Large-Scale Monitoring of DHT Traffic

Ghulam Memon, Reza Rejaie[†]
University of Oregon
{gmemon, reza}@cs.uoregon.edu

Yang Guo
Corporate Research, Thomson
Yang.Guo@thomson.net

Daniel Stutzbach
Stutzbach Enterprises
daniel@stutzbachenterprises.com

***Abstract*—Studying deployed Distributed Hash Tables (DHTs) entails monitoring DHT traffic. Commonly, DHT traffic is measured by instrumenting ordinary peers to passively record traffic. In this approach, using a small number of peers leads to a limited (and potentially biased) view of traffic. Alternatively, inserting a large number of peers may disrupt the natural traffic patterns of the DHT and lead to incorrect results. In general, accurately capturing DHT traffic is a challenging task.**

In this paper, we propose the idea of minimally visible monitors to capture the traffic at a large number of peers with minimum disruption to the DHT. We implement and validate our proposed technique, called *Montra*, on the Kad DHT. We show that *Montra* accurately captures around 90% of the query traffic while monitoring roughly 32,000 peers and can accurately identify destination peers for 90% of captured destination traffic. Using *Montra*, we characterize the traffic in Kad and present our preliminary results.

I. Introduction

Almost a decade ago, Distributed Hash Tables (DHTs) were presented as an elegant approach to design structured Peer-to-Peer (P2P) networks [9], [12]. DHTs enable individual peers to efficiently search the system and determine the availability of a desired item among participating peers. Despite their appealing features, DHTs were not widely deployed until a few years ago. Therefore, most of the early studies on DHTs focused on extensive simulations, analysis and small scale deployments [5], [4]. The recent availability of widely-deployed DHTs with almost four million concurrent users (*e.g.*, Kad) provided an opportunity to characterize both user- and protocol-driven aspects of a DHT in action [14]. Characterizing different aspects of a deployed DHT not only reveals various opportunities for performance improvement but also sheds light on the interactions between user and protocol dynamics in practice.

Many characterizations of deployed DHTs require accurate measurement of their traffic. A commonly used approach to capture traffic in a deployed DHT, is to add instrumented peers that passively participate in the DHT and log exchanged messages [8], [2]. Deploying a small number of monitoring peers may not provide a representative and sufficiently detailed view of traffic in the system. Alternatively, inserting a large number of instrumented peers artificially increases the number of peers, which may disrupt the target DHT and lead to incorrect results. In addition, this approach requires significant resources. Furthermore, the connectivity of active peers (with real users) and thus observed traffic could be different from passive peers that do not submit any query.

[†]This material is based upon work supported in part by the NSF under Grant No. NeTs-NBD-0627202.

This paper presents a new technique for scalable monitoring of DHT traffic, called Montra. The key idea in Montra is to make monitors minimally visible. This minimizes the disruption of monitors on the system and significantly reduces the required resources for each monitor. We implement Montra in the form of a highly parallel, scalable, python based client and validate it over Kad DHT. We show that our implementation of Montra can concurrently monitor around 32,000 Kad peers using a moderately configured PC (an Intel Core 2 Duo with 1 GB RAM), while dropping 0.009% percent of packets. We use our own crawler [13], running on a separate machine, for continuous discovery of monitored peers. Montra can capture 90% of query traffic observed by monitored peers and identify destination peers for 90% of captured traffic. Finally, we demonstrate the capabilities of Montra by presenting our preliminary characterization of traffic in Kad DHT.

The rest of this paper is organized as follows: Section II provides a brief overview of Kad. In Sections III and IV, we present Montra and validate two angles of its accuracy, respectively. Section V presents our preliminary characterization of Kad traffic using Montra. Section VI reviews the related work and Section VII concludes the paper.

II. Kad Background

In this section, we briefly describe the most relevant features of Kad to serve as a specific DHT for our proposed measurement technique. Kad is a popular DHT system with millions of simultaneous users [11] and is based on Kademlia [6]. Kad is part of the eMule [1] file-sharing software to provide a distributed keyword and file search service. Similar to other DHTs, a Kad peer is involved in *search* and *publish* operations for content (*i.e.*, files and keywords). Performing these operations on a given ID, called the *target ID*, occurs in the following two steps:
(i) Lookup Phase: A client performs a lookup on the target ID to find several alive peers near the target ID. While publishing, content is replicated at 10 nodes to ensure availability of content even after departure of few peers. While searching, queries are sent to multiple nodes to get maximum unique results. During the lookup phase, the client sends a Request message which carries the target ID, and the requested number of contacts c. The value of c reveals whether a lookup message is followed by a publish message (c = 4) or a search message (c = 2). Whenever a peer receives a Request message, it checks its own routing table to find the c closest contacts to the target ID. Then, these contacts are embedded in a Response message and sent to the requesting peer. Both publish and search operations start with Request messages.

(ii) Search/Publish Phase: Once several alive peers near the target ID are identified, the requesting Kad client sends either Search or Publish messages to these target peers. Publish messages contain the cryptographic hash for a keyword or a file, and notify the target node that the request originator is a source for the content. Search messages also contain the cryptographic hash for a keyword or a file. The target node responds to search messages with the addresses of the nodes that previously published the same target ID.

III. Methodology

A common approach for capturing traffic in a DHT is to randomly place a few instrumented peers within the network. Each instrumented peer passively monitors and logs traffic. Using a small number of monitors may not provide a representative view of traffic pattern. Alternatively, inserting a large number of monitors may be infeasible due to the required computational resources. More importantly, such a brute force technique artificially increases the number of peers, and could disturb the traffic pattern [10].

To resolve this conundrum, we notice that traffic can be classified into two categories:

- *Destination traffic*: A lookup request received by peer P_i is destination traffic if P_i's ID is the closest to the target ID in the message using DHT's closeness metric (*e.g.*, XOR distance in Kad).
- *Routing traffic*: Any request that is not destination traffic, is routed toward its destination by peer P_i and is considered as routing traffic for peer P_i.

The rate of destination traffic at peer P_i depends on the popularity of the content that is mapped to P_i's assigned ID space, *i.e.*, destination traffic is primarily user-driven. However, the rate of routing traffic at each peer is determined by peer's connectivity and overall content popularity. We focus on measuring destination traffic because it represents user behavior and more importantly it is much more tractable. In the rest of this section, first, we describe how to monitor the destination traffic at a single Kad peer. Then, we discuss how to minimize the visibility of monitors in order to avoid any disruption in the target DHT. Next, we describe how Montra handles peer churn. After that, we comment on applicability of Montra to other DHTs. Finally, we describe an extension to Montra for obtaining content metadata from Kad.

A. Monitoring a Single Kad Peer

Suppose peer P_t is an arbitrary target peer in Kad. Let A_t be the portion of Kad address space assigned to P_t. Typically P_t is the closest node to the identifiers in A_t. Let P_m be the monitoring peer of target peer P_t. When peer P_t receives a request message from a request originator, P_r, it responds with the set of peers from its routing table that are closest to the requested ID. P_m captures this destination traffic in the following three steps as shown in Figure 1:

1) At the start of the monitoring process, the monitor P_m introduces itself in the DHT in following 2 steps:
   a) P_m places itself next to the target peer P_t in the ID space by setting its ID as $ID(P_m) = ID(P_t)$ XOR 1
   b) P_m adds itself to the target peer's routing table by exchanging Hello messages to become visible to the target peer. [1]
2) When P_t receives a request from the peer P_r, with a destination in A_t, P_t replies with P_m's address because according to P_t's routing table, P_m is one of the closest peers to the requested ID.
3) When P_r learns about P_m, it sends the same request to P_m. Thus, P_m receives a copy of requests destined for A_t. P_r sends a request to P_m because it is looking for several alive peers close to the target ID, in order to publish content at or retrieve search results from multiple peers.

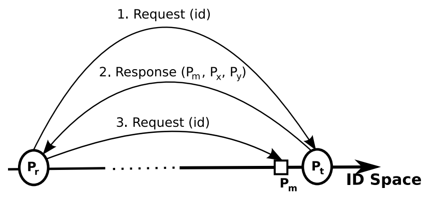


Fig. 1: Message exchanges for adding a monitor and capturing destination traffic

B. Minimally Visible Monitors

As mentioned earlier, large number of monitors, while necessary to collect a complete view of the system, may change and/or disrupt the system. We solve this problem by introducing *Minimally Visible Monitors (or MVMs)*. The basic idea is to minimize the visibility of each monitor, P_m, by maintaining its presence only at its target peer, P_t. P_m only responds to the messages issued by P_t and silently ignores messages from all other peers. Peers that may learn about MVM, from P_t, consider P_m as departed peer and do not add it to their routing tables. As a result, an MVM neither routes traffic nor stores content. Since MVMs are essentially invisible, placing a large number of MVMs does not cause disruption and/or change the system. Furthermore, the lightweight nature of MVMs helps in deploying large number of monitors while using minimal resources.

C. Identifying Destination Traffic

In addition to receiving the majority of destination traffic, P_m may also receive a small fraction of P_t's routing traffic. This occurs when P_m is one of the top *c* contacts in P_t's routing table for some routing requests. For example, this scenario may occur at one hop before a request reaches its destination. A single MVM cannot distinguish between routing

[1]Further details on the required message exchange between the target and monitor peer can be found in [7].

traffic and destination traffic. It has no information whether any other peer closer to the target ID received the same message. In order to accurately distinguish destination traffic from routing traffic, we monitor peers in a continuous region of the ID space. The continuous ID space is called the *monitoring zone*. A zone is specified by x high order bits of the ID (or *prefix*) that is common among all peers in that zone. For example, $0\mathrm{x}a4$ is a prefix for an 8 bit zone (x = 8). Any requests for an ID that has the same zone prefix and enters the monitored zone has a destination in that zone. Therefore, by monitoring all the peers in the entire zone, we capture all destination traffic for the zone. Although a request may be observed by multiple MVMs in a zone, during the post processing phase the closest MVM that received the request is considered its actual destination.

D. Coping with Churn

To accurately monitor the destination traffic at all peers within a given zone in the presence of churn, it is important to quickly identify newly arriving peers and attach a monitor to them. Interestingly, we do not need to remove MVMs for departing peers. Since each MVM is only visible to its target peer, an MVM does not receive any traffic after its target peer departs. Using our high speed crawler [13], we crawl the target zone back-to-back in order to ensure timely discovery of new and departing peers. Then, we attach a new monitor to each new peer and place the monitors of departed peers into the pool of idle monitors for efficient resource management. Note that we only attach monitors to those peers that are not behind NAT boxes. Peers behind NAT boxes do not participate in routing lookup messages.

E. Generality of Montra

While this paper focuses on monitoring Kad, we believe that our proposed technique can be adapted to other real-world DHTs (*e.g.,* Azureus [2], Mojito [3]). Real-world DHTs must ensure the availability of content in the presence of churn. A common technique to achieve this is to lookup multiple peers close to the target ID for both searching and publishing content, as described in Section II. Montra leverages the need to "lookup multiple nodes" to capture each lookup message. Note that actual publish and search messages are not observed by an MVM since they are sent directly to the target peer. However, the lookup message often contains some information (*e.g.*, number of requested contacts in Kad lookup) that reveals whether it is associated with a publish or search message.

F. Extracting Content Metadata

While monitoring Kad, the following information can be extracted from the captured Request messages: *(i)* the type of request (publish or search), *(ii)* the requested content ID, and *(iii)* the ID of the destination peer. From this information, however, we are unable to determine whether a request (search or publish) is for a keyword or a file. Neither can we learn about the characteristics of requested files (*e.g.,* size).

[2]Azureus uses an implementation of Kademlia to operate in trackerless mode.

[3]Mojito is an implementation of Kademlia and is used by Limewire.

We extend our measurement technique to collect more information as shown in Figure 2. When an MVM receives a Request message from the request originator during the lookup phase, it may send a response to the originator. The response does not carry any next hop contacts, but it informs the request originator that the MVM is alive and can receive a request during the second phase. As a result, the MVM receives the (search or publish) requests during the Search/Publish phase that carry the following additional information about the files or keywords being requested: *(i)* the type of content (file or keyword) being requested/published, and *(ii)* the size of the file when the requested content is a file.

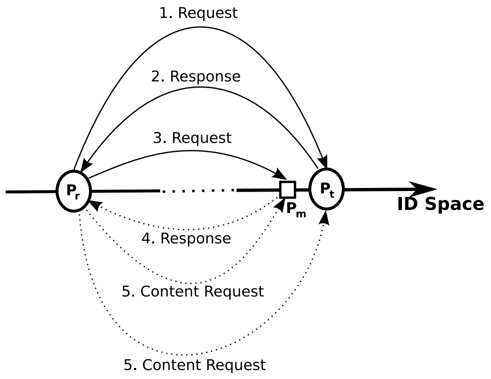


Fig. 2: Extended Technique

This extension, shown with dotted lines in Figure 2, results in a slight disruption to regular operation of the system. More specifically, a content that is published at 10 closest peers, is instead published at 9 closest peers. In addition, sending Response to a Request message increases the visibility of the MVM. To minimize the side-effects of this extension, MVMs respond to requests for a given content ID only once, in order to obtain the above additional information. MVMs do not respond to any further request for the previously observed content ID.

This extension is specific to Kad. We believe Montra, in its original form, can be used to capture traffic from any real-world DHT. However, we also envision such DHT-specific extensions to Montra to extract more fine-grained information from different DHTs.

IV. Validation

In this section, we examine the accuracy of our proposed measurement technique. The accuracy of Montra can be validated from the following two perspectives: *(i)* how accurately Montra catches the traffic destined for IDs in a target zone, and *(ii)* how accurately Montra assigns the captured messages to the right destination peers within the zone. To answer these two questions, we conduct two types of experiments as follows:

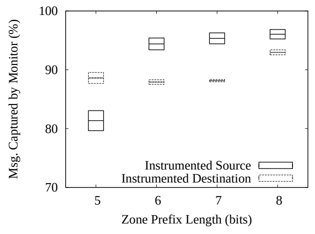
- **Instrumented Source:** An instrumented peer is placed at a random Kad ID outside the monitored zone. It generates request messages with random target IDs within the monitored zone and logs request IDs as well as the observed destination peers. During the post-processing of logs, the number of generated messages is compared with the number of messages captured by Montra. Furthermore, the destination peers recorded by Montra are compared with the actual destination peers observed by the instrumented source.

- **Instrumented Destination:** An instrumented peer (*i.e.*, destination) is placed inside the monitored zone. The destination peer logs all the destination messages it receives. We then examine whether these messages are captured by Montra and whether Montra correctly identifies the instrumented peer as the destination peer.

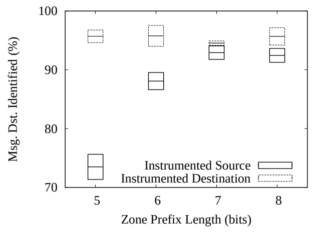
The size of a target zone is the primary factor that affects the accuracy. Therefore, zone size is the main variable for assessing the accuracy of Montra. Note that decreasing the zone prefix length by one bit doubles the size of the target zone. On average, the approximate number of peers in a zone can be estimated as $N_{peers-in-zone} = N_{total-peers}/2^L$, where L is prefix length. For example, a prefix length of 0 includes the entire system. A prefix length of 1 includes half the system. A prefix length of 2 includes one-quarter of the system, and so forth. At the time of this writing, we estimate the total number of non-NAT Kad peers to be around two million. Table I shows the approximate zone size as a function of the zone prefix length. Due to memory constraint, the MVM monitoring tool is able to monitor a zone with prefix length up to five bits.

Prefix Length	Simultaneous Peers
0	2,000,000
1	1,000,000
2	500,000
3	250,000
4	125,000
5	62,500
6	31,250
7	15,625
8	7,812

TABLE I: Approximate Zone Sizes



(a) Percentage of messages that are captured by Montra



(b) Percentage of captured messages that correctly identify the destination

Fig. 3: Montra Validations

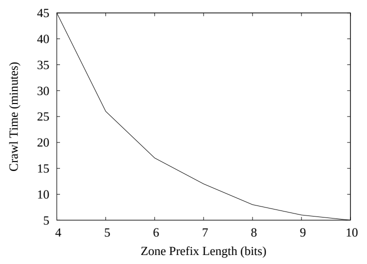


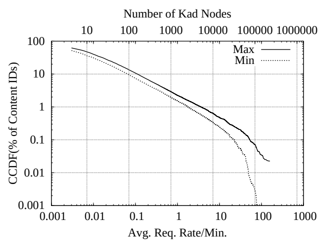
Fig. 4: Crawl duration as a function of zone prefix length

Figure 3a presents the percentage of messages destined to the monitored zone that are correctly captured by Montra in both instrumented source and destination experiments as a function of zone size (*i.e.*, prefix length). The top and bottom lines of each box reflects the 95% confidence interval and the middle line represents the mean. Figure 3a demonstrates that approximately 90% of messages are captured successfully by the monitor regardless of zone size. However, zone size has a more pronounced impact on the accuracy in the instrumented source experiments in Figure 3a. For zones with 6, 7 and 8-bit prefixes, around 95% of messages are captured. But the accuracy of instrumented source experiments drops to around 82% of messages for 5-bit zones. To explain this, we note that decreasing the prefix length of a target zone by one bit doubles the population of peers in a zone which in turn increases the time to crawl a zone. Figure 4 depicts the average crawl time of a zone for a given prefix length. Longer crawling time of a zone leads to a longer delay in discovering the new peers and thus a longer delay in attaching an MVM to new peers which in turn leads to a larger error. Zone size does not have the same impact on the instrumented destination experiments, since we only count the messages that are received by the instrumented peer. The instrumented peer is added at the beginning of the experiments and stays in the system for the entire duration of the experiments, masking the impact of peer churn. Figure 3b presents the fraction of monitored messages that are correctly mapped to their destination peers. In most cases, Montra correctly identifies the destinations of approximately 90% of the messages. However, in the instrumented source experiment over large zones (*i.e.*, 5-bit prefix length), this percentage drops to 73% due to the long crawling duration for larger zone size. *In summary, the validation experiments show that Montra captures close to* 90% *of messages for zones up to* 6*-bit prefix length. Furthermore, Montra correctly pinpoints the destination peer of around* 90% *of the captured messages.*

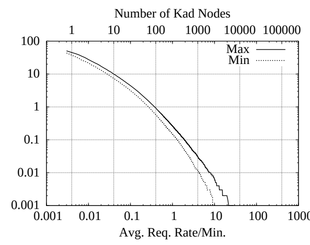
V. TRAFFIC CHARACTERISTICS

To demonstrate the benefits of Montra, we present our preliminary results in characterizing Kad traffic using Montra.

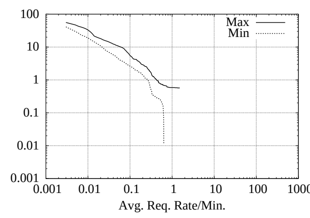
Data Sets: Given our validation results in Section IV, we focus on zones with prefix length of 6 to strike a good balance between monitoring accuracy and the number of monitored peers. A 6 bit zone contains approximately 32,000 non-NAT peers on average. Between May 2008 and August 2008, we



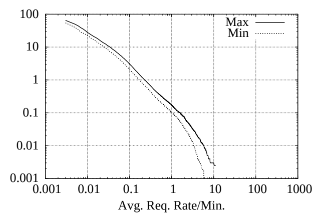
(a) Publish Keyword Rate



(b) Publish File Rate



(c) Search Keyword Rate



(d) Search File Rate

Fig. 5: Min-max envelopes of CDFs for rates of different types of requests

collected 50 traces by monitoring 44 (out of 64) unique 6-bit Kad zones. Each measured zone was monitored for a 6-hour period since publish file requests in Kad are sent once every 4 hours. We started measurement of each zone at different times of the day (namely 3pm, 9pm, 3am, and 9am PST) in order to observe any potential variability caused by the time-of-day.

To succinctly present the Cumulative Distribution Function (CDF) of desired properties for all 44 monitored zones, we use min-max envelopes of these CDFs. More specifically, for each x value, we identify the min and max y values across all CDFs and use them to draw the envelopes. The gap between these min and max lines indicate the variability of the CDF across different zones.

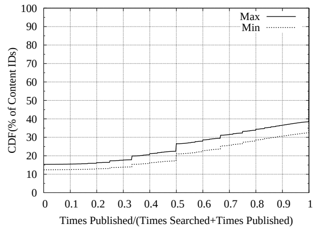
Message Rates: We begin by examining the rates at which Keywords and Files are published and searched. Figure 5 separately shows the min-max envelope for the Complementary Cumulative Distribution Functions (CCDFs) of Publish Keyword, Publish File, Search Keyword and Search File across all zones in log-log scale. Figure 5a demonstrates that 10% of published keyword content IDs have a request rate of more than 0.1 per minute, while 0.1% have a request rate of more than 30 per minute. For all four types of messages, the distribution is heavily skewed, with the vast majority of requests targeting a tiny fraction of the observed content IDs. This result is consistent with our earlier results from unstructured file-sharing systems [15].

Comparing the request rates across different message types reveals that the publish rates vary over a wider range than search rates. For example, some keywords have publish request rate greater than 100 requests per minute, while the highest observed Keyword Search request rate is less than 2 requests per minute. Presumably, this is in part because publish requests are automatically generated while search requests are triggered mainly by user action. Thus, the majority of request messages are generated by Kad nodes for the purpose of DHT protocol maintenance. The traffic generated by user activity accounts for only a small percentage of total traffic.

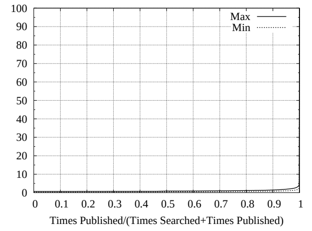
Publish requests are sent periodically. Given the observed request rate and the known re-publish interval (from eMule source code), we can estimate the number of Kad nodes in the system that publish a given file or keyword. The estimated number of Kad nodes that publish content is shown as a second x-axis on the top of Figures 5a and 5b. For example, some popular files are published at the rate of 30 requests per minute. Since each source sends a Publish File message once every four hours, this rate is equivalent to approximately 7,200 concurrent Kad nodes who possess a given file.

Relation Between Published & Searched Files: A DHT in essence provides a distributed mechanism for users who publish and users who search files to find each other. This section examines the balance between availability and demand for individual files.

We quantify the balance between the publish and search rate for content using the following ratio ($\frac{P}{S+P}$) for each content ID, where P is the rate of Publish Requests and S is the rate of Search Requests. In order to properly measure availability, we identify content publishers by using IP, port combination and discard duplicate publish requests. We do not discard any search requests because these requests show actual user demand. Figure 6a and 6b depict the CDF of ($\frac{P}{S+P}$) per content ID for files and keywords, respectively. Interestingly, Figure 6a reveals that 15% of files are searched but never published ($\frac{P}{S+P} = 0$). It is highly likely that these files recently became popular (*e.g., The Dark Knight*) and are not widely available in the system yet. On the other hand, 60% of files are published but never searched during our measurement window ($\frac{P}{S+P} = 1$). The availability of these files show that they were popular at one point but are not searched anymore. Figure 6b shows that 95% of keywords are published but never searched during our measurement window ($\frac{P}{S+P} = 1$). The significant imbalance between searched and publish rate for keywords indicates that only a small fraction of the keywords associated with files are actually used by users in searches.



(a) Publish vs Search Requests for Files



(b) Publish vs Search Requests for Keywords

Fig. 6: Relationship between Publish and Search Requests

VI. Related Work

There have been a few prior studies on empirical characterization of traffic in large scale P2P networks. Some of these studies have focused on unstructured P2P networks (*e.g.*, [3]) and thus are not directly applicable to DHTs. We are only aware of three prior empirical studies on traffic in deployed DHTs. Using 258 passively participating peers, Falkner *et al.* [2] examine core DHT characteristics such as the session length of peers, policies for including new peers in the DHT, and the availability of content in the Kademlia-based Azureus DHT. Qiao *et al.* [8] use four passively participating peers to collect traffic samples in Kademlia-based Overnet DHT. This study characterizes those DHT features that are related to finding files, such as the success and failure of queries, and the overhead of query traffic. They also evaluate the existence of keyword based hot-spots. Our study monitors two orders of magnitude more peers than [2] and four orders of magnitude more peers than [8].

Steiner *et al.* [10] developed a traffic monitoring tool for Kad, called Mistral, which captures traffic by placing a large number of peers into a zone. In the study they place around 65,000 monitoring peers into an 8-bit zone (which normally would contain only around 8,000 peers; see Table I). Additionally, Mistral routes any incoming traffic only to its own monitors, effectively shutting out most natural peers. The technique appears sound, but the study provides no measurements validating the accuracy and comprehensiveness of Mistral for capturing destination traffic.

Mistral and Montra both use tens of thousands of monitors. More precisely, Steiner *et al.* use twice as many monitors as our study. However, our monitors are more efficiently placed, allowing us to monitor four times as much address space with half the monitors. To monitor an 8-bit zone as they did, we would need only around 8,000 peers. In a nutshell, their approach is to add a large number of monitors to guarantee that the monitors will observe traffic. Our approach is to surgically add a smaller number of monitors to cover the same area.

Since Montra does not significantly disrupt the overlay structure of the existing peers, it provides richer data compared to Mistral. In addition to monitoring the quantity of traffic entering a zone, Montra can determine the final peer that received the message. The additional information could be used to study the distribution of load across peers or to measure how many packets fail to reach the theoretically correct final peer.[4]

VII. Conclusion & Future Work

In this paper, we argued that accurately capturing traffic in a DHT is challenging and then presented a scalable technique to achieve this goal without disrupting a target DHT. Our key idea is to keep the monitors minimally visible and lightweight. We implemented our proposed technique, validated its accuracy over Kad DHT, and then presented preliminary characterization of traffic in Kad. To continue this work, we plan to extend Montra to support other widely deployed DHTs, namely Azureus, Mojito, and conduct detailed characterization of traffic in these large scale DHTs.

[4]We plan to study these in future work.

References

[1] emule. Website. http://www.emule-project.net.

[2] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a million user DHT. In *Proceedings of IMC 2007*.

[3] A. S. Gish, Y. Shavitt, and T. Tankel. Geographical statistics and characteristics of P2P query strings. In *Proceedings of IPTPS'07*

[4] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of DHT routing geometry on resilience and proximity. In *Proceedings of SIGCOMM '03*.

[5] J. Li, J. Stribling, R. Morris, M. Kaashoek, and T. Gil. A performance vs. cost framework for evaluating DHT design tradeoffs under churn. In *Proceedings of INFOCOM 2005*.

[6] P. Maymounkov and D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of IPTPS '01*.

[7] G. Memon. Characterizing traffic in widely-deployed DHT. Technical Report CIS-TR-2009-01, 2008. http://ix.cs.uoregon.edu/∼gmemon/pubs/kad_tr_2008.pdf.

[8] Y. Qiao and F. E. Bustamante. Structured and unstructured overlays under the microscope: a measurement-based view of two P2P systems that people use. In *Proceedings of ATEC '06*.

[9] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01*.

[10] M. Steiner, W. Effelsberg, T. En Najjary, and E. W. Biersack. Load reduction in the Kad peer-to-peer system. In *DBISP2P '07*.

[11] M. Steiner, T. En-Najjary, and E. W. Biersack. A global view of Kad. In *Proceedings of IMC '07*.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM '01*.

[13] D. Stutzbach and R. Rejaie. Capturing Accurate Snapshots of the Gnutella Network. In *Proceedings of Global Internet Symposium '05*.

[14] D. Stutzbach and R. Rejaie. Improving lookup performance over a widely-deployed DHT. In *Proceedings of INFOCOM 2006*.

[15] D. Stutzbach, S. Zhao, and R. Rejaie. Characterizing Files in the Modern Gnutella Network. In *Multimedia Systems Journal 2007*.