University of Surrey

Improving Query Routing Efficiency in Peer-to-Peer Networks

N. Antonopoulos and J. Salter

March 2004

Submitted for publication in Information Processing Letters

Computing Sciences Report

CS-04-01

One of the authors (JS) gratefully acknowledges the support of this work by the EPSRC under its Doctoral Training Grant scheme.

Improving Query Routing Efficiency in Peer-to-Peer Networks

Nick Antonopoulos and James Salter

Department of Computing, University of Surrey, Guildford, Surrey, United Kingdom. GU2 7XH

{n.antonopoulos, j.salter}@surrey.ac.uk

Keywords: data structures, information retrieval, peer-to-peer network, distributed hash table

1. Introduction

Distributed Hash Tables (DHTs) such as Chord [4] have been proposed as structured lookup services for Peer-to-Peer (P2P) networks. Chord organises nodes into a ring which requires O(log *n*) hops to route a query through an overlay of *n* nodes. By organising the network as a ring of Chord rings and adding further structural elements and intelligence, we are able to further reduce the hops required in average case scenarios.

Researchers have realised the benefits of adding extra structure to increase scalability and reduce message counts. Example work includes PLANES [5], where one Chord ring is simply segmented into several smaller Chord rings with little or no regard to the information contained in each ring. Each altruistic node is a member of both topology layers, with no separation between clusters and the higher-level topology. In [2]'s case study instantiation of a modified version of Chord, reduction in hops can only be achieved by assuming more powerful machines with greater availability and connectivity are inserted into the top level overlay network and have less probability of failure than nodes at the lower level. Their goal has been to improve overlay network performance by exploiting locality and availability rather than the relationship between data elements stored.

We present enhancements to this previous work by adding Preference Lists and intelligently creating super nodes and keyword rings only when required, reducing the possibility of over-fragmentation and enhancing scalability and performance.

2. Ring of Chord Rings

Our resource discovery system is designed to answer keyword-value queries as opposed to the keyword only queries that have been the focus of most previous research. Thus, the super ring layer of our topology (figure 1) organises a Chord ring of indexed keywords, splitting the index over the nodes (super nodes) in the ring. Super nodes are responsible for pointing to one or more keyword rings. Each keyword ring is a Chord ring of indexed values associated with a single keyword. Each node in the keyword ring holds a list of IP addresses of machines hosting resources matching the value(s) and keyword for which the node has responsibility. Previous research such as [5] has utilised the hierarchical topology to simply partition groups of nodes into clusters rather than creating a true hierarchy of labels as in our system. Effectively, our system looks up two independent but related keys whereas others lookup a single key in two stages.

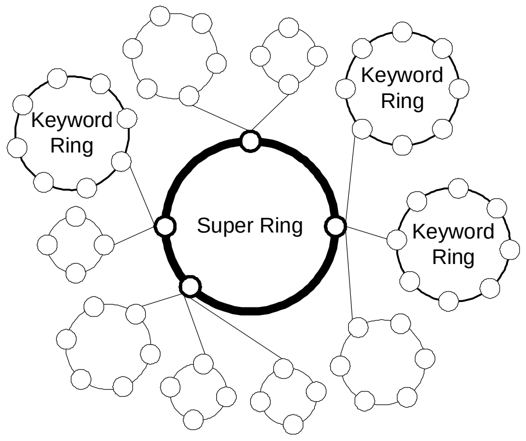

Figure 1: Two-layered topology featuring a central super ring pointing to multiple keyword rings

3. Querying

A query q_{kv} comprises two components, a keyword q_k and a value q_v. The first stage in query processing utilises a Chord lookup to locate the super ring node responsible for identifier h_{qk} where h_{qk} is the hash of q_k. This node holds the address of a node n_{qk} (or addresses of multiple nodes for fault tolerance) in the keyword ring N_{qk} responsible for keyword q_k.

The second querying stage enters keyword ring N_{qk} via node n_{qk} to locate the node n_{qv} responsible for identifier h_{qv} where h_{qv} is the hash of q_v, again using the standard Chord lookup algorithm. Node n_{qv} lists all resources matching q_{kv}. This list R_{qkv} is returned to the query originator so it can choose a resource r_{qkv} such that $r_{qkv} \in R_{qkv}$.

We can show a reduction in hops required to route a query over a standard Chord implementation: Given a Ring of Chord Rings *C* with *s* nodes in the super ring and *k* nodes in each keyword ring, then the comparable single layered Chord ring *C'* has one node for each node in all keyword rings, that is $w \cdot s \cdot k$ nodes where *w* is the number of keyword rings for which each super node is responsible. Given a ring *C* (or *C'*) denoted by *WC(C)* being the worst case number of hops required to route a query and *AC(C)* being the average case number of hops required to route a query, then from [4] for a single Chord ring $WC(C') = \log_2(w \cdot s \cdot k)$ and $AC(C') = ½\log_2(w \cdot s \cdot k)$. For a Ring of Chord Rings $WC(C) = WC(\textit{super ring hops})+1+WC(\textit{keyword ring hops}) = \log_2(s)+\log_2(k)+1 = \log_2(s \cdot k)+1$ and $AC(C) = ½\log_2(s \cdot k)+1$.

Lemma 3.1. *With C and C' as above then if w>2 then WC(C')>WC(C) and if w>4 then AC(C')>AC(C).*

Proof. If $w>2$ then $WC(C') = \log_2(w \cdot s \cdot k) > \log_2(2 \cdot s \cdot k) = \log_2(s \cdot k)+\log_2(2) = \log_2(s \cdot k)+1 = WC(C)$. If $w>4$ then $AC(C') = ½\log_2(w \cdot s \cdot k) > ½\log_2(4 \cdot s \cdot k) = ½\log_2(s \cdot k)+½\log_2(4) = ½\log_2(s \cdot k)+1 = AC(C)$. □

Table 1 illustrates the reduction in hops required to route through a super ring hosting information on 10000 keyword rings, depending on the number of keyword rings each super ring node is responsible for. A 50% reduction of worst case hops required is realised by making each super ring responsible for 1% of the keyword rings in the system.

Table 1. Reduction in routing hops required if super ring nodes become responsible for multiple keywords.

№. keyword rings per super ring node	**Worst-Case Hops**
1	14
2	13
5	11
10	10
20	9
50	8
100	7

The ability to host information about multiple keyword rings on a single super node comes from our detachment of the two topology layers, at a cost of one extra hop between the super ring and selected keyword ring. In [2] nodes in their top layer are also members of the bottom layer, meaning detachment would be impossible unless bottom layer nodes were permitted to be members of more than one group.

Further reduction in hops can be realized by introducing preference lists [1]. If a preference list L stores the address of a keyword ring N_{qk} when a query q_{kv} containing keyword q_k has been submitted, the super ring can be bypassed completely and worst case hops becomes $\log_2(k)+1$ where k is the number of nodes in N_{qk} and $k>0$. Where a preference list also lists resources R_{qkv} matching value q_v for keyword q_k, the entire indexing structure can be bypassed, effectively reducing the hop count to 0 for queries where $q_{kv} \in L$. The lookup procedure is shown in pseudocode in figure 2.

```
sub Search_Keyword_Value($q_k$, $q_v$)
    if $q_k \notin L$ then
        $h_{qk}$ = Super_Ring_Hash($q_k$)
        $n_{qk}$ = Find_Keyword_Ring_Node($h_{qk}$, $q_k$)
        $h_{qv}$ = Keyword_Ring_Hash($q_v$)
        $R_{qkv}$ = Search_Keyword_Ring($n_{qk}$, $h_{qv}$, $q_v$)
    else  // utilise preference list information
        if $q_v \in L$ then
            $R_{qkv}$ = Get_Resources_From_Pref_List($q_k$, $q_v$)
        else
            $n_{qk}$ = Lookup_KRing_EntryPoint_In_Pref_List($q_k$)
            $h_{qv}$ = Keyword_Ring_Hash($q_v$)
            $R_{qkv}$ = Search_Keyword_Ring($n_{qk}$, $h_{qv}$, $q_v$)
        end if
    end if
    return $R_{qkv}$
end sub
```

Figure 2. Pseudocode illustrating lookup procedure.

An advantage of separating a query into keyword and value components and splitting them into the two tiers of the topology is that a traversal of the super ring is all that is necessary to determine the non-existence of a keyword. In Chord and other hierarchical topologies utilising a single key, a traversal through the entire topology is necessary. Therefore, with a Ring of Chord Rings, negative responses can be received quicker than positive responses, leading to a reduction in wasted hops in our system.

Lemma 3.2. *With C, C' and q_k from above and Q_k being the set of all keywords in the system, then if $q_k \notin Q_k$ then WC(C')>WC(C) and AC(C')>AC(C).*

Proof. If $q_k \notin Q_k$ then $WC(C') = \log_2(w \cdot s \cdot k) > \log_2(s) = WC(C)$ and $AC(C') = ½\log_2(w \cdot s \cdot k) > ½\log_2(s) = AC(C)$. □

The system can support multiple keyword-value pairs in a query (utilising a boolean AND operation) by first selecting a keyword-value pair that is likely to yield a result set in the minimum number of messages (for example, a keyword listed in a Preference List would be selected over one where a traversal of the super ring would be necessary). The query is processed as if it were for the single keyword-value pair. A further query is then sent to each machine matching the single pair query until one matching all keyword-value pairs is found.

4. Registering Resources

A new resource *r* registers itself with the system by submitting an update query using the standard query methods described above for each r_{kv} keyword-value combination associated with it. The worst case number of hops required to register a resource is therefore $y(\log_2(s \cdot k)+1)$ where *y* is the number of keyword-value combinations to register the resource against.

If a resource wishes to register itself against a new keyword that does not exist within the system, a new keyword ring N_{rk} must be created. However, the ring is only created once the number of q_k queries for the keyword or the number of registered resources for r_k rises above a threshold. Until that point, information regarding r_{kv} is held at the super node responsible for identifier h_{rk} where h_{rk} is the hash of r_k. If a super ring node does not have available capacity to cope with expected query demand for the new keyword q_k, a new super ring node will be selected and inserted into the super ring. We call this process Lazy Ring Creation. The method enables us to maximise the use of available node capacity and achieve the properties exposed in Lemmas 3.1 and 3.2.

Nodes are selected to participate in the new keyword ring or to become a new super ring node by traversing a Chord ring containing every physical machine in the structure to find a machine with the required capacity *z*. A random identifier *i* is generated and a request routed around the ring using the standard Chord method towards the physical machine p_i. The machine at every hop is tested to determine whether it has the spare capacity $Z \geq z$ to become a member of the new keyword ring. If machine p_i is reached and no machines with $Z \geq z$ have been found, the request continues to be routed around the ring between neighbouring nodes until it is resolved or machine p_i is again reached (meaning there are no machines in the network with $Z \geq z$, thus the network is close to full capacity and possible collapse). Figure 3 presents the node selection heuristic in pseudocode.

```
sub Get_Available_Node(c)
    i = Random(max_id_in_physical_ring)
    nodefound = -1
    do
        Chord_hop_towards(i)
        if $Z \geq z$ then
            nodefound = current_node_id
        end if
    until at $p_i$ or nodefound > -1
    if nodefound = -1 then
        do
            if $Z \geq z$ then
                nodefound = current_node_id
            end if
            Hop_to_next_node()
        until at $p_i$ or nodefound > -1
    end if
    return nodefound
end sub
```

Figure 3. Node selection heuristic.

The above process is continued to find the required number of nodes, the keyword ring is built and then populated with resource information held in contention.

Given a ring with *P* nodes, *G* of which have $Z \geq z$, then if we make m-1 hops the probability of finding *u* nodes with $Z \geq z$ is a hypergeometric distribution [3]: $\Phi = \frac{\binom{G}{u-1}\binom{P-G}{m-u}}{\binom{P}{m-1}}$. Thus the probability of requiring *m* steps to find *k* nodes with $Z \geq z$ is Φ multiplied by the probability that the next node has $Z \geq z$: $\theta = \frac{G-u+1}{P-m+1}$ where there are P-m+1 remaining nodes of which G-u+1 have $Z \geq z$. Thus the mean number of hops required is $M(P,G,u) = \sum_{m=u}^{P-G+u} m\Phi\theta = m\frac{\binom{G}{u-1}\binom{P-G}{m-u}}{\binom{P}{m-1}}\frac{G-u+1}{P-m+1}$. The worst case number of hops required is $\log_2(P)+P-G+u$ if $G\geq u$ and $\log_2(P)+P$ if $G<u$ (there are not enough nodes with $Z\geq z$ in the ring to satisfy the request).

In Figure 4 the mean number of hops required to find a node with $Z\geq z$ is plotted for selected values of u against varying G, in a ring where P=100 nodes. Similar plots can be achieved for different values of P. It is clear that the node selection heuristic is scalable in usual case scenarios where a reasonable number of nodes have spare capacity compared to the number of nodes being sought.

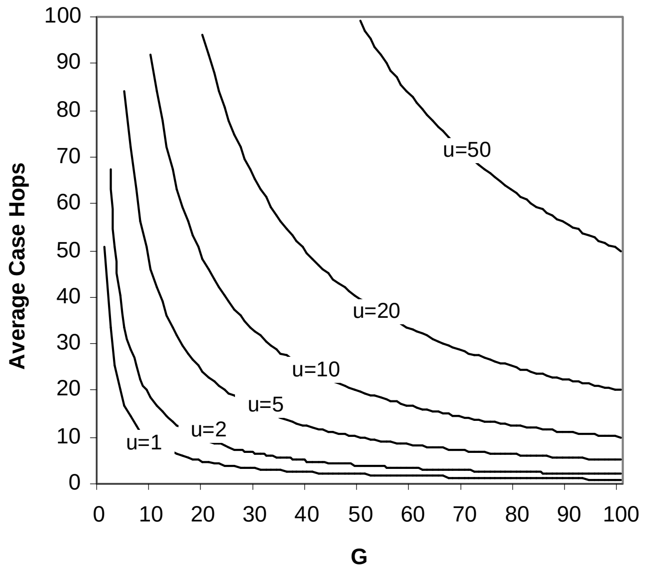

Figure 4. Mean number of hops required to find u nodes in a 100 node physical node ring with G nodes having $Z\geq z$.

We do not hold an ordered global list of the "best" nodes, as this would lead to one node being targeted with requests from multiple nodes simultaneously building keyword rings. The random selection of identifier i and the initial Chord routing reduces the likelihood of multiple requests concurrently targeting a node in the ring with spare capacity.

[2] select their superpeers from a list of local candidate nodes. This could lead to situations where a group comprises only low powered nodes, one or more of which would be promoted to the top-level overlay, thereby reducing performance of the part of the topology shared system-wide. In addition, this method involves a monitoring process and the periodic broadcast of an ordered list of candidate nodes

to all the peer nodes in the group. The authors do mention that the broadcast would not be cost-effective in large groups and peers could learn “lazily” about network changes. However, the monitoring process must continually take place, generating an unbounded number of messages over time. If nodes actively send updates to superpeers in their group, the message cost for every node to send an update is O(n) every update cycle where n is the number of nodes in the network (approximately P messages in a system of equivalent size to ours). In a dynamic environment such as a P2P network, this monitoring must take place frequently to maintain accurate information, generating a large and unscalable volume of messages.

In comparison to the push strategy outlined in [2], our pull strategy does not generate any update messages. Instead, messages are only used when an available node is searched for in the ring of physical nodes. Figure 4 shows that in the majority of scenarios the mean hops required to discover available nodes is much smaller than P. As the message cost for the push strategy is at least P every update period, it is clear the pull strategy should generate less messages in all scenarios apart from where many nodes need extra capacity simultaneously and there are very few nodes with spare capacity available (the network is close to collapse).

Additionally, using Lazy Ring Creation maximises the use of super node capacity and makes the selection of new nodes a rarer event than in an eager scheme where keyword rings are automatically created and, as in [2], a super node must be created for *each* keyword ring. This also has the side-effect of reducing the size of the super ring, meaning less hops are required to traverse it during query processing.

5. Conclusions

We have shown that, by creating two independent topology layers and associating a super node with multiple keyword rings, we are able to reduce the number of hops required to route a keyword-value pair query through a Chord-like distributed hash table. The non-existence of a keyword can be determined by routing a query through only the super ring layer, rather than requiring a full traversal as would be necessary in other systems. By using Lazy Ring Creation we reduce the number of super nodes and keyword nodes created, thereby increasing the scalability of the network and reducing further the number of hops required to route a query. Selection of nodes to participate in rings by traversing a ring of physical nodes has been shown to be more scalable than a method involving the constant monitoring of candidate nodes.

References

[1] N. Antonopoulos, J. Salter, Towards and Intelligent Agent Model for Resource Discovery in Grid Environments, in: Int'l Conf. Applied Computing 2004, Lisbon, Portugal, 23-26 March 2004, pp. II 187-II 191.

[2] L. Garcés-Erice, E. W. Biersack, K. W. Ross, P. A. Felber, G. Urvoy-Keller, Hierarchical Peer-to-peer Systems, in: Parallel Processing Letters, Vol 13, No 4, December 2003, pp. 643-657.

[3] W. Mendenhall, T. Sincich, Statistics for Engineering and the Sciences, 1995, Prentice-Hall, pp. 178-180.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan, Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, in: ACM SIGCOMM 2001, San Diego, CA, USA, August 27-31, 2001, pp. 149-160.

[5] P. Triantafillou, PLANES: The Next Step, in: Peer-to-Peer Network Architectures, SIGCOMM Workshop on Future Directions in Network Architectures (FDNA-03), Karlsruhe, Germany, August 27, 2003.