Curiata: Uma arquitectura P2P auto-organizável para uma localização flexível e eficiente de recursos

Curiata: Uma arquitectura P2P auto-organizável para uma localização flexível e eficiente de recursos

João Alverinho, João Leitão, João Paiva, and Luis Rodrigues *

{jalveirinho,jleitao,jgpaiva,ler}@gsd.inesc-id.pt, INESC-ID/IST

Resumo As arquitecturas *entre pares* têm vindo a emergir como uma solução viável para suportar serviços de localização de recursos em sistemas distribuídos de larga escala. A maioria das soluções baseia-se em redes estruturadas (*DHT*s) ou não-estruturadas. As DHTs são mais eficientes para procuras exactas, enquanto que as soluções não-estruturadas apesar de menos eficientes são mais flexíveis. Neste artigo propomos uma nova solução auto-organizável que combina as abordagens estruturada e não-estruturada. Resultados experimentais extraídos através de simulação mostram que a nossa solução consegue oferecer uma boa precisão nas respostas às interrogações, com reduzido custo de mensagens e baixa latência.

Abstract Peer-to-Peer arquitectures have emerged as a viable solution to support resource location services in large-scale distributed systems. Most solutions are based on either structured (*DHT*s) or unstructured overlay networks. DHTs excel on *exact-match* queries, whilst unstructured solutions despite being less efficient are more flexible. In this paper we propose a novel self-organizing solution that combines both structured and unstructured approaches. Experimental results through simulation show that our solution is able to offer good precision in query responses, while keeping a low message cost as well as a low latency.

1 Introdução

Desde o aparecimento do Napster[4] em 1999, os sistemas entre-pares (P2P) têm sido alvo de desenvolvimento e investigação, tanto na academia como na indústria. As aplicações deste tipo de tecnologia incluem partilha de ficheiros[3,1], distribuição de conteúdos[2], partilha de processamento[11], voz sobre IP[5], entre outras. Um serviço fundamental em qualquer sistema P2P, é a localização de recursos. Dado que os sistemas P2P tipicamente almejam suportar um número extremamente elevado de participantes em que cada um destes pode partilhar múltiplos recursos (tempo de CPU, espaço em disco, ficheiros,etc), o espaço de procura pode ser enorme. Soluções centralizadas têm-se provado inviáveis devido a limitações na sua capacidade de escala e fiabilidade. Por outro lado, uma

* Este trabalho foi parcialmente suportado pelo financiamento pluri-anual do INESC-ID através do programa PIDDAC e pelos projectos "Redico" (PTDC/EIA/71752/2006) e "HPCI" (PTDC/EIA-EIA/102212/2008).

procura exaustiva em todos os participantes também não é viável. Assim sendo, o desenho e concretização de serviços distribuídos de localização de recursos é de enorme relevância.

Existem dois tipos principais de sistemas entre-pares: estruturados e não estruturados. Tipicamente, os sistemas P2P estruturados concretizam uma tabela de dispersão distribuída (DHT). Estes sistemas suportam procuras exactas de forma muito eficiente. Contudo, os sistemas estruturados fornecem suporte reduzido à execução de interrogações complexas e/ou inexactas. Adicionalmente, os sistemas estruturados podem ter uma manutenção dispendiosa em ambientes altamente dinâmicos. Uma alternativa passa por usar sistemas não-estruturados, que têm custos de manutenção reduzidos e suportam a execução de qualquer tipo de interrogações sem custos adicionais. No entanto, estes sistemas não conseguem resolver interrogações de forma eficiente. De facto, nos sistemas não-estruturados a localização de recursos é tipicamente realizada através de soluções de procura não guiada (cega), que não produzem resultados satisfatórios no caso geral. Estas abordagens usam mecanismos baseados em inundação (do Inglês, *flooding*), uma solução extremamente dispendiosa e ineficiente; ou em encaminhamento aleatório de uma única mensagem (conhecido como *Random Walk*), uma estratégia com elevada latência e que pode exibir uma fraca Recolha[1] (do Inglês, *Recall*) nos resultados da procura de recursos que não sejam extremamente comuns.

Este artigo apresenta o *Curiata*, um sistema de localização de recursos escalável e eficiente que combina os benefícios das abordagens estruturadas e não-estruturadas. A nossa solução permite flexibilidade nas interrogações, como na maioria das abordagens não-estruturadas, mantendo a rapidez e eficiência providenciada pelas soluções estruturadas (baseadas em DHT's). O modo de operação do *Curiata* inspira-se na organização das sociedades humanas. Durante as duas primeiras décadas da república romana, a população organizava-se em unidades chamadas *curia* de natureza étnica; as *curia* reuniam-se numa assembleia, a *comitia curiata*, para fins legislativos, eleitorais e judiciais, onde os cônsules tinham uma papel especial. De modo análogo, na nossa solução, os participantes organizam-se autonomamente numa rede sobreposta não estruturada onde os nós com recursos semelhantes estabelecem relações de vizinhança através de um processo de baixo custo executado em segundo plano (a *curia*). Para além disto, os nós de cada *curia* elegem representantes para se juntarem a uma rede sobreposta estruturada (a *curiata*). Os membros da rede estruturada são utilizados como ponto de contacto para outros nós com conteúdos semelhantes. Assim sendo, a camada estruturada é utilizada para encaminhar eficientemente as interrogações para regiões da camada não-estruturada que contenham nós que partilhem o tipo de recursos que são procurados. Após ser encaminhada na rede estruturada, a interrogação é propagada pelos membros da *curia* utilizando técnicas de redes não estruturadas como inundação limitada ou encaminhamento aleatório. Além de fornecer uma infra-estrutura que permite a execução eficiente e flexível de interrogações, o *Curiata* também pretende atingir uma *recolha* elevada, com um reduzido custo de mensagens, independentemente da raridade dos recursos.

[1] Esta métrica é defenida em detalhe na secção 4.

Este artigo está organizado da seguinte forma. A Secção 2 fornece uma panorâmica geral dos trabalho relacionado. A Secção 3 descreve o nosso sistema em maior pormenor. Na Secção 4 apresenta-se os resultados da avaliação, enquanto que na Secção 5 registam-se alguns comentários finais que concluem o artigo.

2 Trabalho Relacionado

Os algoritmos de localização de recursos para sistemas P2P têm sido estudados intensivamente existindo diversas soluções propostas na literatura. A abordagem mais simples consiste em adoptar um esquema centralizado, onde um único nó é responsável por manter informação acerca da localização de todos os recursos disponíveis no sistema[4,15]. Dado que um índice central mantém conhecimento global dos recursos disponíveis em todo o sistema, este pode facilmente processar interrogações complexas. Contudo, o custo de processamento imposto a um único nó, para manter informação actualizada sobre todos os recursos do sistema e para processar todas as interrogações geradas por cada participante, pode ser demasiado elevado num ambiente dinâmico de larga escala.

Os sistemas P2P estruturados, que concretizam tabelas de dispersão distribuídas, suportam procuras exactas com um custo em número de mensagens logarítmico com o tamanho do sistema[16,19,18]. No entanto, as DHTs fornecem pouco suporte para interrogações inexactas, dado que decompor uma interrogação complexa em várias interrogações exactas não é trivial e pode até ser impossível. A maioria das soluções existentes (por exemplo [6] e [17]) apesar de permitirem pesquisas mais complexas, apresentam ainda assim uma flexibilidade reduzida ou custos de sinalização e comunicação maiores que os do *Curiata*

Dadas as limitações das redes sobrepostas estruturadas no suporte a interrogações inexactas, torna-se atractivo utilizar redes não-estruturadas, dada a sua maior flexibilidade e menor custo de manutenção. O recurso à inundação com raio limitado é a técnica mais imediata para realizar a localização de recursos sobre redes não-estruturadas[20]. Contudo, esta técnica é muito dispendiosa (devido à duplicação de mensagens) e pode ser ineficiente na localização de recursos raros. As interrogações podem também ser disseminadas recorrendo a percursos aleatórios[14,9], ou encaminhamento informado[8], técnicas menos dispendiosas mas com maior latência e menor recolha. A eficiência das interrogações pode ser melhorada utilizando técnicas como o enviesamento da rede sobreposta para que esta se aproxime de uma rede pequeno-mundo (do inglês, *small-world*), replicando todos os índices de recursos na vizinhança directa de cada nó, e encaminhando as interrogações para nós com maior grau. O GIA[7] é um exemplo conhecido de um sistema que combina estas técnicas. Estas soluções, para além de obrigarem os nós a manter estado adicional, degeneram em configurações onde as interrogações são apenas processadas por uma pequena fracção dos participantes. Adicionalmente, estas soluções não são desenhadas para lidar convenientemente com interrogações que visem recursos raros.

Alguns sistemas propõem a utilização de super-nós[22], onde os participantes se organizam numa hierarquia com dois níveis. Os nós no nível superior mantêm índices consolidados dos recursos partilhados pelos participantes do nível inferior que se ligam a si. Nestes sistemas os super-nós processam a maioria das interrogações e os custos de manutenção dos índices podem facilmente tornar-se

proibitivos num ambiente dinâmico. Apesar de também usarmos uma topologia hierárquica com dois níveis, no *Curiata*, todos os participantes contribuem activamente para a disseminação e processamento das interrogações. Adicionalmente, os participantes não necessitam de manter índices relativos aos seus vizinhos, sendo que, apenas informação genérica sobre as categorias dos recursos dos vizinhos é necessária de forma a enviesar a topologia.

3 *Curiata*

A arquitectura do *Curiata* combina uma camada não-estruturada (a *curia*) com uma camada estruturada (a *comitia curiata*). A nossa solução apresenta cinco componentes principais: i) O *índice de recursos*, que descreve os recursos disponíveis localmente no participante. ii) A camada de *rede não-estruturada enviesada*, que utiliza um protocolo distribuído e auto-organizável para garantir que os participantes estabelecem relações de vizinhança com outros participantes cujos recursos sejam similares. iii) A camada da *rede estruturada*, que é activada somente quando um participante é eleito como cônsul. iv) O *módulo de eleição de cônsules*, que utiliza um protocolo colaborativo para seleccionar os participantes que fazem parte da rede estruturada. v) O *módulo de encaminhamento das procuras*.

Índice de Recursos No *Curiata*, assumimos que os recursos podem ser classificados num conjunto de categorias. O índice de recursos mantém um registo local de todas as categorias dos recursos do participante assim como o número de recursos disponível para cada uma dessas categorias. Esta informação é utilizada para identificar participantes que possuem recursos similares. O esquema de classificação utilizado é ortogonal ao nosso sistema. Por exemplo, uma biblioteca distribuída de artigos sobre informática poderia utilizar o *ACMComputing Classification System* para classificar o conteúdo. Um repositório de música poderia extrair as categorias necessárias para classificar o conteúdo das etiquetas mais utilizadas em aplicações populares como o "Last.fm" (`http://last.fm`).

Adicionalmente, o índice de recursos mantém também, para cada categoria c dos recursos de um participante, a fracção de recursos desse participante que se enquadram nessa categoria. Este valor é denominado como $frac_c$. As categorias são ordenadas de acordo com os valores de *frac*. As primeiras t categorias são utilizadas para definir as relações de vizinhança estabelecidas pelo participante na rede não-estruturada.

Camada de Rede Não-Estruturada O propósito desta camada é organizar todos os participantes que têm recursos disponíveis numa rede sobreposta não-estruturada enviesada. Mais especificamente, propomos que cada participante execute um algoritmo distribuído auto-organizável, para adaptar a topologia da rede sobreposta de acordo com os recursos disponíveis em cada nó. O nosso algoritmo consiste numa versão especializada do X-BOT [12], adaptada para ir de encontro a alguns dos requisitos da nossa arquitectura. O X-BOT é um protocolo distribuído que enviesa a topologia de uma rede não-estruturada simétrica (com as características descritas em [13]) dada uma função de proximidade que forneça uma medida de "distância" entre dois nós no sistema. No caso particular do *Curiata*, a função de proximidade reflecte a similaridade entre os recursos que

dois nós disponibilizam. O objectivo desta estratégia é o de conseguir processar interrogações eficientemente, limitando a procura à área da rede onde existe maior probabilidade de existirem os recursos.

Mais precisamente, a camada da *curia* opera do seguinte modo. Cada nó mantém dois conjuntos de vizinhos, designados por *vista activa* e *vista passiva*. O conjunto de vizinhos da vista activa define a rede que é utilizada para propagar as interrogações. Assim, o tamanho da vista activa d define o grau do nó na *curia*. Os vizinhos da vista passiva são utilizados para explorar a rede e encontrar outros participantes com recursos similares. A vista passiva é actualizada periodicamente por um processo de actualização aleatória das vistas de baixo custo [13]. A vista activa é actualizada através do processo de coordenação introduzido pelo X-BOT utilizando a estratégia que se descreve a seguir.

A *curia* divide a vista activa em t segmentos; cada um destes segmentos dedica-se a uma das primeiras t categorias. O segmento para a categoria c tem uma dimensão $segmento_c$ no intervalo $\lceil smin, d \cdot \text{frac}_c \rceil$. Onde $smin$ é o tamanho mínimo de um segmento, e é definido como $\frac{d}{2t}$. Por exemplo considere-se um sistema onde a *curia* está configurada para seleccionar vizinhos de acordo com as primeiras 5 categorias ($t = 5$). Considere-se um participante p tal que as primeiras 5 categorias $(c1, \ldots, c5)$ têm as seguintes fracções associadas: $(0.5, 0.2, 0.1, 0.1, 0.1)$. Se o participante p apresentar um grau $d = 20$, a sua vista activa seria dividida da seguinte maneira: $(10, 4, 2, 2, 2)$. Considerando este particionamento da vista activa, um nó utiliza o seguinte algoritmo para enviesar os seus vizinhos:

1. O primeiro objectivo do participante quando se junta à rede não estruturada é preencher a sua vista activa, independentemente da similaridade dos seus potenciais vizinhos. Assim sendo, enquanto o número de vizinhos for menor que d, o nó preenche a sua vista activa sem levar em consideração os segmentos definidos para cada categoria.
2. Após ter a vista activa cheia, a próxima prioridade para o participante é ter vizinhos que pertençam às suas t categorias.
3. De seguida, o participante tenta substituir os seus vizinhos por novos vizinhos de modo a que cada segmento da vista activa seja preenchida por participantes com recursos que correspondam à categoria desse segmento.
4. Finalmente, assim que este último critério seja atingido, o participante deixa de executar o algoritmo de auto-organização e mantém os seus vizinhos inalterados enquanto estes não abandonarem a rede.

A selecção do protocolo X-BOT é justificada pela capacidade que este protocolo exibe de enviesar a topologia sem no entanto permitir que a rede apresente um coeficiente de aglomeração excessivo, nomeadamente este protocolo opera de forma a proteger a conectividade global da rede não estruturada (a rede não se particiona em "sub-redes" desconexas entre si). Adicionalmente a operação do protocolo X-BOT garante que o número de vizinhos dos nós se mantém constante.

Camada Estruturada O objectivo da camada estruturada consiste em encaminhar as procuras para zonas da *curia* onde exista maior probabilidade de se

encontrarem os recursos desejados. Para tal, uma fracção dos nós que pertencem à rede não-estruturada também se juntam a uma DHT (como o Chord [19] ou o Pastry [18]). Estes nós são eleitos de forma a representarem uma categoria associada aos recursos dos nós numa dada região da rede não-estruturada, e são designados por *cônsules regionais*. A camada estruturada (a DHT), opera como uma assembleia de representantes de cada uma das diferentes regiões no espaço não-estruturado.

Eleição de Cônsules Em cada região de raio r na rede não-estruturada, se existe uma categoria c que é a principal categoria de um nó (ou seja, a categoria com a maior fracção na vista activa desse participante) nessa região, existe um participante que representa c na DHT. Esse participante designa-se por *cônsul regional* para a categoria c, ou simplesmente c-cônsul. Um c-cônsul junta-se à DHT com um identificador construído através da concatenação dos bits mais significativos de *hash(c)* com os bits menos significativos de *hash(node_id)*. Isto garante que múltiplos contactos para a mesma categoria, em diferentes regiões, têm identificadores diferentes mas, posicionam-se numa região consecutiva no espaço de endereços da DHT.

Um c-cônsul usa a rede não-estruturada para periodicamente enviar um *sinal* para os nós na sua vizinhança de raio r. Nós que recebem este sinal abstêm-se de competir para se tornarem c-cônsules. Se um nó que possui a categoria c como a sua principal categoria não receber nenhum sinal durante um determinado intervalo de tempo, este decide competir com outros potenciais candidatos para se tornar um contacto regional para c.

Para tal, esse nó envia para os nós na sua vizinhança de raio r um *sinal* de *promoção* a cônsul. Passado um intervalo de tempo t pré-definido, caso o nó não tenha recebido nenhum *sinal* de *promoção* proveniente de outro nó, ele considera que a sua *promoção* teve sucesso e tenta juntar-se à camada estruturada. Quando múltiplos nós competem, um protocolo de eleição (tipo *bully*) é utilizado para seleccionar que nó se torna c-cônsul. Por exemplo, dando prioridade ao nó que possua mais recursos da categoria a que se candidata.

Um nó que é eleito para ser um c-cônsul, pode utilizar um outro cônsul de uma categoria c' como ponto de contacto para se juntar à DHT ou, se não conhecer nenhum, realiza um percurso aleatório na rede não-estruturada para encontrar um nó que tenha o contacto de um cônsul.

Procuras Seguidamente descreve-se a forma como são executadas procuras no nosso sistema. A nossa arquitectura não restringe o formato nem a linguagem utilizada nas interrogações. Existe apenas um requisito: a partir da interrogação deve ser possível extrair o conjunto $\mathcal{Q}$ de categorias relevantes aos recursos alvo da interrogação se direcciona. Por exemplo, assuma-se que a interrogação procura por uma música por Aldina Duarte; assim sendo, terá que ser possível extrair categorias como *Música*, *Fado* e *Portugal*.

Uma procura é executada do seguinte modo: i) Primeiro, a interrogação é encaminhada para um membro da DHT; ii) Seguidamente uma cópia da interrogação é encaminhada para cada categoria $c \in \mathcal{Q}$ utilizando a DHT. Cada cópia será recebida por um c-cônsul para essa categoria. De modo a promover balanceamento de carga, para cada categoria c, a interrogação é encaminhada para um

identificador composto por *hash(c)*||{s bits aleatórios}. Isto garante que diferentes interrogações são injectadas na camada não-estruturada através de diferentes representantes dessa categoria; iii) Cada c-cônsul inicia um percurso aleatório de comprimento $\frac{k}{|\mathcal{Q}|}$ na sua vizinhança. Estes percursos são guiados, sendo a interrogação apenas encaminhada para vizinhos que possuam recursos da categoria c; iv) Cada nó visitado pela interrogação executa-a e, caso possua os recursos procurados, adiciona o seu identificador à interrogação; v) Finalmente, quando a interrogação efectua um número máximo de saltos na rede, é retornada à fonte a lista de todos os nós encontrados que satisfazem a interrogação.

O objectivo é processar cada interrogação com um custo de mensagem aproximado k. No protótipo actual, o valor de k é estático. Contudo, k poderia ser ajustado dinamicamente consoante uma estimativa da raridade do recurso a ser procurado (por exemplo, baseado nos resultados retornados por procuras anteriores). Para recursos mais comuns um valor de k mais limitado poderá ser suficiente para a localização dos mesmos, enquanto que para recursos raros pode ser útil aumentar o número de saltos que o percurso aleatório pode efectuar na rede.

O custo total de mensagens de uma interrogação é a soma de $\frac{k}{|\mathcal{Q}|} \times |\mathcal{Q}|$, com o custo de chegar a um membro da DHT a partir da fonte (tipicamente 1 salto), e o número de saltos na DHT necessários para chegar ao c-cônsul respectivo (na ordem de $c \cdot \ln T$ onde T é o número de nós na DHT).

É possível configurar o sistema de modo a promover um valor baixo de T, ajustando os valores de d (número de vizinhos de cada nó) e de r (raio do sinal enviado pelos cônsules). Como trabalho futuro, pretendemos estudar formas de ajustar os valores destes parâmetros em tempo de execução, por exemplo de forma a diminuir o valor de r no caso de existir um pico ao número de interrogações efectuadas, promovendo um aumento do número de cônsules e assim, um melhor balanceamento de carga ao nível dos nós na DHT.

4 Avaliação Experimental

Nesta secção apresentam-se os resultados obtidos com uma concretização do nosso sistema para o simulador *Peersim*[10]. Para as experiências foi utilizada uma rede com 10.000 nós e 11 categorias. A cada nó no sistema é atribuída uma categoria das onze. Os recursos na rede estão também eles associados a uma única categoria e possuem um identificador único. Os recursos de cada categoria c são alocados aleatoriamente em nós nessas categorias, existindo 5 nós distintos na rede que tenham esse recurso. Configuraram-se os nós para terem $d = 20$. A Tabela 1 sumariza o número de nós em cada categoria e o número total de recursos únicos associados a cada categoria. O raio das regiões para eleição de cônsules foi configurado com um valor de 2 e, neste cenário aproximadamente 200 participantes são eleitos como cônsules e juntam-se à camada estruturada(DHT). Consideramos ainda que cada categoria se insere numa das 3 classes de popularidade que se seguem: *Categorias Raras:* categorias com menos de 100 nós (Categorias G a K); *Categorias Intermédias:* categorias com 100 a 1.000 nós (Categorias D a F); *Categorias Comuns:* categorias com mais de 1.000 nós (Categorias A a C).

Nestas experiências foram realizadas 10.000 procuras, com origem em nós aleatórios na rede, que foram definidas de forma a possuírem um único recurso alvo. Mais à frente serão apresentados os resultados para categorias comuns e raras.

Arquitecturas Avaliadas Comparamos o desempenho de quatro arquitecturas diferentes:

Percursos Aleatórios numa Topologia Aleatória (PATA): Corresponde a um sistema que utiliza apenas uma rede não-estruturada aleatória, onde não se aplica um processo de auto-organização. Nesta arquitectura, não existe DHT e os vizinhos não dependem da similaridade dos seus recursos.

Percursos Aleatórios Guiados numa Topologia Aleatória (PAGTA): Corresponde a um sistema que usa uma rede não-estruturada aleatória, onde não existe enviesamento da topologia. Contudo, nesta arquitectura as procuras são guiadas utilizando um mecanismo similar ao utilizado no *Curiata*.

Percursos Aleatórios Guiados numa Topologia Enviesada (PAGTE): Corresponde a um sistema que utiliza uma rede não-estruturada enviesada pelo mesmo processo de auto-organização utilizado no *Curiata*. Nesta arquitectura as procuras também são guiadas. Contudo não existe DHT para encaminhar as interrogações.

Curiata: Concretização completa da arquitectura descrita neste artigo.

Interrogações Nas simulações, cada interrogação é concretizada recorrendo a um único percurso aleatório guiado com comprimento $k = 128$ ou $k = 256$. Este comprimento é medido a partir do nó que produz a interrogação. Assim, todos os saltos na rede são considerados, incluindo os saltos necessários para atingir o cônsul mais perto na DHT para a categoria referente a cada interrogação.

Para simplificar a análise dos resultados, cada interrogação nas simulações procura apenas por um único recurso. Dado que cada recurso está associado a apenas uma categoria, esta categoria é utilizada para guiar a interrogação tanto na arquitectura PAGTA, como na arquitectura PAGTE e também no *Curiata*. Note-se que o facto de os recursos possuírem identificadores únicos é apenas um artefacto de simulação. Como referido anteriormente, o *Curiata* permite a utilização de interrogações arbitrariamente complexas. Assim, a procura por um recurso específico simula uma qualquer interrogação complexa que é satisfeita apenas pelo recurso com esse identificador.

Métricas Nesta avaliação experimental consideramos as seguintes métricas: i) *Taxa de Sucesso:* Percentagem de interrogações que encontram pelo menos

Categoria	Nm. de nós	Recursos Únicos	Categoria	Nm. de nós	Recursos Únicos
A	5000	5	*G*	75	5
B	2500	5	*H*	40	5
C	1250	5	*I*	30	5
D	625	5	*J*	20	4
E	300	5	*K*	10	2
F	150	5			

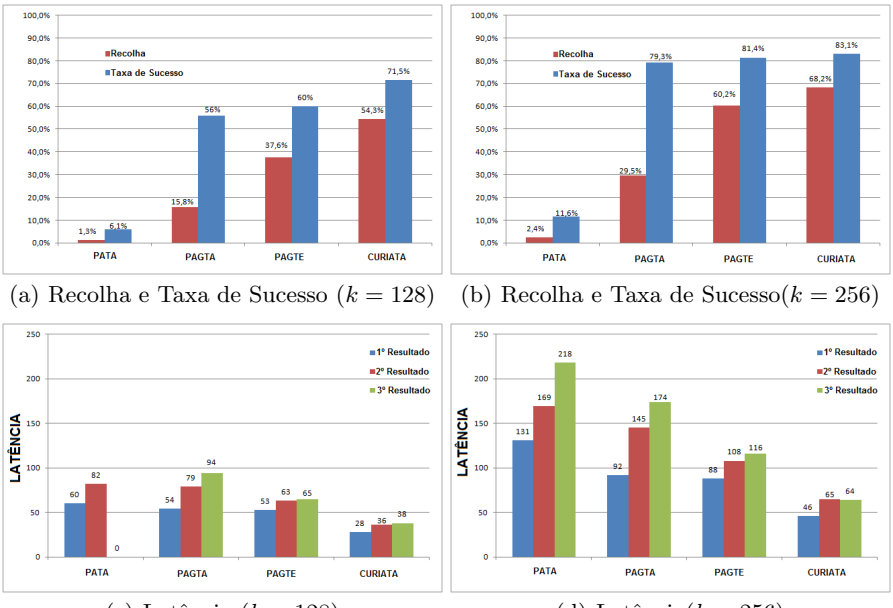
Tabela 1. Distribuição de recursos na rede

uma cópia do recurso procurado; ii) *Recolha:* Percentagem de cópias do recurso procurado que é encontrada pela interrogação (em comparação com o número total de cópias desse recurso existentes no sistema); iii) *Latência:* Número de saltos necessários para encontrar x cópias do recurso procurado. Em particular, estamos interessados nos valores de latência associados a encontrar 1, 2 e 3 cópias do recurso.

Desempenho Global Apresentam-se agora os resultados globais para as interrogações no cenário descrito acima. Neste caso os resultados não são discriminados de acordo com as diferentes classes de popularidade das categorias, dado que o objectivo é fornecer uma visão geral do desempenho do *Curiata*.

Como ilustrado nas figuras Fig 1(a) e Fig 1(b), o *Curiata* possui um desempenho superior às restantes arquitecturas em termos de taxas de sucesso e recolha. Os valores de recolha para a arquitectura PATA quase duplicam quando k passa de 128 para 256 dado que os percursos aleatórios conseguem explorar o dobro dos nós. Note-se no entanto que o desempenho do *Curiata* é muito melhor que as restantes soluções com $k = 128$. Isto é um claro sinal de que o *Curiata* consegue atingir bons resultados mesmo com valores mais conservadores de k.

Naturalmente, num caso limite em que k seja um valor suficientemente grande, todas as arquitecturas (PATA,PAGTA,PAGTE e *Curiata*) conseguiriam encontrar todos os recursos existentes na rede, atingindo taxas de Recolha e Sucesso de 100%. No entanto a eficiência do *Curiata*, atingindo elevadas taxas de sucesso e recolha, para valores baixos de k é um sinal claro do menor custo (em termos de comunicação) da nossa solução.



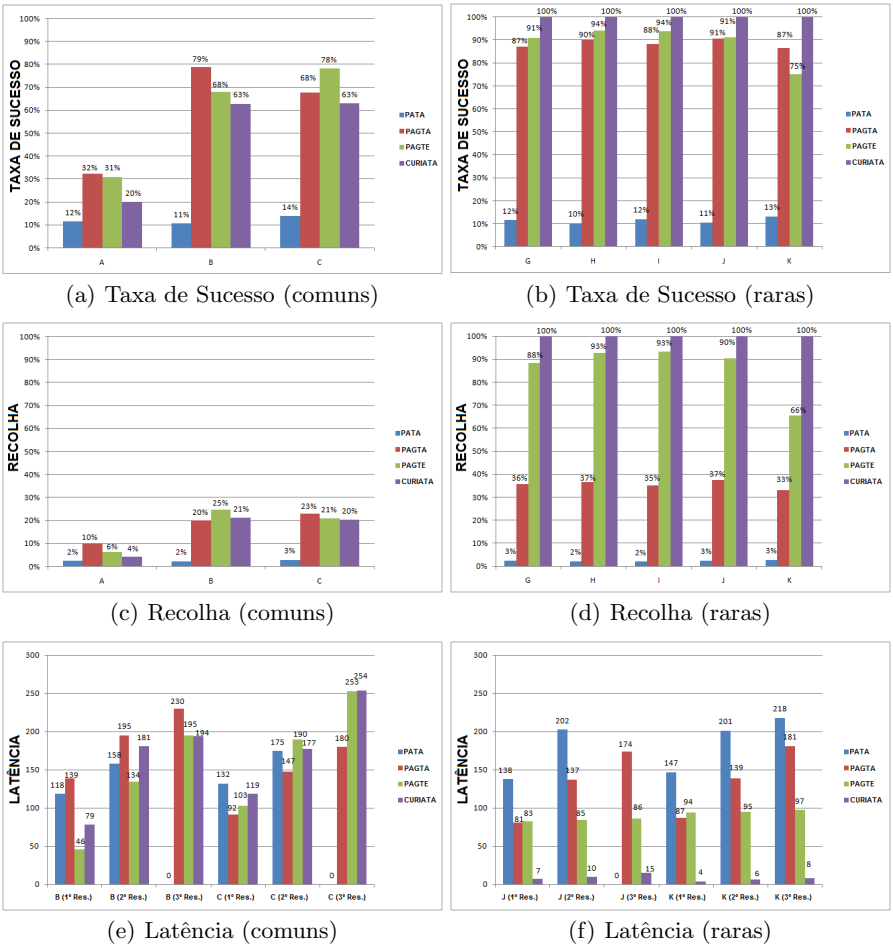
(a) Recolha e Taxa de Sucesso ($k = 128$) (b) Recolha e Taxa de Sucesso($k = 256$)

(c) Latência ($k = 128$) (d) Latência($k = 256$)

Figura 1. Desempenho Global

As figuras Fig 1(c) e Fig 1(d) apresentam os valores de latência. Devido ao encaminhamento na DHT, o *Curiata* apresenta valores de latência significativamente menores. Note-se que a falta de suporte da DHT leva a um cenário onde não existem diferenças significativas no número de saltos necessários para encontrar o primeiro resultado para uma interrogação. Adicionalmente, o enviesamento da topologia da *curia* combinado com a utilização da DHT permite ao *Curiata* apresentar valores de latência mais baixos associados à localização do segundo e terceiro resultados para cada interrogação. Note-se que o valor de latência para o terceiro resultado utilizando PATA é igual a zero dado que esta solução não localiza 3 resultados numa mesma interrogação para k=128. O mesmo não sucede para k=256.

Dado que o rácio de sucesso não é de 100%, quando se altera o valor de k de 128 para 256 o número de saltos na rede aumenta. Isto sucede devido ao au-



(a) Taxa de Sucesso (comuns)

(b) Taxa de Sucesso (raras)

(c) Recolha (comuns)

(d) Recolha (raras)

(e) Latência (comuns)

(f) Latência (raras)

Figura 2. Desempenho das Procuras na Categorias Raras e Comuns

mento do número de interrogações que de facto conseguem localizar pelo menos um resultado. Note-se que estes resultados apenas tomam em consideração o número de saltos para interrogações que tiveram sucesso (isto explica os resultados apresentados na Fig 1(d), onde o número de saltos na rede necessários para o terceiro resultado é menor do que para o segundo resultado quando $k = 256$ utilizando o *Curiata*).

Desempenho por Popularidade da Categoria A Fig 2 apresenta os resultados para interrogações que visam recursos em categorias raras (nomeadamente, as categorias G, H, I, J, e K) e categorias comuns (as categorias A, B, e C). Devido a constrangimentos de espaço apenas exibimos os resultados para $k = 256$, já que este é o melhor cenário para as restantes soluções consideradas.

Os resultados obtidos mostram que o *Curiata* não oferece vantagens significativas para interrogações que visam recursos em categorias comuns. Isto sucede porque a DHT não é necessária quando as categorias são muito populares (dado que os recurso de categorias comuns estão disponíveis em todas as regiões). De facto, a performance do *Curiata* para categorias comuns é até ligeiramente inferior à das arquitecturas PAGTE e PAGTA. Tal sucede dado que se torna desnecessária a utilização da camada estruturada para encaminhar as interrogações para zonas da *curia* correspondentes a essas categorias (dado estas serem muito comuns).

Em contraste, o *Curiata* distingue-se no caso particular de interrogações relativas a recursos pertencentes a categorias raras. Neste caso, o *Curiata* consegue atingir uma recolha e uma taxa de sucesso perfeitos sendo o seu desempenho superior a todas as outras soluções em termos de latência. A Fig 2(f) mostra pormenorizadamente os resultados para a latência apenas para as categorias raras J e K. Os resultados mostram que o *Curiata* oferece um ganho ao nível da latência muito significativo quando comparado com as restantes alternativas. Isto advém do facto de a DHT posicionar as interrogações na região relevante da camada não-estruturada de uma forma bastante precisa. Dado que estas regiões são pequenas, o *Curiata* consegue facilmente visitar todos os nós relevantes e assim localizar eficientemente todas as cópias dos recursos que são visados por cada interrogação.

5 Conclusões

Neste artigo introduzimos o *Curiata*, uma arquitectura para localização de recursos em sistemas P2P de larga escala. O *Curiata* combina técnicas de redes estruturadas e não-estruturadas de modo a oferecer uma elevada recolha e baixa latência para interrogações complexas que visam localizar recursos raros. O *Curiata* não apresenta uma perda de desempenho na resolução de interrogações relativas a recursos raros em categorias comuns quando comparado com outros esquemas. Estes resultados indicam que, quando se procura por um recurso que pertence a múltiplas categorias, são obtidos melhores resultados quando as categorias mais raras são utilizadas de forma preferencial. Como trabalho futuro, planeamos explorar mais aplicações para o *Curiata*. Por exemplo, planeamos integrar a DHT *Cubit*[21] no *Curiata* para desenvolver uma infra-estrutura descentralizada de rastreio e localização de *torrents* que consiga lidar com erros cometidos pelos utilizadores quando descrevem e classificam os conteúdos.

Referências

1. Bittorrent. `http://bittorrent.org/beps/bep_0003.html`.
2. Coral. `http://www.coralcdn.org/`.
3. Gnutella. `http://rfc-gnutella.sourceforge.net/`.
4. Napster. `http://www.napster.com`.
5. *An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol*, 2006.
6. A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In *In Proc. of the 2nd P2P'02*, page 33, Washington, DC, USA, 2002. IEEE Comp. Society.
7. Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like p2p systems scalable. In *SIGCOMM '03: Proc. of the 2003 conference*, pages 407–418, New York, NY, USA, 2003. ACM.
8. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *Proc. of the 22nd ICDCS'02*, pages 23–32, 2002.
9. C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.*, 63(3):241–263, 2006.
10. M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris. The Peersim simulator. `http://peersim.sf.net`.
11. E. Korpela, D. Werthimer, D. Anderson, J. Cobb, and M. Leboisky. Seti@home-massively distributed computing for seti. *Computing in Science and Engineering*, 3(1):78–83, Jan/Feb 2001.
12. J. Leitão, J. P. Marques, J. Pereira, and L. Rodrigues. X-bot: A protocol for resilient optimization of unstructured overlays. In *Proc. of the 28th IEEE SRDS'09*, pages 236–245, Niagara Falls, New York, U.S.A., September 2009.
13. J. Leitão, J. Pereira, and L. Rodrigues. Hyparview: a membership protocol for reliable gossip-based broadcast. In *Proc. of the 37th IEEE/IFIP DSN'07*, pages 419–429, Edinburgh, UK, June 2007.
14. Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.
15. R. Raman, M. Livny, and M. Solomon. Matchmaking: An extensible framework for distributed resource management. *Cluster Computing*, 2(2):129–138, 1999.
16. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. of the 2001 ACM SIGCOMM Conference*, volume 31, pages 161–172, New York, NY, USA, October 2001. ACM.
17. P. Reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *(Unpublished Manuscript)*, pages 21–40, 2002.
18. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Int. Conf. on Distributed Systems Platforms*, pages 329–350, November 2001.
19. I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, pages 149–160, New York, NY, USA, 2001. ACM.
20. D. Tsoumakos and N. Roussopoulos. Analysis and comparison of p2p search methods. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 25, New York, NY, USA, 2006. ACM.
21. B. Wong, A. Slivkins, and E. G. Sirer. Approximate matching for peer-to-peer overlays with cubit. Technical report, Computing and Information Science Technical Report, Cornell University, Dec. 2008.
22. Beverly Yang and Hector Garcia-Molina. Designing a super-peer network. *Data Engineering, International Conference on*, 0:49, 2003.