem das funcionalidades base de praticamente todos os *sites* de redes sociais, existem também algumas que variam bastante, específicas de cada site. Como exemplos temos a partilha de músicas ou vídeo, sistema de troca de mensagens instantâneas, uso de telemóveis, entre outras. Estes *sites* tendem também a ter públicos-alvo específicos, separados por áreas geográficas, linguísticas, religiosas, aplicacionais, etc.

2.4 Modelo TCP/IP

Neste capítulo é feita uma descrição do modelo TCP/IP, sendo este uma parte importante da arquitetura proposta nesta dissertação. Primeiro é feita uma breve introdução à história do modelo. Seguidamente, são descritas as suas camadas, usando como referencia o modelo OSI (*Open Systems Interconnection*), na qual a arquitetura TCP/IP pode ser mapeada.

2.4.1 História resumida do TCP/IP

O TCP/IP é uma arquitetura de redes que divide as funcionalidades em diversas camadas. Esta arquitetura define o que cada camada deverá fazer, não definindo como tais funções são desempenhadas. Cabe aos protocolos de cada camada definir a forma como essas funções são desenvolvidas.

O *Internet Protocol* (IP) é o protocolo primário da camada de rede (terceira camada) do modelo OSI, o qual tem como função principal efetuar o encaminhamento de datagramas, e o necessário endereçamento. O *Transmission Control Protocol* (TCP) é o protocolo primário da camada de transporte (quarta camada) do modelo OSI, sendo responsável pelo estabelecimento e gestão de conexões e pelo transporte de dados confiáveis entre processos de software em diferentes máquinas.

Devido à importância destes dois protocolos, as suas abreviaturas foram escolhidas para representar o conjunto TCP/IP. Estes dois são os mais importantes porque a maioria das suas funcionalidades mais críticas estão implementadas na terceira e quarta camada do modelo OSI. No entanto, o TCP/IP necessita do trabalho conjunto de bastantes protocolos e tecnologias diferentes para tornar uma rede funcional fornecendo devidamente as aplicações necessárias.

O TCP/IP foi inicialmente criado como parte de uma rede de investigação nos Estados Unidos da América pela ARPA (*Advanced Research Projects Agency*). Inicialmente esta rede, chamada ARPAnet, foi desenhada para usar um número de protocolos que tinham sido adaptados de tecnologias já existentes. No entanto, todos eles tinham as suas falhas ou limitações quando usados na ARPAnet. Aqueles que estavam a desenvolver esta nova rede aperceberam-se que tentar usar estes protocolos já existentes, podia eventualmente levar a problemas conforme a ARPAnet ganhava uma maior de utilização.

Em 1973, foi iniciado o desenvolvimento de um conjunto de protocolos desenhado para a ARPAnet. Inicialmente, o nome atribuído a este conjunto foi TCP, que nesta altura significava (*Transmission Control Program*). A primeira versão do antecessor do TCP moderno foi escrita em 1973, depois foi revista e formalmente documentada em Dezembro de 1974.

Os testes e o desenvolvimento do TCP continuaram pelos anos que se seguiram. Em Março de 1977, foi documentada a versão 2 do TCP. Agosto do mesmo ano foi uma altura importante de mudança no caminho evolutivo do TCP/IP. Foi nesta altura que

Jon Postel publicou um artigo (*Internet Engineering Note Number 2*) [Postel 1977] contendo a seguinte citação:

> *We are screwing up in our design of internet protocols by violating the principle of layering. Specifically we are trying to use TCP to do two things: serve as a host level end to end protocol, and to serve as an internet packaging and routing protocol. These two things should be provided in a layered and modular way. I suggest that a new distinct internetwork protocol is needed, and that TCP be used strictly as a host level end to end protocol.*

O que Postel quis essencialmente demonstrar, é que o TCP da altura tentava fazer demasiado, abrangendo tanto a terceira como a quarta camada do modelo OSI (como referência, pois este ainda não ter sido criado na altura).

Foram as observações de Postel que levaram à criação da arquitetura TCP/IP, dividindo o antigo TCP em dois novos protocolos. O TCP da camada de transporte e o IP da camada de rede, originando assim o nome TCP/IP.

O processo de divisão começou em 1978 com a escrita da versão 3 do IP. Mas só em 1980 é que o primeiro padrão formal para as versões do IP moderno foi criado (versão 4). Esta é a razão pelo qual o IP moderno começar apenas na versão 4 e não na 1. A partir desta altura o TCP/IP tornou-se rapidamente a arquitetura protocolar de redes padrão usada para correr a ARPAnet. Mais e mais, computadores e redes foram-se juntando à ARPAnet usando o TCP/IP, e foi assim que a Internet surgiu.

2.4.2 Fatores de sucesso do TCP/IP

Seguidamente são descritas as principais razões para o sucesso do TCP/IP.

Além do facto de ter sido desenvolvido com a Internet, o que lhe garantiu uma grande parte da sua fama e utilização, o TCP/IP tem algumas características que foram

essenciais para o seu crescimento, entre elas, seguidamente são descritas as mais importantes:

- **Sistema de endereçamento integrado**: O TCP/IP incluí um sistema de identificação e endereçamento de aparelhos numa rede. O sistema foi desenhado para permitir que os dispositivos tenham um endereço independentemente dos detalhes de como cada rede é construída. Esta funcionalidade foi melhorando ao longo dos tempos para atender às necessidades crescentes da Internet.
- **Desenhado para encaminhamento**: O TCP/IP foi especificamente desenhado para facilitar o encaminhamento de informação numa rede de complexidade arbitrária. Na verdade, este protocolo está conceptualmente mais preocupado com as ligações entre as redes do que com as ligações com os dispositivos.
- **Independência da rede subjacente:** O TCP/IP opera principalmente na terceira camada e superiores, e inclui métodos que lhe permitem funcionar em quase qualquer tecnologia de camadas inferiores. Esta flexibilidade significa que é possível misturar uma variedade de diferentes redes subjacentes e liga-las todas com TCP/IP.
- **Escalabilidade:** Uma das características mais importantes do TCP/IP é a escalabilidade que os seus protocolos provaram ter. Ao longo das décadas foi sendo capaz de provar o seu valor conforme a Internet foi crescendo, de uma pequena rede, para uma enorme rede com milhões de utilizadores.
- **Padrões e processo de desenvolvimento abertos:** Os padrões do TCP/IP não são proprietários, estão totalmente disponíveis ao público. Além disso, os processos usados para desenvolver estes padrões estão também disponíveis. Esta particularidade assegura que qualquer pessoa com interesse nos protocolos do TCP/IP tem oportunidade de contribuir para o seu desenvolvimento, consequentemente aumentando a sua aceitabilidade geral.

É também importante salientar que além do TCP/IP ter sido o principal protocolo de suporte ao crescimento da Internet, é também o mais usado em redes privadas. Conforme a Internet continua a crescer, também as capacidades e funcionalidades do TCP/IP se expandem. Neste contexto, é bastante provável que este continuará a ser uma grande parte da enorme rede mundial no futuro próximo.

2.4.3 Serviços TCP/IP e Operações Cliente/Servidor

O modelo TCP/IP é baseado no conceito de uma camada fornecer serviços às camadas acima. Este abrange várias camadas do modelo OSI, e portanto, coletivamente proporciona serviços deste tipo de várias formas. Conceptualmente, é possível dividir os serviços do TCP/IP em dois grupos: serviços fornecidos a outros protocolos e serviços fornecidos diretamente aos utilizadores finais.

O primeiro grupo consiste em funcionalidades essências implementadas pelos protocolos principais do TCP/IP, como o IP, TCP e UDP (*User Datagram Protocol*). Estes serviços foram desenhados para realizar as funcionalidades de rede do modelo. O IP por exemplo, da camada de rede, fornece funcionalidades, como endereçamento ou encapsulamento de datagramas, que são usadas por protocolos de camadas superiores, como o TCP e UDP da camada de transporte.

O segundo grupo contém os serviços fornecidos aos utilizadores finais. Estes têm o propósito de facilitar as aplicações usadas pelos utilizadores a aceder à Internet e a outras redes TCP/IP. Por exemplo a WWW (*World Wide Web*), é uma das mais importantes aplicações de acesso à Internet. A WWW funciona através do HTTP (*Hypertext Transfer Protocol*), um protocolo da camada de aplicação do modelo TCP/IP. O HTTP por sua vez usa serviços fornecidos por protocolos de camadas inferiores.

Uma das propriedades mais caracterizantes dos serviços TCP/IP é estes operarem primariamente numa estrutura cliente/servidor. Esta estrutura pode ser usada tanto para

hardware como *software*, e foi nela que o desenho dos protocolos e aplicações do TCP/IP foram baseados. O cliente inicia a comunicação ao enviar um pedido de dados ou outra informação a um servidor (geralmente um computador preparado para receber pedidos de múltiplos clientes), este responde passando, a informação pedida ao cliente, enviando uma mensagem de erro, ou informação relativa à possível localização da informação requerida.

Os termos "cliente" e "servidor" podem ser confusos no TCP/IP na medida em que estes podem ser usados em diferentes situações e ocasionalmente, simultaneamente. Estes termos diferem em uso consoante a situação, nomeadamente:

- ***Hardware***: O "cliente" é normalmente um computador de capacidades medianas. É aquele que geralmente inicia as conversações com o servidor ao fazer pedidos. O "servidor" tende a ser uma máquina com mais capacidade computacional, habitualmente capaz de processar vários pedidos simultaneamente.
- ***Software***: Geralmente é orientado conforme o *hardware*, ou seja, é espectável que um computador cliente tenha *software* cliente, *software* este que executa os pedidos, recebe as respostas e faz a interação com o utilizador final. O *browser* usado para aceder à Internet é um bom exemplo. O *software* servidor é aquele que processa os pedidos recebidos do cliente e envia as respostas. Não tem necessariamente uma interface de utilizador e pode ser especificamente desenvolvido em consonância com um *hardware* mais computacionalmente capaz. É necessário salientar que nem sempre esta visão se verifica. Existem casos em que um dispositivo pode estar a correr *software* de cliente e de servidor simultaneamente.
- **Situações transacionais:** Em qualquer situação de transação de informação, o cliente é aquele que faz o pedido, o servidor é aquele que lhe fornece a resposta. Esta visão é independente do *software* ou *hardware* usado, portanto é espectável que se verifiquem variados casos em que um computador com

software de cliente, responda a um pedido, sendo este considerado servidor nessa transação.

2.4.4 A arquitetura de camadas do modelo TCP/IP

O modelo OSI é constituído por sete camadas que representam uma divisão funcional das tarefas necessárias para implementar uma rede. É uma arquitetura conceptual criada pelo *International Organization for Standardization* (ISO), usada para mostrar como vários protocolos e tecnologias se encaixam na implementação da rede. No entanto, não é o único modelo que tenta fazer esta divisão. Na verdade o modelo TCP/IP foi criado antes do modelo de referência OSI, mas ambos são bastante parecidos na sua constituição. Referia-se que o OSI é uma arquitetura conceptual que não tem implementação na prática, sendo a arquitetura TCP/IP aquela cuja implementação prática mais se aproxima da arquitetura OSI (mas existem outras). Existem poucas diferenças na correspondência entre as camadas. Mas visto que o modelo OSI é mundialmente reconhecido, é comum usá-lo para explicar o modelo TCP/IP.

O modelo TCP/IP usa quatro camadas que logicamente equivalem às seis camadas de topo do modelo OSI, demonstrado na ilustração 11. A camada física não é contemplada pelo modelo TCP/IP, porque a camada de ligação de dados é considerada o ponto em que a pilha TCP/IP se interliga com o *hardware* de rede subjacente. Seguidamente é feita a descrição das camadas a começar por baixo, e a devida correspondência com o modelo OSI.

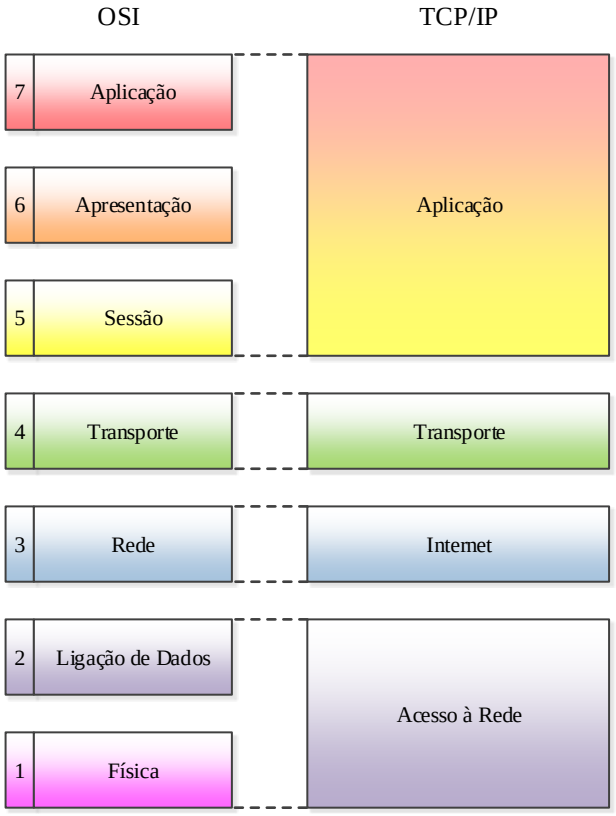

Ilustração 11 - Comparação entre o modelo de referência OSI e o modelo TCP/IP

2.4.4.1 Camada de Acesso à Rede

Esta camada, tal como o nome indica, tem a função de fornecer aos protocolos, que estão a correr a níveis mais elevados, o acesso à rede. Esta camada não inclui nenhum dos protocolos principais do TCP/IP, mas é uma parte importante da arquitetura. Equivale à segunda camada do modelo OSI (camada de ligação de dados). No entanto, em muitas intranets TCP/IP, não existem protocolos do TCP/IP a correr nesta camada, simplesmente porque não é necessário. Como exemplo, considera-se uma rede Ethernet. Neste caso, os protocolos da camada de acesso à rede não são usados porque a rede Ethernet faz o tratamento das funcionalidades desta camada.

A camada de acesso à rede é responsável por estabelecer ligações ponto a ponto entre dois *routers* sucessivos ou entre router e anfitriões (estas ligações podem ser feitas através de *switches* ou *hubs*). Cada ligação pode usar um protocolo diferente como PPP (*Point-to-Point Protocol*) ou HDLC (*High-level Data Link Control*), entre outros, e um meio de comunicação diferente como cabos de par entrelaçado, fibra ótica, etc..

A transmissão de informação nesta camada é feita através de *frames* (meio de encapsulamento de mais baixo nível da arquitetura TCP/IP, visto que é a partir deste que se retiram os dados para cada uma das outras camadas de forma crescente e ordenada), usando o endereço MAC (*Media Access Control*) dos dispositivos como meio de identificação da origem e do destino. O MAC é composto por 48 bits e representado por seis grupos de dois dígitos hexadecimais. Para que as *frames* possam ser enviadas para um determinado destino, a estação que está a fazer o envio, precisa de encontrar o endereço MAC que corresponde ao endereço IP de destino incluído no pacote da *frame*. Este mapeamento é feito através do *Address Resolution Protocol* (ARP) que consiste no seguinte: uma estação tem uma tabela ARP que mapeia endereços IP com endereços MAC. Quando um anfitrião tem um pacote para enviar ou retransmitir, este tenta encontrar o endereço IP na tabela ARP, para extrair o MAC correspondente. No caso de não existir nenhuma entrada na tabela correspondente a esse endereço, este faz difundir um pacote ARP com informação sobre o IP destino pretendido. A estação distinta por este endereço responde com um pacote do qual é feita a extração do endereço MAC, inserindo uma nova linha na sua tabela ARP com o mapeamento correspondente. Esta entrada na tabela ARP é temporária, se passado um determinado tempo sem tráfico passar para (ou ser recebido de) esta estação, a estrada nesta tabela é removida e o processo é reiniciado.

A camada de acesso à rede é ainda composta por duas subcamadas: A subcamada de controlo de ligações lógicas, que lida com controlo de erros e de fluidez, e a subcamada de controlo de acesso ao meio, que tem a responsabilidade de coordenar quando é as estações podem transmitir e em que formato.

2.4.4.2 Camada de Internet

Esta camada corresponde à camada de rede do modelo OSI. É responsável pelas tarefas típicas da terceira camada, como endereçamento, empacotamento de dados, manipulação e entrega e roteamento. É nesta camada que está situado o protocolo IP, sendo este o mais relevante. A nova versão deste protocolo é chamada de IP versão 6, e irá ser usada para suportar as crescentes necessidades presentes e futuras da Internet.
A versão corrente do IP (IPv4) é baseada em trocas de datagramas sem estabelecer conexões. Isto porque uma rede pode ser dinâmica, está constantemente a mudar, e os pacotes enviados para o mesmo destino, podem seguir caminhos diferentes e chegar fora de ordem. No caso de chegarem de facto fora de ordem, a reordenação seria feita por outra camada.
No IP, cada nó tem que decidir qual o melhor caminho para enviar cada pacote, e cada um destes pacotes transporta informação sobre o endereço destino.
Seguidamente é descrito um exemplo de funcionamento de transporte de pacotes com o IP:

- Um determinado anfitrião recebe dados na camada de aplicação através de um processo específico. Formata estes dados para transmissão, estabelece uma sessão com um outro anfitrião destino, segmenta os dados e transfere-os para a camada de transporte;
- A camada de transporte recebe os dados da camada de aplicação, adiciona os dados necessários ao pacote, como a porta do endereço de origem e destino, e transfere-o para a camada de internet;
- A camada de internet adiciona os dados referentes à terceira camada do modelo OSI, como o IP da origem e destino, e passa o pacote para a camada de acesso à rede para uma transferência ponto a ponto;

- A camada de acesso à rede recebe o pacote e adiciona os dados relacionados com a segunda camada do modelo OSI, como o endereço MAC da origem e destino, bits redundantes de controlo de erros, etc.. Apos a composição do pacote até aqui, este é passado para a camada física do modelo OSI, que no modelo TCP/IP é controlada pelo *hardware* do computador.
- A camada física recebe a *frame* originada das adições da camada de acesso á rede, formata os bits para transmissão, como o tipo de codificação digital ou esquema de modulação (depende do meio), e começa a transmissão dos bits para o anfitrião destino;
- A camada física do *router*, recebe os símbolos enviados, converte-os para bits e transfere-os para a sua camada de acesso à rede;
- A camada de acesso à rede do *router* agrupa os bits em *frames* e verifica se há erros. Se existirem erros, dependendo do equipamento, pode utilizar uma forma de correção de erros, ou utilizar o *Automatic Repeat reQuest* (ARQ) para pedir o reenvio da *frame* ao anfitrião de origem.
- Uma vez que a *frame* seja corretamente recebida na segunda camada do router, a informação referente a esta camada é removida e o pacote (que estava dentro da *frame*) é transferido para a terceira camada;
- A camada de internet do *router* recebe o pacote, lê o endereço IP do destino e consulta a sua tabela de roteamento. Desta tabela extrai a interface para a qual o pacote deve ser enviado, passa novamente para a segunda camada para adicionar a informação necessária para criar a nova *frame*, e envia-a para a camada física dando início à transmissão para o próximo router. Este processo repete-se até a *frame* alcançar o anfitrião destino.
- A *frame* ao alcançar o anfitrião destino, é passada para a camada de acesso à rede onde é extraído pacote, que por sua vez sobe para a camada de internet,

esta também extraí a sua informação e passa os dados para a camada de transporte.

- Na camada de transporte existem dois caminhos possíveis:
  - Se o protocolo usado é o TCP, este verifica se existem erros, se uma erro for detetado, não é enviada uma resposta de receção bem-sucedida, o que leva a camada de transporte do anfitrião de origem a atingir o tempo limite de espera e a recomeçar a transmissão do pacote. Após a receção bem-sucedida, é ainda verificada a sequência correta dos pacotes, e caso seja necessário, são feitas correções. Por fim são removidos os dados referentes à camada de transporte e o pacote é transferido para a camada de aplicação;
  - Se o protocolo usado é o UDP, são apenas removidos os dados referentes à camada de transporte e o pacote é transferido para a camada de aplicação;
- A camada de aplicação do anfitrião destino reagrupa os vários pacotes recebidos pela camada de transporte, faz as necessárias conversões de dados e transfere-os para o processo aplicacional específico em causa.

2.4.4.3 Camada de Transporte

A função principal desta camada é facilitar a comunicação numa rede e fornecer qualidade de serviço. Está encarregue de permitir que ligações lógicas sejam feitas entre dispositivos, para que possa haver a troca de dados com ou sem segurança. É também nesta camada que é feita a identificação do processo da aplicação específica de origem e de destino.

O nome que se dá ao formato da mensagem da camada de transporte do modelo TCP/IP é segmento e pode ser de dois tipos, *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP).

O TCP é orientado à conexão, o que significa que é necessário estabelecer uma conexão antes de haver troca de dados. Usa técnicas de deteção de erros como o *Cyclic Redundancy Check* (CRC), ou o *Positive Acknowledgement with Retransmission* (PAR) para retransmissão automática de pacotes corrompidos, para fornecer fiabilidade nos pacotes entregues. Um pacote quando é recebido, pode ou não conter erros, se não forem detetados erros, é enviada uma mensagem de receção bem-sucedida. No caso de serem encontrados erros no pacote, esta mensagem não é enviada, o que leva o anfitrião origem a atingir o tempo limite de espera pela confirmação de receção bem-sucedida, e a proceder a uma retransmissão do pacote.

O TCP tem como funções principais assegurar o seguinte:

- Dados enviados com fiabilidade;
- Pacotes recebidos na sequência correta;
- Pacotes perdidos são detetados e corrigidos;
- A duplicação de pacotes é evitada.

Assim sendo, este modo de utilização da camada de transporte e ideal para serviços que necessitam de dados confiáveis.

O UDP é não orientado à conexão, e não fornece serviços de fiabilidade. Os dados são transmitidos sem ser necessário o estabelecimento de uma conexão. Apesar de não proporcionar a fiabilidade nos pacotes, tem a vantagem de não introduzir atrasos no envio dos pacotes e também é importante denotar que a confiança nos dados pode ser atingida através de outros meios pelas camadas superiores. Este é o modo ideal para serviços que resistem a algum nível de erros.

A camada de transporte do modelo TCP/IP tem alguns elementos que podem ser considerados da camada de sessão do modelo OSI. Como exemplo, o TCP estabelece

ligações que podem persistir por longos períodos de tempo. Estas geram alguma confusão visto que também podem ser consideradas sessões.

2.4.4.4 Camada de Aplicação

Esta é a camada de topo do TCP/IP e é uma camada ampla, visto que abrange a quinta, sexta e sétima camada do modelo OSI. É aqui que vão ser executadas todas as funcionalidades da camada de sessão, apresentação e aplicação do modelo OSI, transparecendo a ideia que com esta abrangência se perde detalhe em comparação ao modelo referência. No entanto, reflete melhor a natureza desfocada das divisões entre as funções destas três camadas. Na verdade, em certos casos é bastante difícil atribuir uma camada específica a um determinado protocolo.

A camada de aplicação é a que incluí a maior variedade de protocolos. Incluí alguns como o HTTP, FTP e SMTP para fornecer serviços aos utilizadores finais, assim como o SNMP, DHCP e DNS para fornecer serviços administrativos, entre outros. É importante salientar que é nesta camada que a arquitetura proposta nesta dissertação se encaixa, tirando proveito de funcionalidades de protocolos de camadas inferiores, como o TCP para estabelecer ligações ou o IP para endereçamento.

2.5 Comparação entre uma arquitetura Cliente-Servidor e Peer-to-peer em um sistema de partilha de ficheiros

É pertinente questionar o porquê de transitar de uma arquitetura cliente-servidor, para uma arquitetura peer-to-peer, visto que nos dias correntes, os *sites* de redes sociais que têm um sistema de transferência de ficheiros usam uma arquitetura cliente-servidor. Portanto, é importante demonstrar as razões que incentivam a proceder à adaptação do sistema.

2.5.1 Hardware

Hardware é uma das grandes razões pelo qual é aconselhável a transição. A gestão e custos associados com servidores de ficheiros são muito elevados. O que normalmente

acontece é os *sites* não fornecerem serviços de alocação de ficheiros, em vez disso, usam uma associação com parceiros dedicados a este tipo de sistema. A partilha é normalmente concretizada usando uma arquitetura cliente-servidor, em que a transferência é feita diretamente entre os utilizadores.

Com uma arquitetura peer-to-peer, é possível resolver estes dois problemas, partilha e alocação, com apenas um sistema. As máquinas dos utilizadores em si são os servidores de ficheiros, logo não existem grandes custos adicionais relacionados com a alocação. Quanto à partilha, será demonstrado na secção seguinte as vantagens de performance no sistema em causa.

2.5.2 Performance

Existem várias diferenças entre uma arquitetura cliente-servidor e uma arquitetura peer-to-peer, como já foi evidenciado ao longo da dissertação. No entanto não se pode assumir que uma arquitetura é melhor do que a outra em todas as situações. Na verdade, a escolha de qual destas arquiteturas deve ser utilizada, é bastante dependente do ambiente do sistema em causa. Como é possível observar na ilustração 12, existem múltiplas variáveis que influenciam o caminho a tomar. É necessário ter compreensão total das necessidades atuais e futuras do sistema.

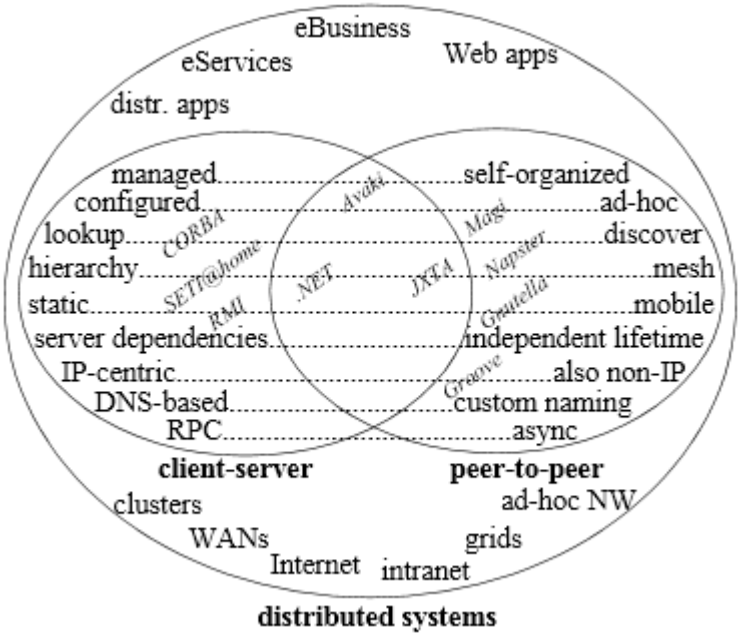

Ilustração 12 - Cliente-Servidor versus Peer-to-peer [Milojicic et al. 2002]

No sistema proposto, a variável mais relevante, é a capacidade de transferência entre os utilizadores. Pretende-se que estes tenham uma forma intuitiva e eficaz de partilhar grandes quantidades de dados, geralmente ficheiros, entre si.

Assim sendo, no sistema proposto, é bastante fácil provar que um sistema peer-to-peer tem efetivamente melhor performance na transferência de ficheiros, em comparação a uma arquitetura cliente-servidor. Isto porque, nos sistemas atuais de partilha de ficheiros em *sites* de redes sociais, as variáveis que afetam a performance de transferência em arquiteturas cliente-servidor, são as mesmas que afetam a performance em arquiteturas peer-to-peer. Os utilizadores tanto são clientes como são servidores, e isto simplifica bastante a demonstração.

Para demonstrar este facto, será efetuada uma comparação entre diversas simulações, de arquitetura cliente-servidor e peer-to-peer, de transferência de um ficheiro de 1000 Mbit entre dois utilizadores, em três ambientes diferentes. No primeiro ambiente os utilizadores têm uma capacidade de carga de 10 Mbit/s e uma capacidade de descarga

de 90 Mbit/s. No segundo ambiente, a capacidade de carga e descarga é de 50 Mbit/s. No terceiro a capacidade de carga é de 90 Mbit/s e a capacidade de descarga é de 10 Mbit/s.

A ferramenta escolhida para efetuar as simulações é uma aplicação desenvolvida em JAVA que consoante os parâmetros fornecidos, gera gráficos em função do tempo decorrido no processo, e do progresso da transferência.

O fator determinante é a **velocidade de transferência média**, que será calculada através da relação entre o tamanho do ficheiro (em Mbit) a transferir e o tempo (em segundos) necessário para completar a transferência. Para efeitos de legitimidade dos resultados obtidos, todos os utilizadores envolvidos em cada simulação têm a mesma capacidade teórica de carga e descarga.

2.5.2.1 Ambiente nº1, carga 10 Mbit/s, descarga 90 Mbit/s

2.5.2.1.1 Simulação Cliente-Servidor

Como é possível observar no gráfico demonstrado na ilustração 13, utilizando a arquitetura cliente-servidor, foram necessários 100 segundos para transferir um ficheiro de 1000 Mbit entre o utilizador 1 e o utilizador 2.

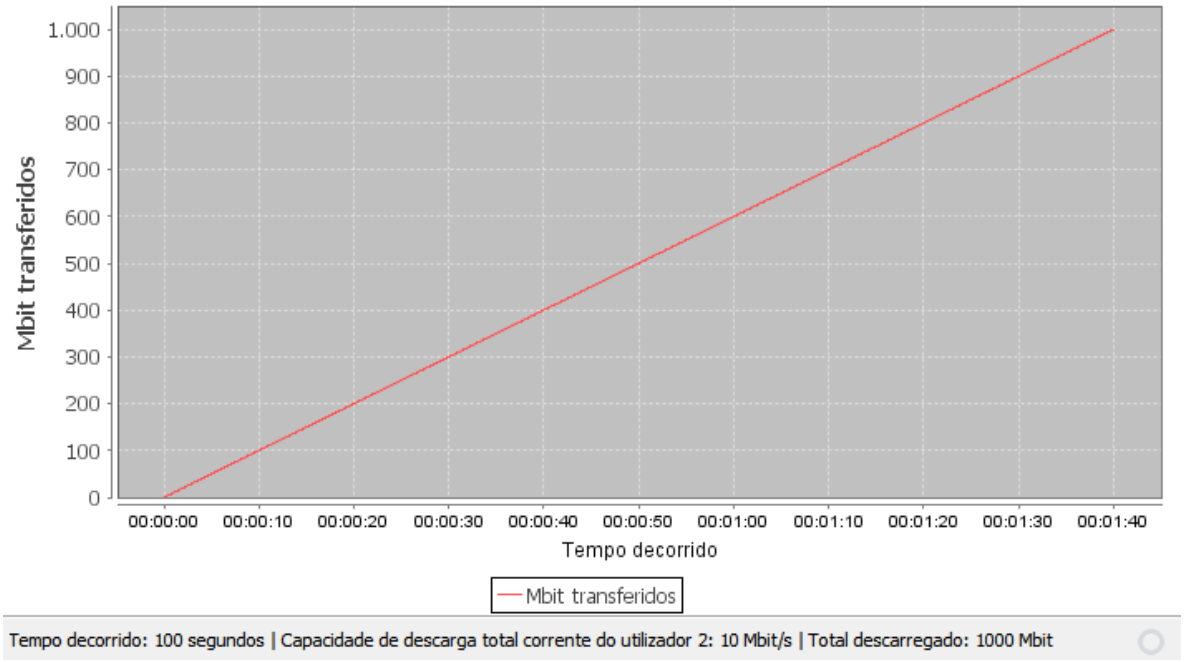

Ilustração 13 – Simulação do ambiente nº1 ao utilizar uma arquitetura cliente-servidor

Portanto, calculando a relação entre o **tamanho de ficheiro** e o **tempo necessário para a transferência**, é possível obter a **velocidade de transferência média** fornecida por este sistema:

1000 / 100 = 10 Mbit/s.

Assim sendo, na melhor das hipóteses, assumindo que não ocorreram quaisquer contratempos durante o processo, a velocidade de transferência máxima teórica é 10 Mbit/s. Pode então concluir-se que, neste ambiente, a velocidade de transferência nunca vai ultrapassar a capacidade de carga do utilizador 1.

2.5.2.1.2 Simulação Peer-to-peer

Na segunda simulação será usado um ambiente teórico comparável ao comportamento real da arquitetura peer-to-peer, assumindo que não ocorrem contratempos no processo. Serão demonstrados três gráficos que visam representar o melhor cenário possível, o

cenário normal e o pior cenário possível. A transferência será iniciada a partir do utilizador 1 para o utilizador 2, tal como na primeira simulação. No entanto, a variante nas diferentes situações é a quantidade de *peers* disponíveis para carregar o ficheiro, ou o tempo que demoram a ficar disponíveis.

No melhor cenário possível, a partir do momento inicial, existem peers disponíveis suficientes para tirar proveito de toda a capacidade de descarga do utilizador 2.

No cenário normal, a quantidade de *peers* irá aumentar ao longo do processo. Para simular este comportamento, a cada 2 segundos será adicionado mais um *peer* ou semente para ajudar a fazer a transferência.

No pior cenário possível existe apenas um *peer*, neste caso o utilizador 1, a fazer a transferência de todo o ficheiro para o utilizador 2. Os resultados podem ser observados nos gráficos demonstrados nas ilustrações 14, 15 e 16.

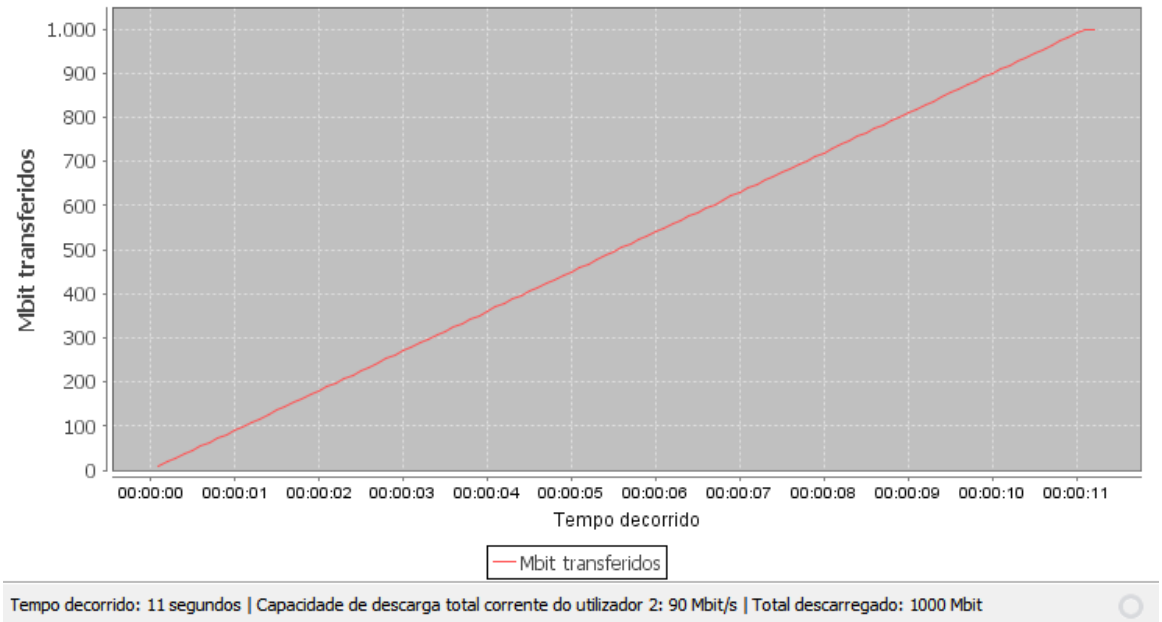

Ilustração 14 - Simulação do ambiente nº 1 ao utilizar uma arquitetura peer-to-peer, no melhor cenário possível

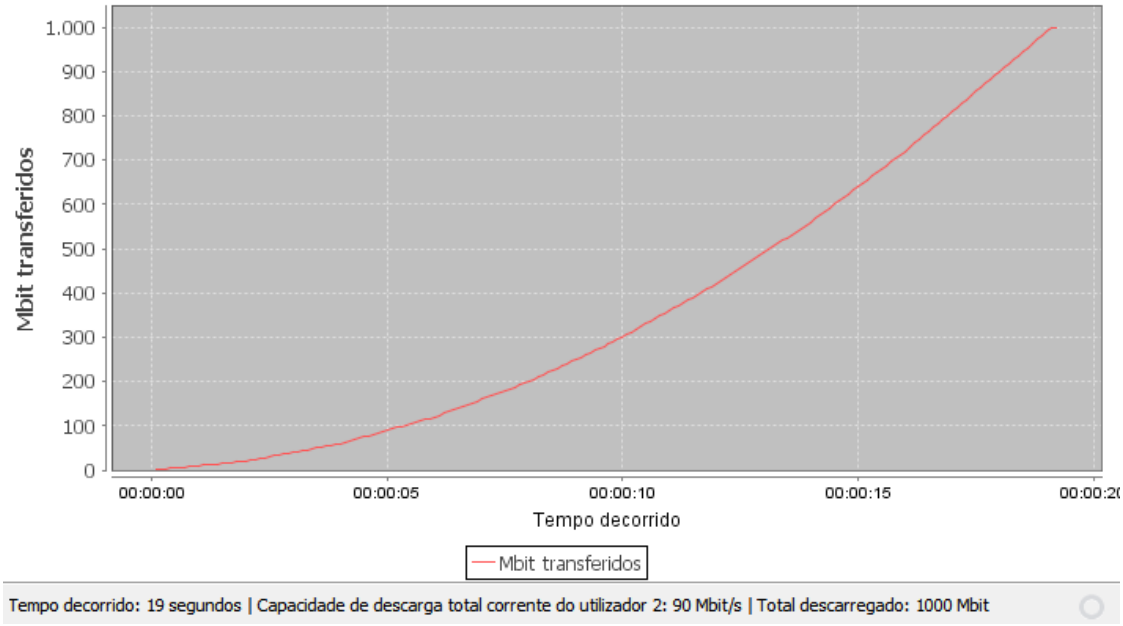

Ilustração 15 - Simulação do ambiente nº 1 ao utilizar uma arquitetura peer-to-peer, no cenário normal

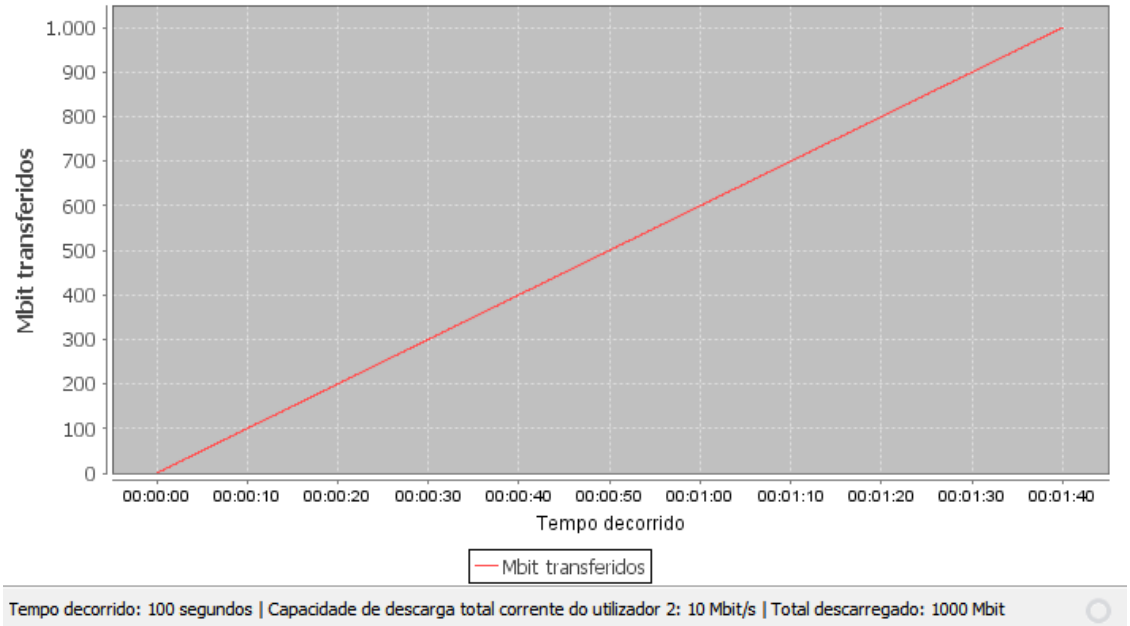

Ilustração 16 - Simulação do ambiente nº 1 ao utilizar uma arquitetura peer-to-peer, no pior cenário possível

Como é possível observar nos resultados obtidos no melhor cenário possível, foram necessários 11 segundos para fazer a transferência do ficheiro de 1000 Mbit. A capacidade de descarga máxima de 90 Mbit/s do utilizador 2 foi atingida desde o momento inicial e durante todo o processo. Assim foi possível obter a seguinte **velocidade de transferência média**:

1000 / 11 = 90 Mbit/s

Segundo os resultados obtidos no cenário normal, foram necessários 19 segundos para fazer a transferência do ficheiro de 1000 Mbit. A velocidade de descarga foi aumentando ao longo do processo, conforme mais *peers* se juntavam a cada 2 segundos para ajudar a fazer a transferência, atingindo o valor máximo de 90 Mbit/s, sendo este a capacidade máxima de descarga do utilizador 2. Assim foi possível obter a seguinte **velocidade de transferência média**:

1000 / 19 = 52 Mbit/s

Ao observar os resultados obtidos no pior cenário possível, foram necessários 100 segundos para fazer a transferência do ficheiro de 1000 Mbit. Um resultado idêntico ao da simulação cliente-servidor. Assim foi possível obter a seguinte **velocidade de transferência média**:

1000 / 100 = 10 Mbit/s

É necessário salientar que, estes valores podem variar dependendo do tempo que o processo demora a juntar outros *peers* à transferência do ficheiro para aumentar a velocidade de descarga. Num ambiente real, é possível alcançar a velocidade máxima de descarga em menos ou mais tempo, dependendo do número de *peers* disponíveis. Quanto mais *peers* existirem disponíveis para carregar, menos tempo será necessário para alcançar a velocidade máxima de descarga e consequentemente menos tempo será necessário para fazer a descarga do ficheiro.

Podemos assim concluir da comparação dos resultados obtidos neste ambiente, que a arquitetura peer-to-peer tem uma performance consideravelmente melhor do que

arquitetura cliente-servidor. O pior dos casos seria existir apenas um *peer* disponível para carga, o que significa que o resultado final seria igual nas duas arquiteturas.

2.5.2.2 Ambiente nº2, carga e descarga 50 Mbit/s

2.5.2.2.1 Simulação Cliente-Servidor

Como é possível observar no gráfico da ilustração 17, foram necessários 20 segundos para fazer a transferência do ficheiro de 1000 Mbit.

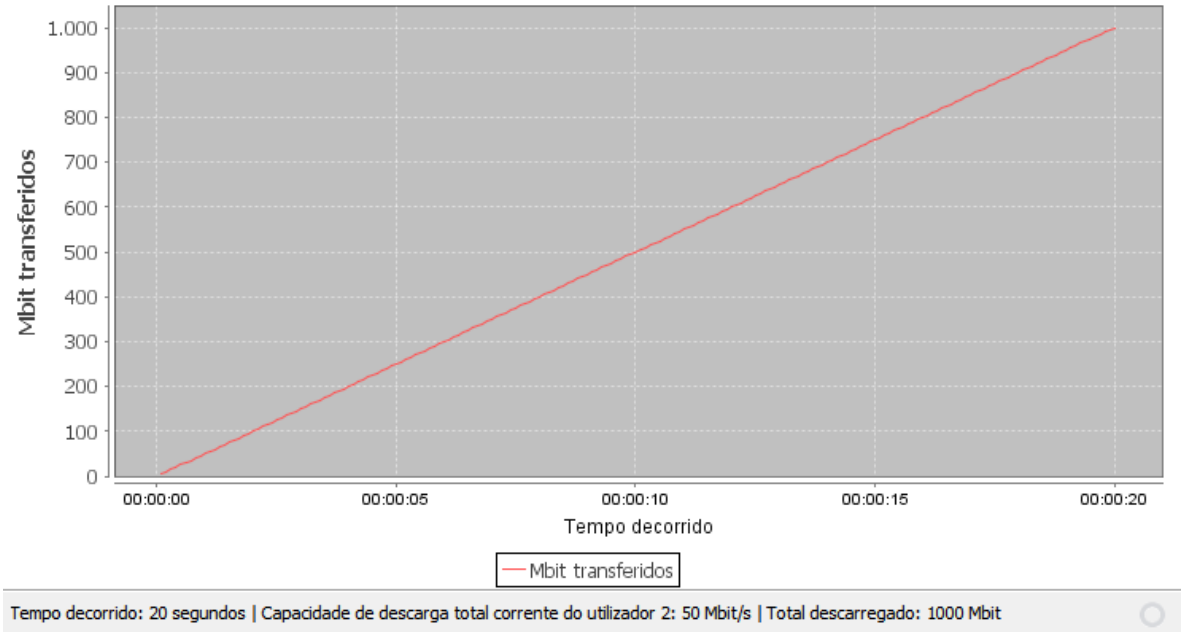

Ilustração 17 - Simulação do ambiente nº 2 ao utilizar uma arquitetura cliente-servidor

Portanto, ao executar o cálculo da **velocidade de transferência média**, obtém-se o seguinte resultado:

1000 / 20 = 50 Mbit/s

Usando a arquitetura cliente-servidor neste ambiente obtivemos uma **velocidade de transferência média** igual à velocidade de carga. O que significa que a largura de banda de descarga foi usada na totalidade devido à capacidade de carga e descarga serem iguais em ambos os utilizadores.

2.5.2.2.2 Simulação Peer-to-peer

Observando os resultados obtidos no gráfico da ilustração 18, na simulação peer-to-peer neste ambiente, foram necessários 20 segundos para fazer a transferência do ficheiro de 1000 Mbit.

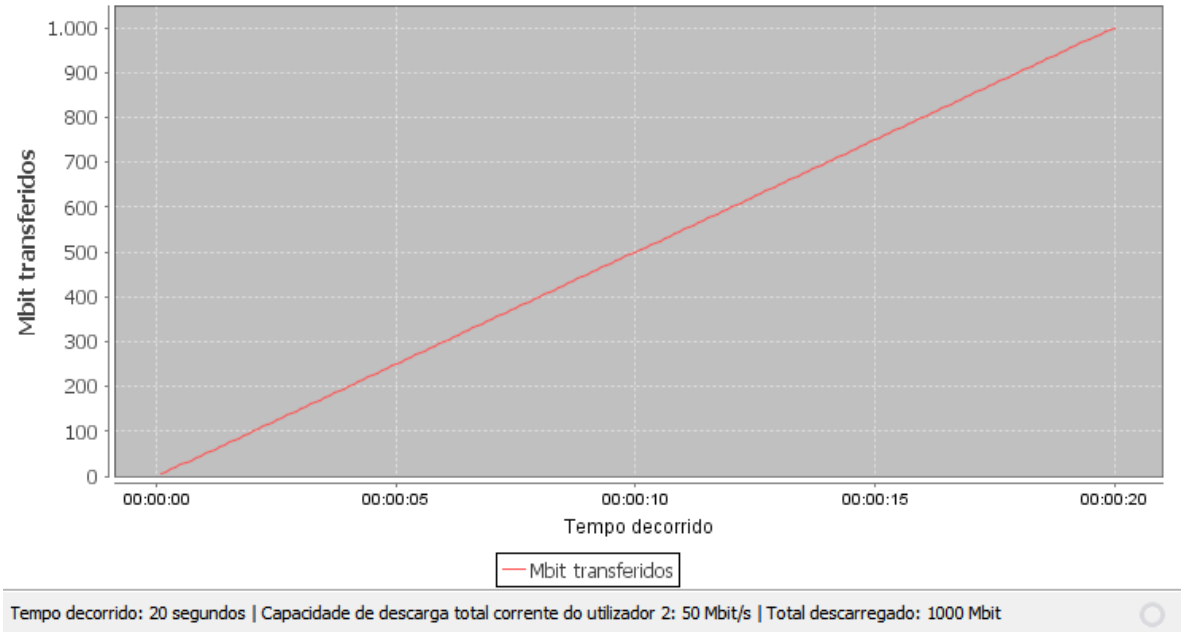

Ilustração 18 - Simulação do ambiente nº 2 ao utilizar uma arquitetura peer-to-peer

Executando novamente o cálculo da **velocidade de transferência média** nesta simulação, obtém-se:

$$1000 / 20 = 50 \text{ Mbit/s}$$

O resultado foi idêntico ao da arquitetura cliente-servidor. O que significa que quando a rede em que o ambiente se encaixa, fornece uma capacidade de carga igual à capacidade de descarga, a utilização de uma arquitetura peer-to-peer é desnecessária visto que é necessário apenas um utilizador para tirar proveito de toda a capacidade de descarga.

2.5.2.3 Ambiente nº3, carga 90 Mbit/s, descarga 10 Mbit/s

2.5.2.3.1 Simulação Cliente-Servidor

Ao observar os resultados obtidos no gráfico da ilustração 19, foram necessários 100 segundos para transferir o ficheiro de 1000 Mbit.

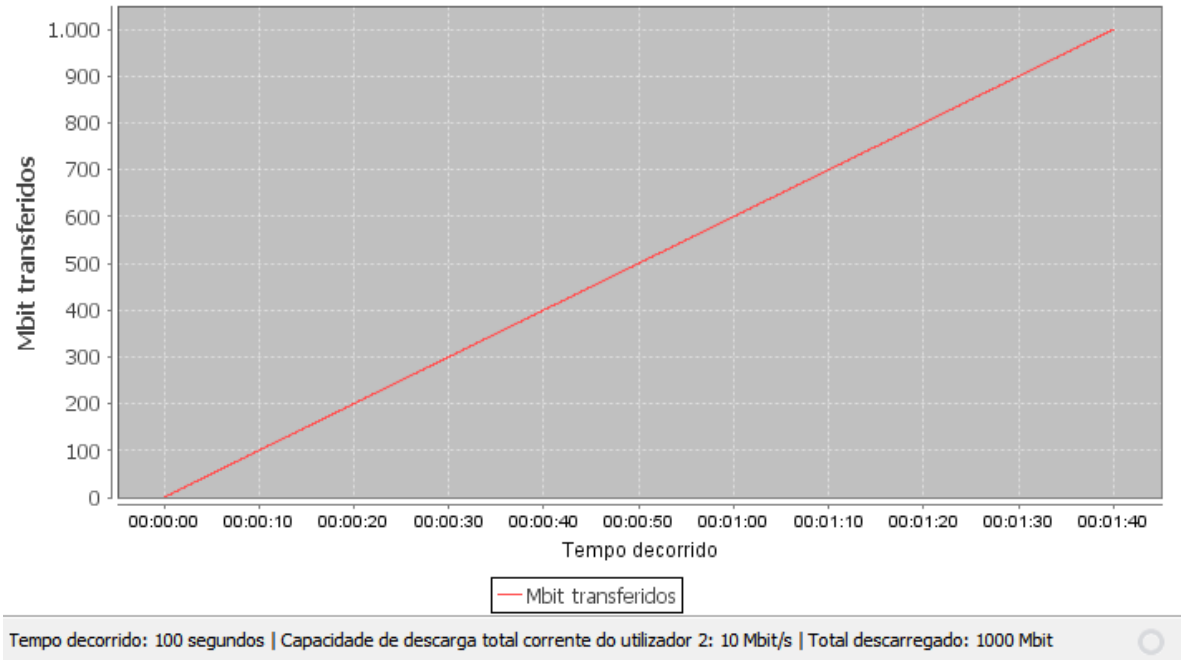

Ilustração 19 - Simulação do ambiente nº 3 ao utilizar uma arquitetura cliente-servidor

Ao executar o cálculo da **velocidade de transferência média**, é obtido o seguinte resultado:

1000 / 100 = 10 Mbit/s

Nesta simulação também foram obtidos os 10Mbit/s. No entanto, o fator que limitou a velocidade de transferência, não foi a capacidade de carga, mas sim a capacidade de descarga. Na verdade neste ambiente, o utilizador 1 poderia estar simultaneamente a carregar para 9 utilizadores e tirar proveito total da capacidade de descarga de cada um deles. Visto que neste ambiente a capacidade de carga é 9 vezes superior à capacidade de descarga.

2.5.2.3.2 Simulação Peer-to-peer

Observando os resultados obtidos no gráfico da ilustração 20, podemos verificar que também foram necessários 100 segundos para fazer a transferência do ficheiro de 1000 Mbit.

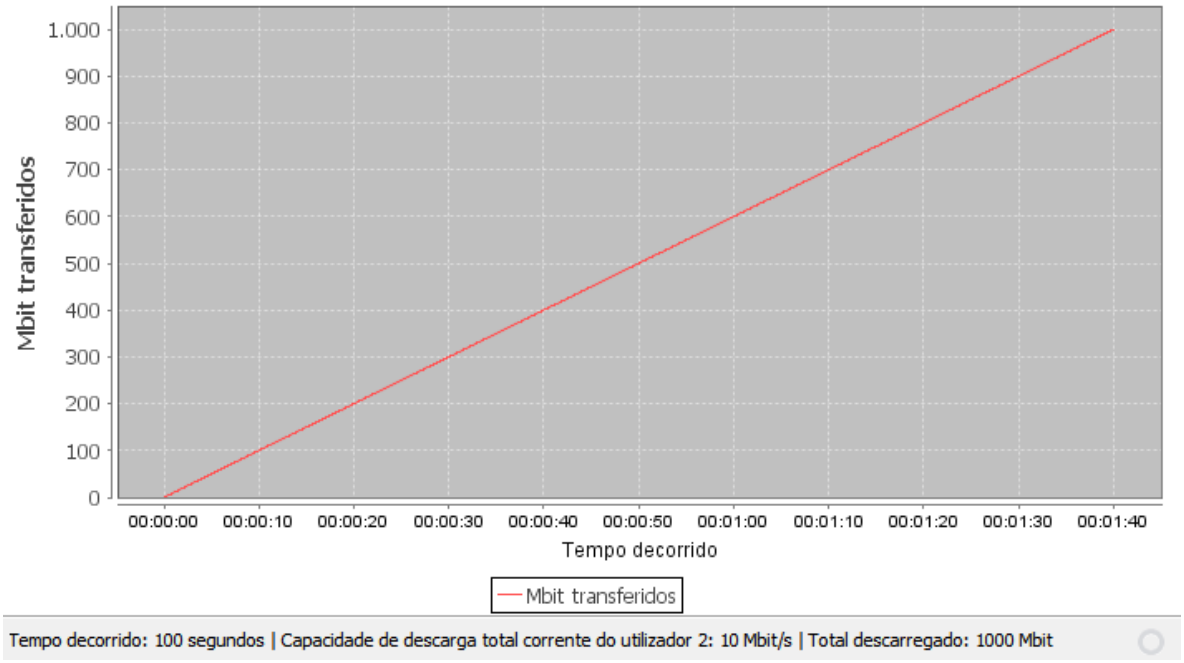

Ilustração 20 - Simulação do ambiente nº 3 ao utilizar uma arquitetura peer-to-peer

Portanto, segundo o cálculo para obter a **velocidade de transferência média**, é obtido o seguinte resultado:

1000 / 100 = 10 Mbit/s

A **velocidade de transferência média** é 10 Mbit/s. Mais uma vez, foi obtido um resultado idêntico ao da simulação anterior. Isto significa que, também num ambiente em que a rede fornece uma capacidade de carga superior à capacidade de descarga, não

existe necessidade de utilizar uma arquitetura peer-to-peer para um sistema de partilha de ficheiros.

2.5.3 Conclusões das simulações

As simulações efetuadas tiveram como objetivo comprovar que uma arquitetura P2P, em comparação a uma arquitetura Cliente-Servidor, é mais eficiente quando utilizada em um sistema de transferência de ficheiros. No entanto, revelou-se que de facto a eficiência é superior mas apenas quando a capacidade de descarga da largura de banda é superior à capacidade de carga, e o número de *peers* ou sementes é maior do que um. Em todos os outros casos, verificou-se que a eficiência é idêntica nas duas arquiteturas. Constata-se também que quanto maior for a diferença entre a capacidade de carga e descarga, maior é o beneficio do sistema P2P. Ou seja, numa rede em que a capacidade de descarga é o dobro da capacidade de carga, teoricamente bastam duas sementes ou *peers* para ocupar toda a largura de banda para o processo de descarga. No entanto se a rede fornecer dez vezes mais capacidade de descarga do que carga, já são necessárias dez sementes ou *peers* para tirar proveito de toda a largura de banda para descarga. Isto é um grande benefício porque significa que é possível sacrificar menos largura de banda para a capacidade de carga e continuar a tirar proveito de toda a capacidade de descarga se o número de utilizadores do sistema P2P for suficientemente elevado.

3 - Descrição da Arquitetura Proposta

Neste capítulo será descrita a arquitetura proposta usando ilustrações e a devida descrição para auxiliar na compreensão.

Para proceder à explicação pode ser considerada a visão de um utilizador que se depara com o *site* de rede social e pretende fazer partilha de ficheiros com os seus contactos.

É também importante salientar que serão feitas múltiplas referências às seguintes listas, que são necessárias e têm papéis fulcrais na arquitetura:

- **Lista de contactos,** com o qual é feita a partilha de ficheiros e outras informações;
- **Lista de ficheiros partilhados**, que contem uma listagem dos ficheiros que estão a ser descarregados, carregados ou simplesmente disponíveis para partilha. É importante salientar que não é necessário dividir estes tipos de transferências em diferentes listas se forem usados os devidos filtros (ficheiros completos ou incompletos, ficheiros publicados pelo utilizador, ficheiros a ser correntemente carregados ou descarregados, etc.). O facto de usar apenas uma lista para ficheiros também ajuda na gestão e intuitividade;
- **Lista de histórico**, para que seja possível consultar o histórico de ficheiros adicionados à lista de ficheiros partilhados, com a data e a forma como foram adicionados;

Este sistema deve também conter uma funcionalidade, que permita fazer a configuração das múltiplas variáveis necessárias para o seu bom funcionamento. Na rede social, deve ser possível alterar pelo menos as seguintes configurações:

- Habilitar ou desabilitar o sistema de transferência de ficheiros;
- Largura de banda máxima usada pelo sistema;
- Número de ligações máximas;
- Encriptação da troca de dados;

- Habilitar ou desabilitar o registo de histórico;
- Localizações físicas no disco (descargas completas e incompletas, ficheiros ".torrent");

Existem ainda mais configurações opcionais, muitas das quais em que a sua ativação é aconselhada. Devem ser consultadas as especificações BitTorrent e adaptar ao sistema em causa.

3.1 Autenticação no sistema

Sendo este um *site* de redes sociais, é necessário proceder à autenticação para ter acesso total às funcionalidades fornecidas. A ilustração 21 demonstra esse mesmo processo.

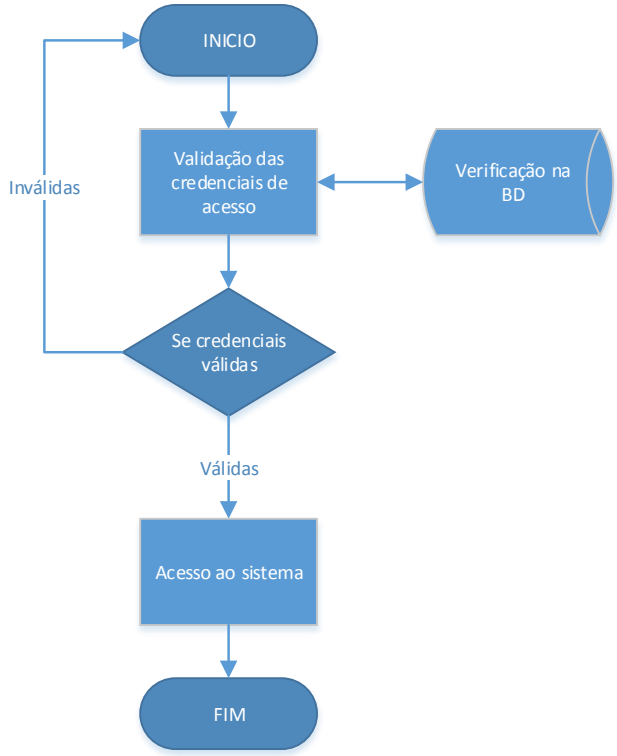

Ilustração 21 – Autenticação no sistema

O conteúdo do processo de autenticação não é estritamente relevante para a arquitetura em si. Pois embora seja necessária a autenticação, a forma como o utilizador é autenticado

pode variar. O importante é existir a autenticação do utilizador no sistema para desbloquear o resto das funcionalidades dependentes de uma identidade.

3.2 Publicação de ficheiros

Após autenticação no sistema, o utilizador deve ter a possibilidade de aceder a um determinado número de funcionalidades, sendo a partilha de ficheiros a parte relevante para a arquitetura proposta. Uma das funcionalidades mais importantes consiste no processo de publicação de ficheiros no sistema para que estes possam ser partilhados. Analisando a ilustração 22, seguidamente é feita uma descrição do processo:

- O utilizador acede à funcionalidade de publicação de ficheiros;
- É apresentada uma forma de selecionar um ou mais ficheiros da sua máquina local que pretende publicar;
- Após a seleção do ficheiro a publicar, devem ser enviados os parâmetros necessários para gerar o ficheiro ".torrent" que ficará alocado na máquina do utilizador num determinado caminho, configurável na funcionalidade "**Configuração do sistema**";
- Seguidamente, antes do ficheiro ficar pronto para partilha, deve ser selecionado o tipo de permissões para acesso a este ficheiro, Este processo corresponde a efetuar a escolha do grupo de utilizadores que têm permissão para visualizar e descarregar este ficheiro da sua **lista de ficheiros partilhados**;
    - Se o utilizador não pretender definir permissões, são aplicadas as permissões pré-definidas, que devem ser configuráveis na funcionalidade "**Configuração do sistema**".
    - Se o utilizador pretender definir as permissões, devem ser apresentadas formas para facilitar esta tarefa ao utilizador. Sugere-se a utilização de grupos de utilizadores e funcionalidades de pesquisa.

- Por fim, é finalmente adicionada uma entrada do ficheiro na **lista de ficheiros partilhados**, para que possa ser descarregado por, ou carregados para, outros contactos. É também registada a informação do ficheiro ".torrent", para que o utilizador tenha a possibilidade de fazer a descarga do mesmo ficheiro noutras máquinas. Deverá também ser possível gerar novamente o ficheiro ".torrent" a partir desta informação.

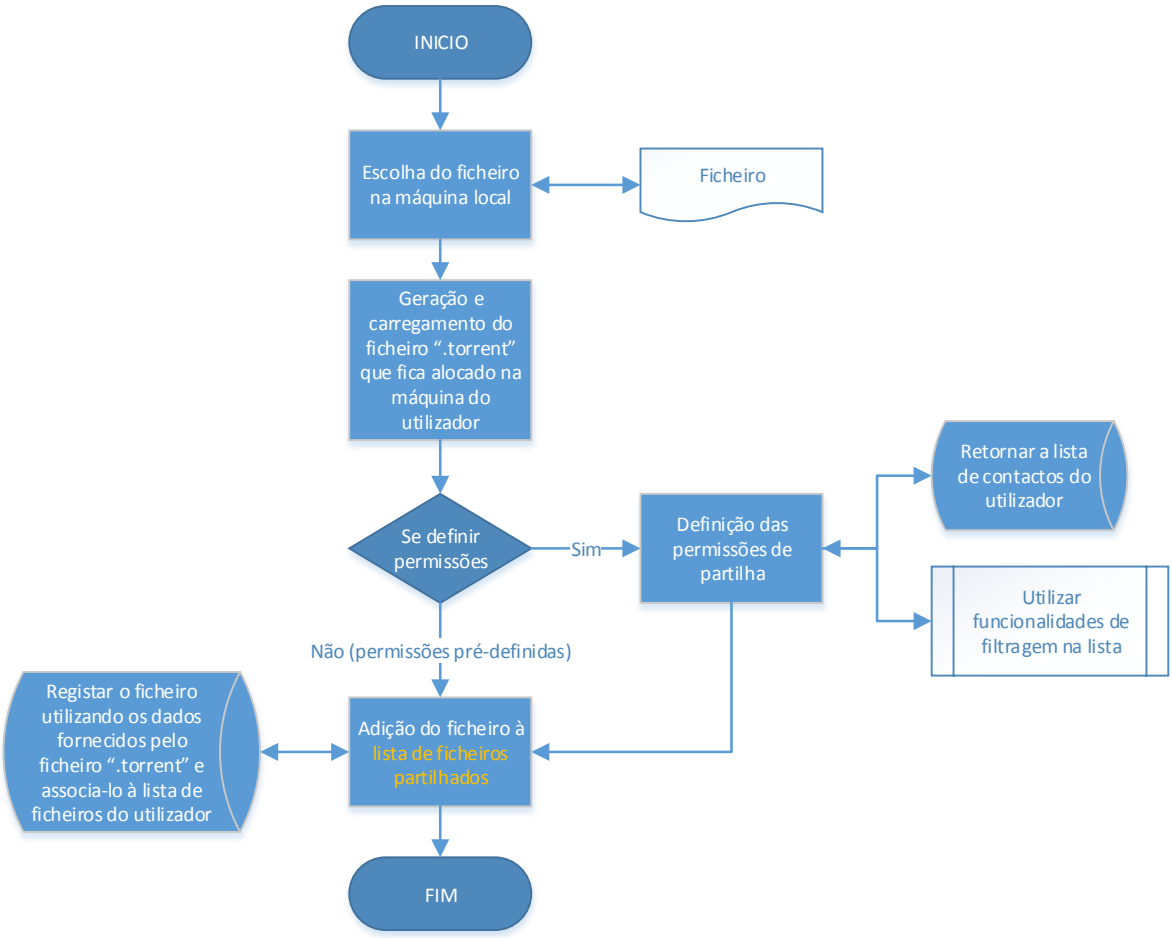

Ilustração 22 – Publicação de ficheiros para partilha no sistema

3.3 Partilhar ficheiros

A partilha de ficheiros é uma funcionalidade que não deve ser confundida com a publicação de ficheiros, embora as duas se possam unir para criar apenas uma.

Esta funcionalidade consiste na possibilidade de intencionalmente, por ação do utilizador, sugerir a descarga de um ficheiro já publicado na sua **lista de ficheiros partilhados** a um qualquer contacto.

Seguidamente é descrito este processo com o auxílio da ilustração 23:

- O utilizador acede à sua **lista de ficheiros partilhados**;
- Seleciona o ou os ficheiros que pretende partilhar (deve também ser implementada uma funcionalidade de pesquisa por uma variedade de parâmetros possíveis);
- Após a seleção dos ficheiros pretendidos para partilha, é necessário escolher o ou os contactos com quem se pretende partilhar (uma vez mais, deve haver também uma funcionalidade de pesquisa de contactos);
    - Esta funcionalidade deve ignorar as permissões de partilha, pois é uma partilha intencional por parte do utilizador e portanto deve ignorar as permissões de visualização dos destinatários;
- Seguidamente deve ser enviado um pedido de partilha para cada um dos contactos selecionados. Este pedido deve consistir na escolha para aceitar ou rejeitar a partilha do ficheiro.
    - Se a partilha não for aceite, deve ser enviada uma notificação ao remetente e o processo para aquele contacto termina;
    - Se a partilha for aceite, tal como no processo "**Publicação de ficheiros**", o destinatário deve ter a opção de definir permissões de partilha antes do ficheiro ser adicionado à sua **lista de ficheiros partilhados**;
- Deve então ser colocada a opção de definir permissões para este ficheiro ao destinatário;

- Se o destinatário não pretender colocar permissões, são aplicadas as permissões pré-definidas, que devem ser configuráveis na funcionalidade "**Configuração do sistema"**.
- Se o destinatário pretender definir as permissões, devem ser apresentadas formas para facilitar esta tarefa ao utilizador. Sugere-se a utilização de grupos de utilizadores e funcionalidades de pesquisa.

- Por fim, é finalmente adicionada uma entrada do ficheiro na **lista de ficheiros partilhados** desse destinatário, para que possa ser descarregado e carregado para outros contactos consoante as permissões atribuídas.

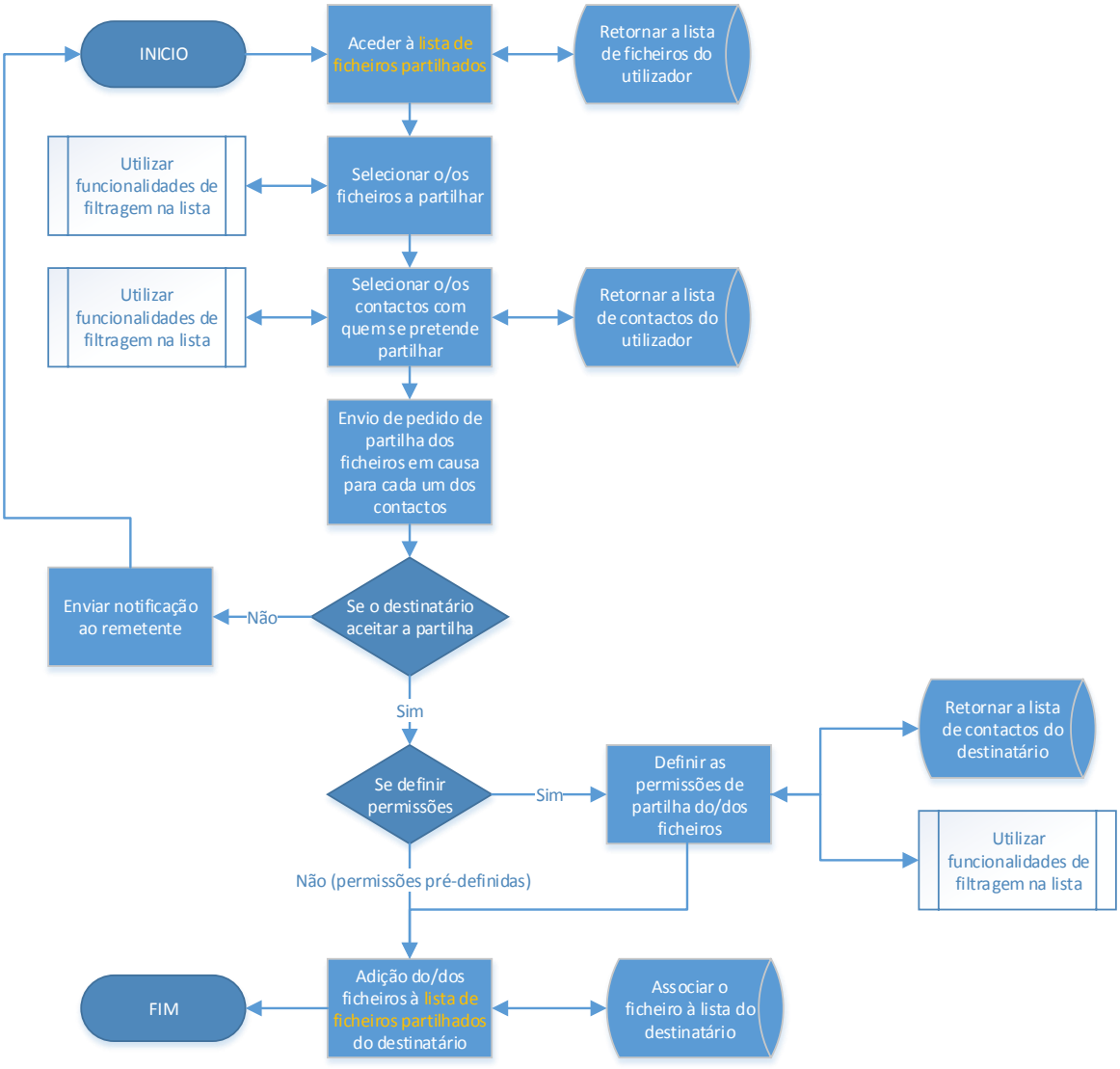

Ilustração 23 - Partilhar ficheiros

3.4 Descarregamento de ficheiros partilhados por contactos

A descarga de ficheiros partilhados por contactos consiste na possibilidade de aceder à **lista de ficheiros partilhados** por esses contactos, escolher o que se pretende, e adiciona-los à **lista de ficheiros partilhados** do utilizador para proceder à descarga.

Seguidamente é descrito este processo com o auxílio à ilustração 24:

- O utilizador acede à **lista de ficheiros partilhados** de uma determinado contacto. Este acesso deve ser proporcionado de múltiplas formas, como sugestões, acesso manual, hiperligações, *etc.;*
- Confrontado com esta lista, o utilizador deve ter a possibilidade de escolher um ou mais ficheiros que pretende obter;
    - Devem ser possibilitadas formas adicionais de procura para facilitar a tarefa;
    - É também importante salientar que deve haver diferenciação entre ficheiros que já estão completamente descarregados e ficheiros ainda por ou a descarregar;
    - Os ficheiros que estão disponíveis na lista dependem das permissões colocadas pelo contacto em cada um deles;
- Após a seleção, deve ser colocada a opção de definir permissões para cada um dos ficheiros selecionados;
    - Se o utilizador não pretender colocar permissões, são aplicadas as permissões pré-definidas, que devem ser configuráveis na funcionalidade "**Configuração do sistema"**.
    - Se o utilizador pretender definir as permissões, devem ser apresentadas formas para facilitar esta tarefa ao utilizador. Sugere-se a utilização de grupos de utilizadores e funcionalidades de pesquisa.
- Por fim, é finalmente adicionada uma entrada de cada um dos ficheiros na **lista de ficheiros partilhados** do utilizador, para que possam ser descarregados e carregados para outros contactos consoante as permissões atribuídas.

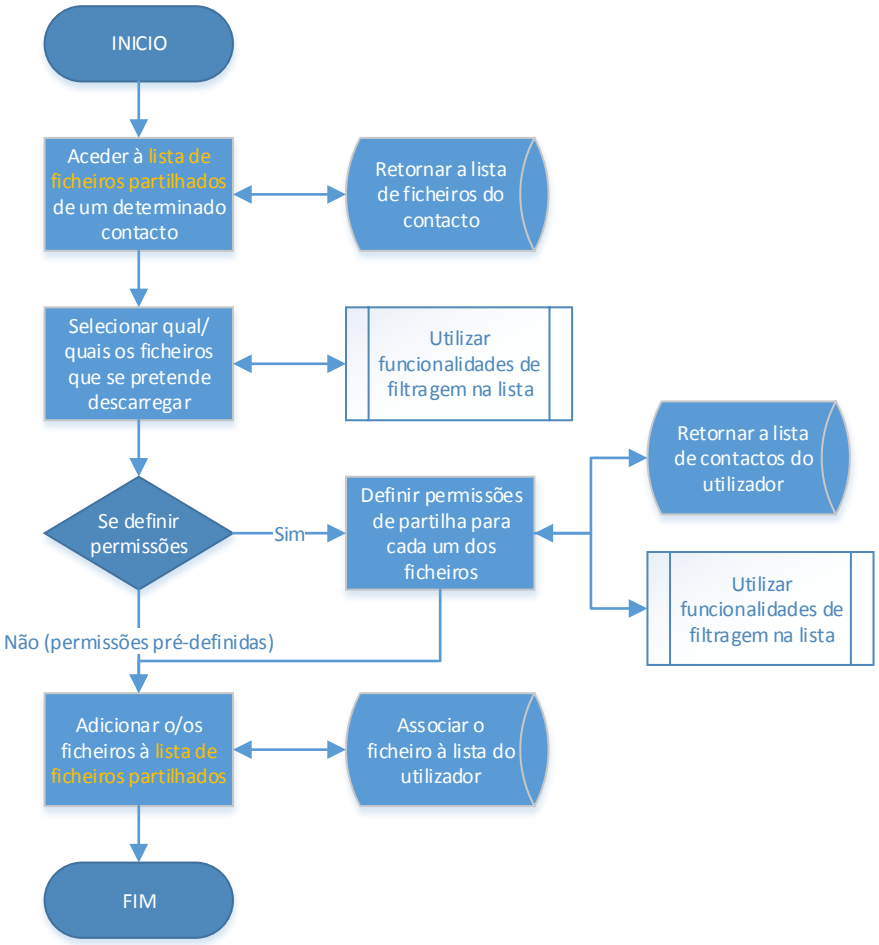

Ilustração 24 – Descarregamento de ficheiros partilhados por contactos

3.5 Classificação de partilhas

A funcionalidade de classificação consiste na possibilidade de um utilizador classificar, segundo um determinado sistema ou escala (um a dez por exemplo), partilhas feitas por outros contactos consoante a experiencia obtida da sua utilização. Esta funcionalidade é importante para a arquitetura, na perspetiva de criar uma forma robusta de confiabilidade em cada partilha. O facto de haver uma autenticação, permite criar a restrição para que cada utilizador possa classificar uma determinada partilha apenas uma vez.

Este sistema também tem a possível vantagem de reduzir a quantidade de partilhas falaciosas ou maliciosas ao longo do tempo, se for utilizado um sistema de classificação apropriado. Seguidamente é descrito o processo de **classificar partilhas** com o auxílio da ilustração 25:

- O utilizador acede à sua **lista de ficheiros partilhados** a partir de uma hiperligação ou botão nomeado com a devida funcionalidade;
- A lista deve ser filtrada com apenas os ficheiros que já foram totalmente descarregados, pois apenas estes podem ser classificados. Não deve ser possível classificar uma partilha se ainda não ouve oportunidade de a avaliar;
- O utilizador escolhe qual o ficheiro ou ficheiros que pretende classificar, tirando proveito de funcionalidades de pesquisa;
- É feita a atribuição da classificação escolhida para cada ficheiro conforme o sistema de classificação utilizado.

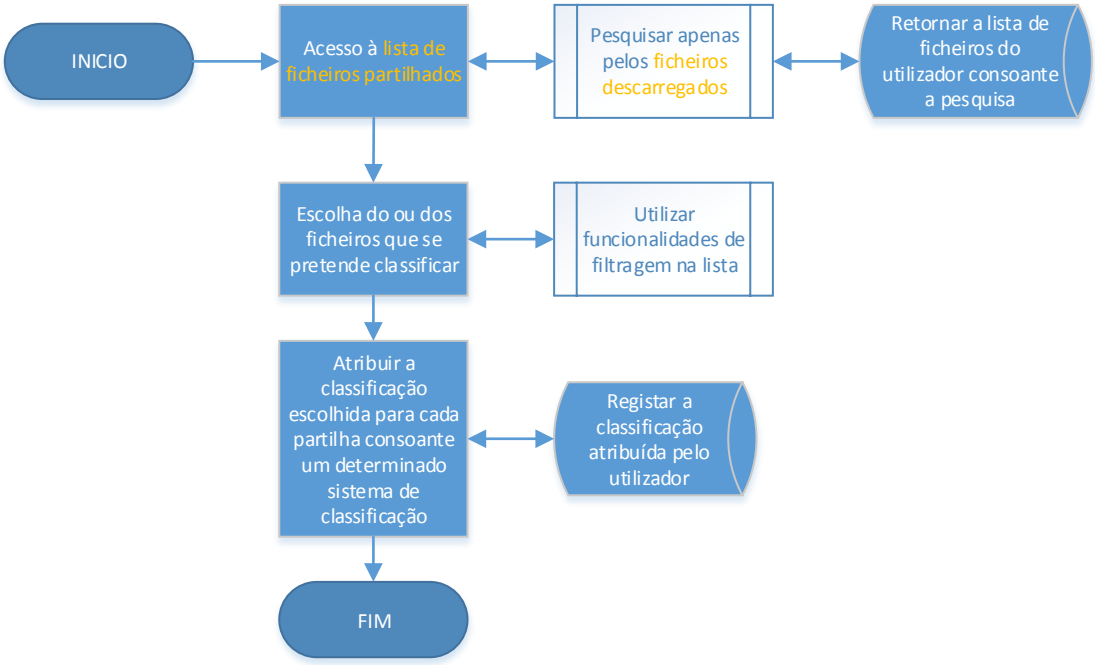

Ilustração 25 – Classificação de ficheiros

3.6 Remoção de ficheiros da lista de ficheiros partilhados

Esta funcionalidade consiste na possibilidade de remover ficheiros da lista de ficheiros partilhados do utilizador. Esta funcionalidade é obviamente importante, principalmente se for colocado um limite de partilhas na lista. No entanto existem alguns cuidados a ter, importantes para o bom funcionamento da arquitetura.

Seguidamente são descritos os passos da remoção com o auxílio à ilustração 26:

- O utilizador acede à **lista de ficheiros partilhados**;
- Escolhe o ou os ficheiros que pretende remover. Devem ser proporcionadas funcionalidade de pesquisa para facilitar a tarefa;
- Depois de escolhidos os ficheiros a remover, deve ser verificado se a remoção do ficheiro da lista torna impossível a sua obtenção por outros utilizadores. Esta verificação pode ser um processo pesado se o sistema não estiver devidamente preparado. Cada ficheiro deve ter uma marcação corretamente atualizada na base de dados, da quantidade de utilizadores que o contêm completamente descarregado. Desta forma é simples fazer esta verificação;
    - Se de facto tornar impossível a obtenção do ficheiro por outros utilizadores, deve ser colocada uma confirmação de remoção ao utilizador com esta informação;
        - Com esta informação, se o utilizador decidir então, não remover o ficheiro, o processo acaba para este ficheiro.
        - Se mesmo com esta informação, o utilizador mantiver a decisão de remover este ficheiro, o processo continua para a próxima fase;
    - Se não tornar impossível a obtenção do ficheiro por outros utilizadores, o processo continua para a próxima fase;
- O ou os ficheiros selecionados, que passaram na verificação, são removidos da **lista de ficheiros partilhados** do utilizador.

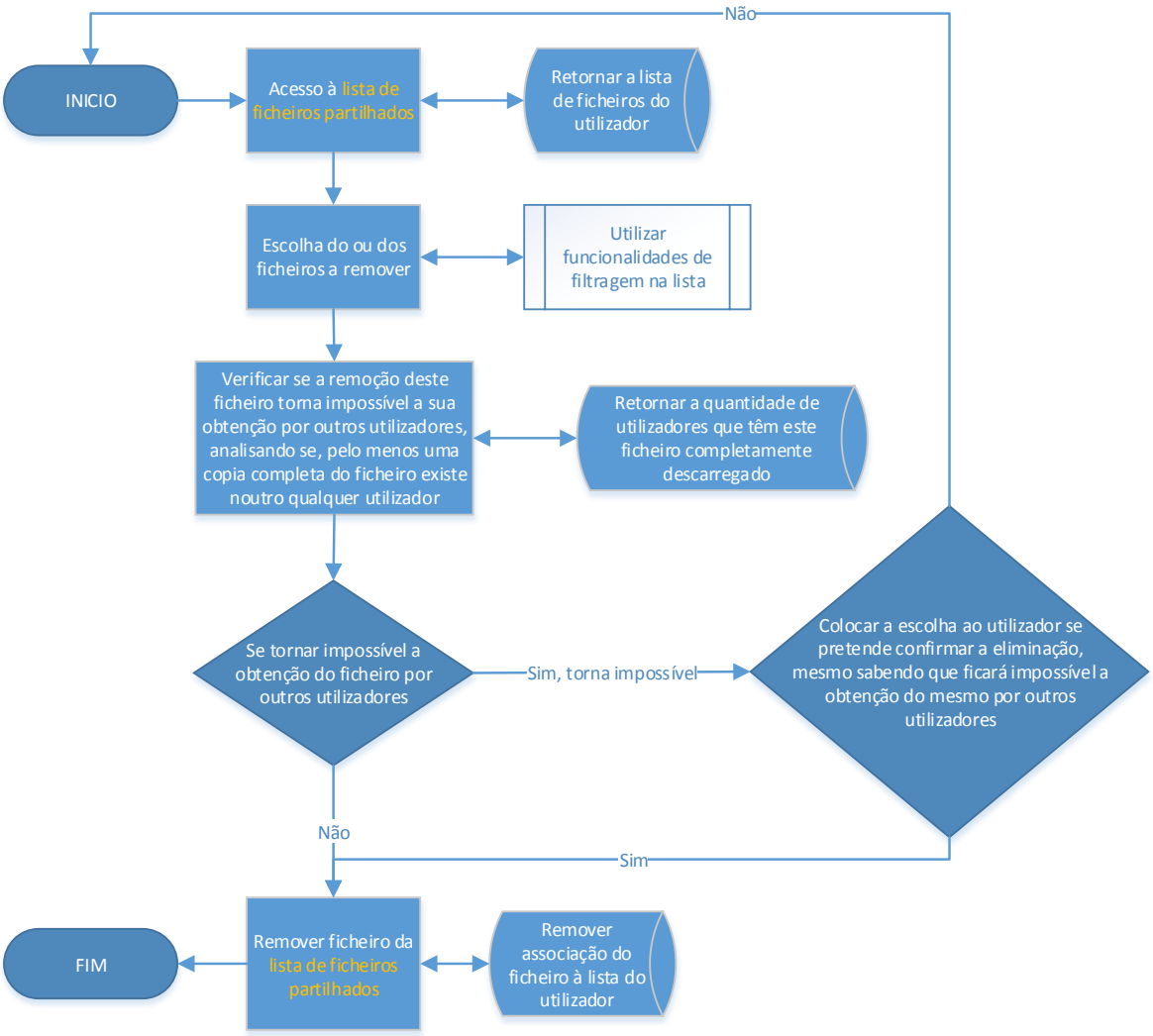

Ilustração 26 – Remoção de ficheiros da lista de ficheiros partilhados

3.7 Remoção dos registos do histórico

Esta funcionalidade consiste na possibilidade de remover o registo de entradas no histórico de ficheiros adicionados à **lista de ficheiros partilhados**. Esta funcionalidade está relacionada com segurança e é importante para os utilizadores que preferem manter a sua confidencialidade.

Seguidamente é descrito este processo com o auxílio à ilustração 27:

- O utilizador acede à **lista de histórico** onde pode visualizar os ficheiros que foram adicionados à sua lista de ficheiros partilhados e quando é que foram adicionados;
- É feita uma seleção dos ficheiros que se pretende remover, utilizando funcionalidades de pesquisa para facilitar a tarefa;
- Os registos são removidos da **lista de histórico** do utilizador.

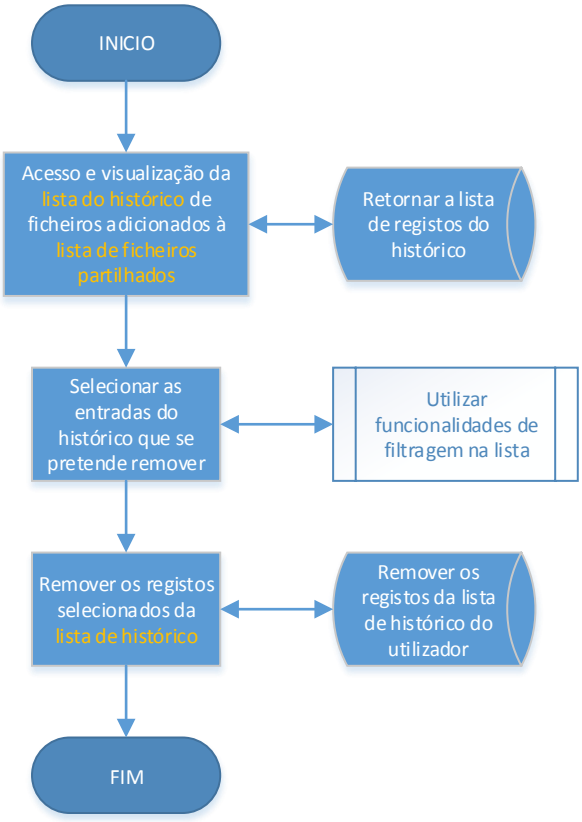

Ilustração 27 – Remoção de registos da lista de histórico

3.8 Configuração do sistema

Esta funcionalidade consiste na possibilidade de alterar as variadas configurações do sistema, referentes à partilha de ficheiros. Esta funcionalidade é importante na medida em que existem múltiplas variáveis que devem ser adaptadas a cada caso (como exemplo,

podemos considerar as diferenças de utilização de banda entre múltiplos utilizadores). Seguidamente é descrito este processo com o auxílio à ilustração 28:

- O utilizador acede às **configurações do sistema de partilha de ficheiros**. Para este efeito, poderá ser utilizada uma página específica;
- São feitas as alterações necessárias ao sistema, consoante as necessidades do utilizador. A alteração das variáveis deve ser opcional, contendo estas um valor pré-definido.
- As alterações feitas são guardadas na base de dados do sistema;
- Os processos correntes devem também ser afetados com as novas configurações.

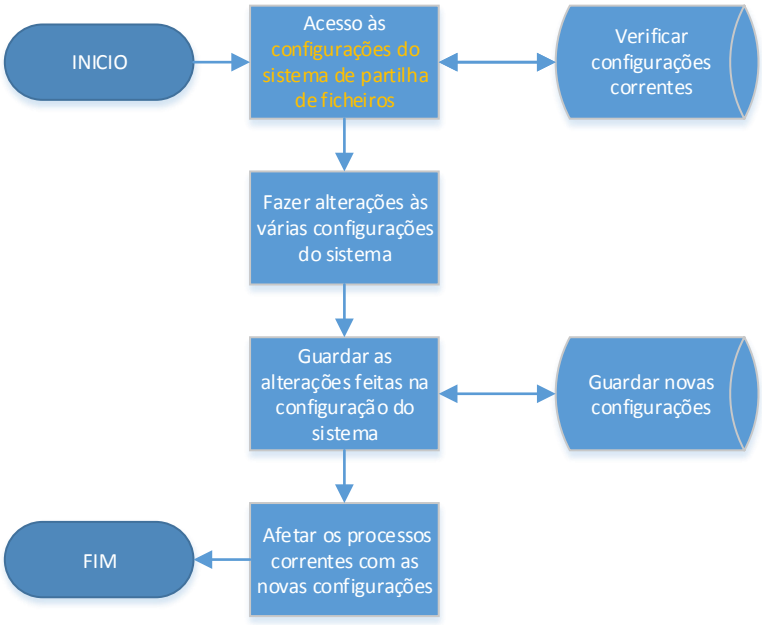

Ilustração 28 - Configuração do sistema de partilha de ficheiros

3.9 Processo de descarga

É importante explicar como é que o processo de descarga de ficheiros BitTorrent se encaixa neste sistema. Para o fazer, é colocada a situação em que um utilizador adicionou um ficheiro que pretende descarregar à sua lista de ficheiros partilhados.

Este ficheiro, irá seguir o comportamento descrito no ponto 2.2 desta dissertação (Partilha de Ficheiros em BitTorrent) para efetuar a descarga. A principal diferença, está na interface de utilizador que fica incorporada no *site*. O processo vai aceder ao servidor de rastreamento para descobrir quais os utilizadores mais próximos que estão correntemente a descarregar ou a carregar o mesmo ficheiro. De seguida é trocada a informação de comunicação com alguns desses utilizadores, dependendo da situação, e o processo pode então prosseguir com a descarga dos pedaços do ficheiro seguindo todos os passos da arquitetura BitTorrent.

Como sugestão, o servidor de rastreamento neste sistema pode ser incorporado com os servidores de contactos, visto que vão conter alguma informação em comum. No entanto é preciso não esquecer que o servidor de rastreamento contem informação volátil, ou seja, os utilizadores que estão a descarregar ou carregar um determinado ficheiro num determinado momento estão constantemente a mudar. É diferente de aqueles que têm o ficheiro na sua **lista de ficheiros partilhados**, pois é uma informação bastante mais estática.

É importante também fazer uma clarificação quanto às permissões de partilha. Existem diferenças entre partilhar um ficheiro da **lista de ficheiros partilhados**, e ser uma semente ou *peer* para descarga de pedaços. Um determinado utilizador pode ter um determinado ficheiro da sua **lista de ficheiros partilhados** classificado como privado, o que significa que não está visível para descarga por outros contactos, mas este pode na mesma ser usado para partilhar pedaços pelo sistema BitTorrent. Esta diferenciação deve estar evidenciada na implementação desta arquitetura. Também é possível bloquear a partilha de pedaços, tornando o acesso a este ficheiro por outros *peers* impossível, mas é importante clarificar esta separação no sistema. Neste contexto, é também possível a situação inversa, ou seja, um determinado ficheiro estar disponível para descarga por outros contactos na **lista de ficheiros partilhados**, mas a descarga no sistema BitTorrent estar bloqueada. O que significa que os outros contactos poderiam usar esta **lista de ficheiros partilhados** para adicionar este ficheiro às suas próprias listas, mas o computador deste utilizador, não seria um dos *peers* ou sementes. Nesta situação, é possível comparar a lista ficheiros partilhados

deste utilizador com os servidores web atuais que servem apenas o propósito de partilhar ficheiros ".torrent".

3.10 Método de implementação

Embora este possa ser considerado um dos maiores desafios, existem vários métodos possíveis para implementar este sistema. Uma sugestão possível é utilizar a Web API BitTorrent, que foi desenvolvida com o propósito de fornecer o acesso às funcionalidades BitTorrent a partir da Web. Esta API encontra-se num processo contínuo de desenvolvimento para fornecer novas funcionalidades e aumentar a sua estabilidade.

4 - CONCLUSÕES

Esta dissertação teve como objetivos propor uma arquitetura de transferência de ficheiros otimizada a redes sociais e provar que esta é capaz de executar a tarefa de forma mais eficaz em relação ao sistema convencional. Foi proposta a utilização de um sistema P2P em relação a um sistema Cliente-Servidor e foram analisados os resultados obtidos a partir de simulações executadas em uma aplicação computacional desenvolvida em linguagem JAVA.

Em face dos resultados obtidos nas simulações, foi possível comprovar que um sistema P2P possibilita um maior proveito da largura de banda no processo de descarga de ficheiros. No entanto, também se verificou que esta estrutura apenas traz benefícios quando a largura de banda dos utilizadores da rede fornece uma capacidade de descarga superior à capacidade de carga. Propicio para a arquitetura proposta, é o facto que nos dias de hoje, a Internet fornecida aos utilizadores é exemplar do primeiro ambiente, ou seja, geralmente a capacidade de descarga é de facto múltiplas vezes superior à capacidade de carga.

A arquitetura proposta nesta dissertação define-se pelo desenvolvimento de funcionalidades extra no *site* de redes sociais, que possibilitam a criação de um sistema de partilha de ficheiros P2P utilizando o protocolo BitTorrent. A arquitetura evidenciada está ausente de especificação das tecnologias a usar para atingir o seu propósito, deixando esse aspeto à escolha de quem proceder ao seu desenvolvimento.

Conforme as conclusões acima tiradas, determina-se que a arquitetura proposta proporciona a possibilidade de fornecer um sistema mais eficiente de partilha de ficheiros para os *sites* de redes sociais. Os utilizadores que já usavam o sistema BitTorrent vão poder continuar a usar como sempre o fizeram, com compatibilidade total com aqueles que vão usar o sistema no *site* de rede social. Aqueles que decidirem usar o sistema com a arquitetura proposta, vão ter a vantagem de tirar proveito das funcionalidades sociais adicionais. Os utilizadores que forem confrontados com o sistema BitTorrent pela primeira vez no *site* de rede social, terão acesso a

uma forma de partilhar ficheiros intuitiva e mais eficiente, em comparação ao sistema convencional, utilizando a velocidade de transferência como fator decisivo.

5 - REFERÊNCIAS

[Marques da Silva 2012] M. Marques da Silva - *Multimedia Communications and Networking*, CRC Press, 1st edition, ISBN: 9781439874844, Boca Raton, USA, March 2012.

[Kozierok 2005] C. M. Kozierok - *The TCP/IP Guide*, Version 3.0, September 2005.

[Cohen 2003] B. Cohen - *Incentives Build Robustness in BitTorrent*, bitconjurer.org/BitTorrent, May 2003.

[Melville et al. 2002] L. Melville, J. Walkerdine, I. Sommerville - *Dependability Properties of P2P Architectures*, in Proceedings of the Third International Conference on Peer-to-Peer Computing, Linköping Sweden, 2002.

[Acosta and Chandra 2005] W. Acosta, S. Chandra - *Unstructured Peer-to-Peer Networks - Next Generation of Performance and Reliability*, IEEE INFOCOM, 2005.

[Pouwelse et al. 2004] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips - *A Measurement Study of the BitTorrent Peer-to-Peer File-Sharing System*, Parallel and Distributed Systems group, Delft University of Technology, The Netherlands, 2004.

[Ellison 2007] N. B. Ellison - *Social Network Sites: Definition, History and Scholarship*, Journal of Computer-Mediated Communication, 2007.

[Ellison et al. 2007] N.B. Ellison, C. Steinfield, C. Lampe - *The Benefits of Facebook ''Friends'' Social Capital and College Students' Use of Online Social Network Sites*, Journal of Computer-Mediated Communication, 2007.

[Postel 1977] J. Postel - *Internet Engineering Note Number 2*. IEN 2, 1977.

[Milojicic et al. 2002] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, Z. Xu - *Peer-to-Peer Computing*. HP Laboratories Palo Alto, July 2002.

[BitTorrent 2012] *BitTorrent and µTorrent Software Surpass 150 Million User Milestone*. http://www.bittorrent.com/intl/es/company/about/ces_2012_150m_users. 2012-01-09. Acedido em: 26-03-2013

[InternetWorldStats 2012] *Facebook Users in the World by Regions*. http://www.internetworldstats.com/facebook.htm. 2012-09-30. Acedido em: 26-03-2013

[Tweetsmarter 2012] D. Larson - *Infographic: Spring 2012 Social Media User Statistics*. http://blog.tweetsmarter.com/social-media/spring-2012-social-media-user-statistics/. 2012-05-15. Acedido em: 26-03-2013

6 - ANEXO A

Neste anexo são demonstrados os excertos de código mais relevantes da aplicação de simulação.

6.1 Código simulação Cliente-Servidor

```
public class FileTransferSimulator_ClientServer implements Runnable {

    XYSeries serie;
    long fileSize_bit;
    PingTester24View view;

    public FileTransferSimulator_ClientServer(XYSeries serie, long fileSize_bit,
PingTester24View view) {
        this.serie = serie;
        this.fileSize_bit = fileSize_bit;
        this.view = view;
    }

    public void run() {
        try {
            GregorianCalendar calendar = new GregorianCalendar(2013, 0, 1, 0, 0, 0);
            //Total transferido em bit
            long transfered_bit = 0;
            //Upload total em bit
            long uploadTotal = 1048576;//1048576;
            //Tempo decorrido em ms
            long totalTimePassed_millis = 0;
            while (fileSize_bit > transfered_bit) {
```

```
                transfered_bit += uploadTotal;
                Thread.sleep(100);
                calendar.setTimeInMillis(calendar.getTimeInMillis() + 100);
                //Adicionar entrada no Gráfico
                serie.add(calendar.getTimeInMillis(), transfered_bit / 1024 / 1024);
                totalTimePassed_millis += 100;
                //Demonstrar o estado do processo
                view.setStatusMessage("Tempo decorrido: " + totalTimePassed_millis / 1000 + "
segundos | Capacidade de descarga total corrente do utilizador 2: " + ((uploadTotal * 10) /
1024 / 1024) + " Mbit/s | Total descarregado: " + (transfered_bit / 1024 / 1024) + " Mbit");
            }
        } catch (Exception ex) {
            System.err.println(ex.toString());
        }
    }
}
```

6.2 Código simulação Peer-to-peer

```
public class FileTransferSimulator_Peer2Peer implements Runnable {

    XYSeries serie;
    long fileSize_bit;
    PingTester24View view;

    public FileTransferSimulator_Peer2Peer(XYSeries serie, long fileSize_bit,
PingTester24View view) {
        this.serie = serie;
```

```
        this.fileSize_bit = fileSize_bit;
        this.view = view;
    }

    public void run() {
        try {
            GregorianCalendar calendar = new GregorianCalendar(2013, 0, 1, 0, 0, 0);
            //Total transferido em bit
            long transfered_bit = 0;
            //Intrevalo de tempo em ms entre cada adição de mais um peer
            long fiveSec = 2000;
            //Upload de cada peer em bit
            long uploadIndividual = 1048576;//5242880;//1048576;
            //Upload total em bit
            long uploadTotal = 9437184;//5242880;//1048576;
            //Download total em bit
            long downloadTotal = 9437184;//5242880;//9437184;
            //Tempo decorrido em ms
            long totalTimePassed_millis = 0;
            uploadTotal = uploadTotal > downloadTotal ? downloadTotal : uploadTotal;
            while (fileSize_bit > transfered_bit) {
                transfered_bit = (transfered_bit + uploadTotal > fileSize_bit ? fileSize_bit :
(transfered_bit + uploadTotal));
                Thread.sleep(100);
                calendar.setTimeInMillis(calendar.getTimeInMillis() + 100);
                //Adicionar entrada no Gráfico
                serie.add(calendar.getTimeInMillis(), transfered_bit / 1024 / 1024);
                totalTimePassed_millis += 100;
```

```
                fiveSec -= 100;
                if (fiveSec == 0) {
                    if ((uploadTotal += uploadIndividual) > downloadTotal) {
                        uploadTotal = downloadTotal;
                    }
                    fiveSec = 2000;
                }
                //Demonstrar o estado do processo
                view.setStatusMessage("Tempo decorrido: " + totalTimePassed_millis / 1000 + "
segundos | Capacidade de descarga total corrente do utilizador 2: " + ((uploadTotal * 10) /
1024 / 1024) + " Mbit/s | Total descarregado: " + (transfered_bit / 1024 / 1024) + " Mbit");
            }
        } catch (Exception ex) {
            System.err.println(ex.toString());
        }
    }
}
```