Evaluation of Dynamic Query Abolishment Methods in Heterogeneous Networks

Elena Meshkova, Janne Riihijärvi and Petri Mähönen
Department of Wireless Networks, RWTH Aachen University
Kackertstrasse 9, D-52072 Aachen, Germany
Email: {eme, jar, pma}@mobnets.rwth-aachen.de

Abstract—We compare the performance of various dynamic query abolishment mechanisms in different unstructured overlay network topologies such as found in several P2P systems. We specifically focus on techniques based on iterative deepening and checking. Both unintelligent and intelligent variants of the methods are used in the study. Additionally, we propose a new mechanism called the chasing wave based on the use of increasing delays for search packets on the forwarding nodes. We show that the proposed chasing wave algorithm trades effectively the increase in propagation delay to substantially lower overhead. The performance of the methods are compared in several network configurations and using several metrics. We make concrete proposals on the suitability of the specific dynamic query abolishment methods for different search algorithms.

I. INTRODUCTION

Service discovery systems have become increasingly important and popular in the last years. Among them are the systems built on top of unstructured overlays, such as unstructured Peer-to-Peer (P2P) systems, that have gained more and more attention due to their resistance to node failures and malicious attacks. Unstructured P2P systems also support partial-matched queries, which allows them to solve in a sense “loosely” formulated queries. Of course, P2P networks are not anymore confined to the research domain, as millions of people use these systems every day. This proliferation of P2P traffic has also created the need to consider the overhead introduced by search and discovery protocols.

The main challenge for unstructured overlays is scalability, i.e. without imposing additional hierarchy on the network (like clustering in KaZaA [1]) it is difficult to support more than some thousands of nodes. There are basically two ways to reduce the imposed search overhead. Either one creates very precise search algorithms carefully designed to be bandwidth-efficient [2], or uses dynamic query abolishment methods (DQAM) to stop search packet propagation after the query answered. The combination of the above two methods can be very efficient [3]. We will also introduce some adaptive techniques to help DQAMs to adapt to changes in the network.

In our study we focus on both adaptive (or “intelligent”) DQAM methods, as well as their unmodified variants. These can be used with virtually any search technique. An appropriately chosen DQAM can further improve the performance of almost any search protocol. We conduct simulations and provide detailed results for nine different protocol combinations: flooding, random walk (RW), iterative deepening (ITD), adaptive iterative deepening (AITD), adaptive iterative deepening with RTT limit (AITD-RTT), checking, adaptive checking (ACHK), the chasing wave (CHW) and the adaptive chasing wave (ACHW). The last two are new query abolishment techniques introduced later in this paper.

The rest of the paper is structured as follows. In the next section we present the methods studied in more detail. In Section III we shortly describe the chasing wave. The simulation environment and set-up are given in Section IV. Section V provides the results obtained together with the associated discussion, and the conclusions are finally drawn in Section VI.
same node numerous times, each time being forwarded to a previously unvisited neighbor. The random walk in many cases considerably reduces the overhead imposed on the network tradeoff being the reduced hit rate, increased round-trip time\(^2\), and highly variable performance.

**B. Dynamic query abolishment methods (DQAM)**

In order to overcome the above mentioned problem of fixed TTL value different DQAMs can be used. The considered dynamic query abolishment methods can be divided into two groups, depending on which of the basic search techniques they rely on. Iterative deepening and its derivatives are flooding-based methods. Checking and its derivatives are RW-based methods. The new chasing wave mechanisms are tested in both groups.

**Iterative deepening (ITD)** is mostly used with flooding. Although it can be used with random walk, this does not make an efficient combination. ITD makes use of successive floods with increasing TTLs [8], so that the searched network diameter gets increased with each iteration. TTL grows until either the searched object is found or the query timeout expires. The method performs well when the search for popular, well-replicated items is conducted. For rare items ITD can produce even worse results than the normal flooding [8].

We also study another variation of iterative deepening, where instead of TTL metric we measure the lifetime of the packet and stop the iteration if the packet exists longer than a certain time. We call the method **iterative deepening with RTT limit (ITD-RTT)**.

**Checking (CHK)** is typically used with random walk techniques [8]. Each search query replica, that is, each walker periodically checks with the source node if the answer to the query was received. If positive answer is received the search stops, otherwise it continues. The checks are done every \(n\) hops.

**III. ADAPTIVE TECHNIQUES**

Adaptive mechanisms are used on top of DQAMs in order to further improve scalability of the search methods\(^3\). Use of adaptive techniques also allows to adjust the overhead/delay tradeoff to the timing requirements of the service searched. If the service needs to be found in minimal time, then DQAMs can be tuned to produce minimal delay at the expense of high overhead, virtually performing as the original search mechanism. If the search time is not critical, then DQAM parameters can be changed to save network resources at the expense of additional delay. These adaptive techniques make use of the previous search results and change DQAM parameters *initial propagation steps* and *propagation steps*, which will be described next.

Same basic approach can be applied to virtually any DQAM. At first the query packets are propagated for a certain hop distance (in case of ITD-RTT for a certain time) without using any query abolishment technique. The propagation depth is equal to the hop distance at which the previous search request for the same resource was answered. If with the initial query propagation no results were obtained, the parameter *propagation step* is used. In the case of ITD it is the value on which in the searched network diameter gets increased each iteration; for checking the number defines how often a check with the source node is done. The *propagation step* is obtained as the change in the hop distance between successive searches conducted by the same node for the same object. The parameter value is obtained using exponential averaging by

\[
\text{Step}_\text{size} = \alpha \cdot \text{Step}_\text{size} + (1 - \alpha) \cdot M, \quad (1)
\]

where \(\text{Step}_\text{size}\) is the step size used, \(M\) is the hop/time distance currently obtained and \(\alpha\) is a numerical smoothing coefficient (experimentally found to be near-to-optimum at a value of \(\frac{5}{6}\) for the test cases).

If the search does not succeed with these values, then the parameters *initial propagation steps* and *propagation steps* are increased in power-law fashion for iterative deepening and checking. This is in order not to produce extra overhead, if the resource simply does not exist in the network. In the extreme case the protocols function as the original search methods.

**IV. CHASING WAVE – A NEW DQAM**

In this section we introduce a new algorithm, falling in the class of dynamic query abolishment methods, called the *chas ing wave (CHW)*. The functioning of the chasing wave is analogous to the chasing game. In our case we chase packets. Whenever the query is answered, we chase the remaining query replicas and stop them. To our knowledge there exist no DQAM schemes similar to the Chasing wave.

All replicas of the query are propagated through the network, but subjected to an increasing delay at nodes as the hop distance from the source node becomes greater. The delay values depend on the time the search packet spends traveling the network. At every node the query replica leaves a mark pointing where it travels further and where it came from. Later these marks are utilized by the chasing packets that are formed at the node where the search query is answered.

The chasing packets are sent along all the paths the query replicas were forwarded on. They experience no delays at the nodes. The chasing packet marks every node it arrives at, forbidding the corresponding search query replicas or duplicate chasing packets to be propagated further. The search packets stop traveling either after being stopped by a chasing packet or after reaching a TTL limit, which can be set quite large. The general formula of the delay experienced by the search query at a network node is:
results.

If the search queries for the same resource were answered more than \( k \) times in a row, we switch to the mode that combines iterative deepening and chasing wave approaches. At the expense of additional delay imposed and additional states introduced the chasing wave method reduces overhead compared to the original search methods (flooding and random walk). CHW also can be used with both of these techniques unlike the previously described DQAMs.

The chasing wave can also be used with adaptive techniques. As other DQAMs it makes use of initial propagation steps and propagation steps parameters. The latter defines the value of parameter \( K \), that is the hop distance after which the chasing packet would meet the query replica in the network with identical links. No power-law increase in the parameter values is done in case of unsuccessful searches. Instead, we introduce a special parameter called stop DQAM. It defines a certain hop depth after which the CHW stops using any of its special features such as issuing chasing packets or marking its way though the network, thus saving considerable overhead. Also, after a number of unsuccessful searches for a certain resource the CHW switches to RW or flooding.

If the search queries for the same resource were answered more than \( k \) times in a row, we switch to the mode that combines iterative deepening and chasing wave approaches. At first the search packet is propagated at the initial depth without any chasing traces and packets formed. If the resource is not found, then after a certain timeout\(^4\) the adaptive chasing wave mechanism starts as described previously. While this approach might appear complex, the adaptive chasing wave achieves good and stable performance as shown by the simulation results presented below.

V. SIMULATION SETUP

To evaluate the performance of these protocols and methods in unstructured overlay networks, such are used by many P2P applications, we conducted a number of simulations. The network sizes in our study ranged from 10 to 1000 nodes. Each simulation was carried out for a period of 4500 seconds, with the search requests issued every 5–10 seconds by a single node. Each search packet carried a payload of 200 bytes. The simulations were conducted using QualNet, a commercial product developed by Scalable Network Technologies [9], [10].

A. Topologies

We experimented with both fixed and heterogeneous network types. The fixed networks were built using either random graphs (Erdős and Rényi model [11]) or random scale-free graphs (Albert-Barabási model [12]). The average node degrees were 4.45 and 4.05 respectively.

Heterogeneous networks consisted of one fixed and several wireless components, containing a total of 50% of wireless nodes. As wireless components we simulated 802.11b (WLAN) subnetworks, each consisting of 20–50 nodes. The fixed network component parameters were as described above. Each wireless subnet was connected to fixed network via one link. The network links were given two parameters: throughput and propagation delay. These values were determined using the distributions taken from real measurements for the Gnutella network [13].

For scenarios with node failures we used exponential distributions of uptimes, downtimes and session durations with the averages of 730, 170 and 900 seconds respectively. Such distributions allow to have non-negligible probabilities of link existence. For example, with these parameters a probability that a random 2-hop link is enabled at a single point of time is about 5%.

For graph construction the GENESIS package [14] developed at the RWTH Aachen University was used, together with GNU Scientific Library (GSL). The latter was used to obtain samples from different random number distributions.

B. Service allocation

All the protocols were implemented on the application layer. No background network traffic was included in the simulations. Nodes were assigned services randomly according to a power-law distribution. This assignment did not depend on the connectivity of the nodes. Without considering dedicated servers that were present in some scenarios, nodes on average hosted 3.2 services. If servers were present in the network, then there were 4 servers assigned that hosted 50% of total variety of services. Servers were distributed uniformly across the network. In the scenario with service reallocations instead of 4 servers there were 15 nodes present that allocated 10% of total number of services.

The total variety of services the network possesses was chosen to depend on the network size. If the network was smaller than 100 nodes it provided 10 different services. If it was larger than 100 nodes 200 various resources were allocated. The growth in the number of services allowed to model networks deployed for different purposes.

C. Metrics

We used several metrics to compare the performance of the protocols studied. Hit rate is defined as the ratio of successful

\[
\text{Delay} = \begin{cases} 
0, & \text{if } \text{HOPS}_\text{CUR} \leq M; \\
2 \cdot \text{Traveling Time} + \left\lceil \frac{\text{Traveling Time}}{\text{HOPS}_\text{CUR}} \right\rceil, & \text{if } M < \text{HOPS}_\text{CUR} \leq \text{TTL} \text{ and } K = 0; \\
\frac{\text{Traveling Time}}{K} + \left\lfloor \frac{\text{Traveling Time}}{\text{HOPS}_\text{CUR} K} \right\rfloor, & \text{if } M < \text{HOPS}_\text{CUR} \leq \text{TTL} \text{ and } K > 0.
\end{cases}
\]

\(^4\)The value of all timeouts are also determined from the previous search results.
searches to total number of queries issued. A ratio of total multiple hits to the total number of queries results in the average number of answers per query. We also measured the average hop distance that a query travels before receiving a first response. The average RTT metric refers to the time spent by the node from sending the query to receiving the first response. To measure overhead created by protocols we used two metrics: the average number of bytes per query and the average number of packets per query.

VI. Simulation Results

We conducted three sets of experiments, repeating each individual simulation instance five times. In the first, static scenario, nodes did not fail and the distribution of services remained the same over time. In the second scenario nodes failed, but the topology did not get reorganized, therefore the chances to a node to get its query answered decreased drastically. The distribution of the failures was described in the previous section. The third scenario was the most interesting and realistic one as service reallocations were introduced. After each 5th query, the node changed the variation services it was hosting without changing the service’s number. This scenario aimed to more seriously test the adaptivity of the protocols.

Due to space constraints we do not describe the results from all the scenarios in detail. Instead we introduce general findings through illustrative examples. Figures show the averages over the simulation runs, together with standard deviation observed.

The hit rate decreased from scenario to scenario. The highest one was obtained for the static scenario, then followed the scenario with service reallocations. The protocols showed the worst hit rate in the scenario with node failures. The RW-based protocols achieved lower hit rate than flooding-based approaches. In extreme cases (heterogeneous networks with node failures) the difference reached 40%–50%. As the hit rate of the protocols belonging to one group (RW-based or flooding-based) does not differ significantly, we do not provide the respective graphs.

5As the hit rate of the protocols belonging to one group (RW-based or flooding-based) does not differ significantly, we do not provide the respective graphs.
In the random graph topologies the protocols usually performed better than in the random scale-free networks. This is because random graphs have more equal node degree distribution (Figures 2 and 3). As the graphs show even the overhead generated in random graphs is lower than in random scale-free graphs. The results obtained in heterogeneous networks were typically worse than in the fixed networks due to the presence of wireless components and bottleneck links. The hit rate difference can reach 25%. Nevertheless, the comparative performances of the tested protocols stay the same, i.e. the relations between the amount of overhead generated or delays created are the same as for the fixed topologies (Figures 4 and 2).

The analysis of the performance of the search methods with and without DQAMs and their adaptive versions show that generally for flooding it is beneficial to use dynamic query abolishment methods. The influence of DQAM on RW is not so drastic. The original methods, RW and flooding, achieve the largest number of answers per query, as other methods aim at finding just one answer per query and limit their query propagation depth considerably. DQAMs achieve savings in overhead at the expense of the RTT increase compared to the original search mechanisms.

The use of adaptive techniques is always justified. Either they allow to considerably decrease the produced overhead, as for ITD and checking, or allow to noticeably reduce the delay introduced as for the CHW (Figures 1 and 5).6

As the simulation results show, the use of checking-based mechanisms with RW does not really make sense, as the overhead savings are not that significant. The chasing wave clearly performs better in all test cases. For example its adaptive version used on top of RW increases the RTT only 1.7 times (from 7 to 12 seconds) while at the same time decreasing the overhead by 30% in the random scale-free topology (from 28 kByte to 19 kByte per query) (Figures 2 and 6). Here we see the clear tradeoff the ACHW offers between an increased delay and the overhead savings.

For flooding the use of DQAM clearly makes sense. Adapt-
tive iterative deepening performs the best here, allowing to achieve the minimal overhead (58% decrease in random scale-free graph) at the expense of a slight delay in answering the queries. ACHW looses slightly to AITD (Figure 1). The adaptive chasing wave loses not so much in terms of the overhead, but the latency it produces. Such high delay in small scale-free networks (about 20 seconds) is due to the fact that the method is able to locate few services that appeared on nodes later due to the reallocation. As the network grows bigger the delay gets smaller to about 3 seconds (Figure 5).

As is visible from Figures 5 and 6 for ACHW the average RTT grew with network size in case of RW-based protocols and decreased in case of flooding-based protocols. This effect comes from the nature of RW and flooding. With network growth the performance of RW typically does not become more consistent. The number of hops to answer the query increases with network size, and paths of different length leading to the results are created all the time. For flooding this does not happen and therefore the adaptive parameters change by a smaller amplitude with network size.

The AITD-RTT caused unexpectedly large overhead. This means that either the adaptive mechanism used for the AITD-RTT is not good enough or such a precise tuning mechanism as this protocol has does not prove itself. We experimented further and discovered that AITD-RTT only wins against normal AITD in case of node degree and link speed based allocation of servers, i.e. when servers were placed on the most connected nodes that have the fastest connection links. Even in such cases the advantage of AITD-RTT is quite small, so it does not really make sense to use this protocol (Figure 4).

VII. Conclusions

The simulations conducted across different network topologies show that the adaptive versions of the DQAM perform better across various network sizes and topologies than their unimproved versions. The unintelligent DQAM methods usually do outperform the adaptive ones on very specific network size ranges, which proves that the adaptive algorithms chosen are not the optimal ones, and further research in intelligent techniques should be pursued. The AITD-RTT almost always performed worse than AITD, and it does not really make sense to switch to this iterative deepening model.

In the case of random walk the advantage of use of the DQAM is not so clear, though the adaptive chasing wave can successfully be used in conjunction with intelligent search algorithms. For flooding the adaptive iterative deepening is definitely the technique to be used. The adaptive chasing wave loses to it a bit, though still performing much better than unmodified flooding. The general recommendation for ACHW is to use this method together with informed search mechanisms, especially random-walk-based methods, and techniques that use a combination of flooding and RW for packet propagation or only probabilistic flooding. The chasing packets of the ACHW can be used by the intelligent search mechanisms for carrying their supplementary data, as the chasing packets travel basically through all the nodes that can be of interest to the informed search methods.

Adaptive DQAMs can adjust their performance to trade the imposed delay to the reduced overhead. For time critical applications the adaptive DQAM parameters can be set large, so that no additional delay will be encountered during the search. For other searches the parameters can be adjusted as described earlier, thus saving the network resources.

ACKNOWLEDGMENT

We would like to thank DFG (Deutsche Forschungsgemeinschaft), European Union (RUNES-project) and RWTH Aachen University for the financial support. We would also like to thank Marina Petrova and other members of the chair for fruitful discussions.

REFERENCES