Montra: A Large-Scale DHT Traffic Monitor

Ghulam Memon[a], Reza Rejaie[a], Yang Guo[c], Daniel Stutzbach[b]

[a]*University of Oregon*
[b]*Stutzbach Enterprises*
[c]*Bell Labs, Alcatel-Lucent*

Abstract

This paper presents a new technique, called Montra, for accurately capturing traffic in a widely deployed DHT. The basic idea is to make the traffic monitors minimally visible to participating peers to avoid disruption in the system. We describe how Montra leverages the required redundancy in published content and routing to minimize disruption on the system. Validations of Montra over two widely deployed DHTs, namely Kad and Azureus, show that it can accurately capture more than 90% of traffic destined to monitored peers. Furthermore, the lightweight nature of Montra allows it to monitor a large number of peers with a moderate amount of resources. We use Montra to characterize several aspects of traffic in our two target DHTs.

Keywords: Traffic Measurement; Peer-to-Peer, DHT; Kad; Azureus; Monitors; Characterization

1. Introduction

Distributed Hash Tables (DHTs) are increasingly incorporated into large-scale Peer-to-Peer (P2P) systems such as eMule [1] and Azureus [2]. As in any other large scale and distributed system, it is very valuable to capture a representative view of system traffic without disrupting its basic operations. Capturing system traffic enables one to identify design flaws, detect performance bottlenecks and determine how users use (or possibly abuse) the system in practice. For example, one may monitor traffic in a DHT to ensure that popular files are properly distributed across the ID space or to check whether botnets command and control messages are embedded into DHT messages [3].

Capturing a representative view of traffic for a large-scale DHT is nevertheless challenging. A common approach is to add instrumented peers that passively participate in the DHT and log exchanged messages [4, 5]. Deploying a small number of monitoring peers may not provide a representative and sufficiently detailed view of DHT traffic. Alternatively, inserting a large number of instrumented peers artificially increases the number of peers [6], which may disrupt the normal operation of the targeted DHT and may lead to biased measurement results.

This paper presents *Montra*, a novel approach for scalable and accurate monitoring of traffic in widely-deployed DHTs. The key idea behind Montra is to make traffic monitors minimally visible in a DHT without sacrificing their ability to capture traffic. Montra achieves its goal by leveraging the redundancy available in DHTs. DHTs require redundancy to ensure the availability of content. We believe that Montra can be extended to any DHT that incorporates common redundancy mechanisms regardless of their routing geometry. Montra minimizes the disruption to the system and significantly reduces the required resources for each monitor.

We first present the basic design of Montra and elaborate on DHT-specific issues. We implement Montra as a highly parallel, scalable, python-based client and validate it over both the Kad and Azureus DHTs. We show that our implementation of Montra can concurrently monitor approximately 32,000 Kad peers and 37,000 Azureus peers using a moderately configured PC (an Intel Core 2 Duo with 1 GB RAM), with more than 95% accuracy. Finally, we use Montra to characterize several aspects of traffic in Kad and Azureus DHTs.

We presented the idea of Minimally Visible Monitors (MVMs) in our earlier workshop paper [7]. This paper expands our earlier work by improving the design and validation and by applying Montra to two DHTs. Section 2

Email addresses: gmemon@cs.uoregon.edu (Ghulam Memon), reza@cs.uoregon.edu (Reza Rejaie), yang.guo@alcatel-lucent.com (Yang Guo), daniel@stutzbachenterprises.com (Daniel Stutzbach)

provides a brief background on our target DHTs. We describe the design of Montra and related issues in Section 3. Validation of Montra is described in Section 4. We present some characterization of traffic in our target DHTs using Montra in Section 5. Section 6 presents the related work and Section 7 concludes the paper.

2. Background

This section briefly describes the Kademlia-based [8] Kad and Azureus DHTs. Kad is employed by eMule [1] to support decentralized file-sharing. Azureus, on the other hand, uses its DHT to support decentralized tracking of BitTorrent swarms. Since both the DHTs differ in their usage, they differ in the content stored as well their operation. Both the DHTs are described below.

The **Kad** DHT stores two different types of content: the list of peers that host a file (*i.e.,* file records) and the keywords (*i.e.,* keyword records) associated with the file. As a result, *publishing* and *searching* of content always proceeds in two steps. When publishing content, first the keyword records are published and then the associated file records are published. When searching content, first the keyword records are searched and then the discovered file records are searched. Keyword records are two-tuples of: (*keyword hash, list of hashes of files matching the keyword*). File records are three-tuples of: (*file hash, file metadata, list of addresses of nodes hosting the file*). Hash values (*i.e.,* keyword hash and file hash) are used as IDs in the publish and search processes.

Both publish and search operations (for keyword records as well as file records) proceed in two phases: the *lookup phase* and the *storage/retrieval phase*. The lookup phase of *publish* returns ten peers whose IDs are closest to the publish ID. In the storage phase, the publishing node asks the peers (returned from the lookup phase) to add data to the records with the matching hash. The lookup phase of *search* finds 300 records whose file or keyword hashes match the search ID. Lookup operations must finish in 140 seconds, otherwise they terminate prematurely with partial results. In the retrieval phase, the searching node asks the peers for their records with the matching hash.

The **Azureus** DHT, unlike the Kad DHT, only stores the list of peers participating in a swarm, *i.e.,* peers downloading the same file. Azureus peers acquire content meta-data out-of-band through torrent files. Each swarm is represented by a 160-bit swarm ID, which is a SHA-1 hash of information contained in the swarm's torrent file. In Azureus, the store operation is called *put* and the retrieve operation is called *get*. The put operation, just like the publish operation in Kad, consists of a lookup phase and a storage phase. In the lookup phase, the request sender finds 20 peers closest to the swarm ID without time constraints. In the storage phase, the request sender sends its TCP port to the discovered peers and is added to the list of swarm participants. The get operation is different from the search operation in Kad, because the lookup and retrieval phases are merged. In a get request, the request sender initiates a lookup for the swarm ID. At each step of the lookup, it queries the discovered peer for swarm participants. The request sender terminates the get operation either after finding 8 other swarm participants or after continuing the lookup for 2 minutes.

In addition to differing in core operations, Kad and Azureus DHTs also differ in assigning IDs to peers. In Kad, each peer chooses its ID uniformly at random. On the other hand, in the Azureus DHT, a peer's ID is the 160-bit SHA-1 hash of the peer's globally visible IP address and port number. [1]

3. Montra Design

The observed traffic by peer *P* in any DHT can be divided into two categories:

- *Destination Traffic*: requests received by peer *P* if *P* has the closest ID to the target ID in the message using the DHT's closeness metric.
- *Routing Traffic*: requests received by peer *P* that must be routed towards another peer closer to the target ID.

The observed destination traffic at peer *P* depends on the popularity of all content IDs mapped to the peer. Montra aims at capturing destination traffic at a target peer; destination traffic represents user behavior, and it is more tractable. In this section, we discuss the basic design of Montra for capturing destination traffic at a single peer. Then, we explain the ability of Montra to minimize any impact on the target DHT, even as it scales to monitoring tens of thousands of peers. Finally, we describe the problem of accurately mapping each request to the proper peer and show how Montra addresses this problem.

[1] Azureus peers take lengths to ensure that even behind a NAT device they will maintain a consistent and unique port on the NAT's globally visible side. Further details are provided in our related technical report [9].

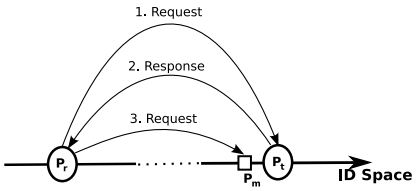


Figure 1: Message exchanges for capturing destination traffic

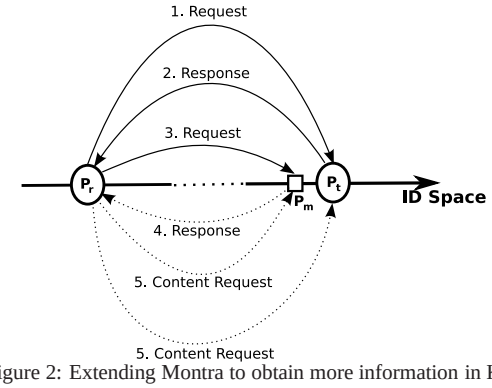


Figure 2: Extending Montra to obtain more information in Kad

The key to Montra's success is a real-world DHT's use of redundancy to ensure content availability. To gracefully cope with the dynamics of peer participation (or churn), an intuitive and commonly-adopted approach is to add redundancy to published content. Each piece of content is published at multiple peers (instead of a single peer) close to the target ID. This redundancy ensures the availability of content even if a fraction of peers leave the system. To effectively use the redundancy in published content, a peer that is searching for content tries to identify multiple peers close to the target ID. Adding redundancy to published content in a DHT implies that the mapping of content IDs to peers is not one-to-one. Since the destination of each request is not unique, the routing process at peer P should provide one (or a few) next hop peer(s) from its routing table that are closest to the target ID even if these next hop peer(s) are more distant than P from the target ID. Montra leverages this feature to accurately capture observed traffic destined to individual peers. More generally, Montra can be extended to capture destination traffic in any DHT that incorporates routing redundancy in a deterministic fashion. To illustrate this point, we describe how Montra is used in two DHTs with different types of redundancy later in this section.

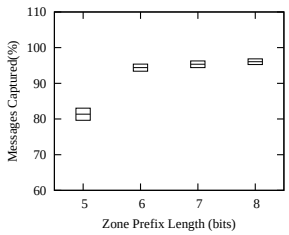
Capturing Traffic at a Single Peer: Suppose A_t is the portion of the DHT address space mapped to peer P_t (*i.e.,* P_t's ID is the closest to the identifiers in A_t). Montra attaches monitor P_m to the target peer P_t in two steps. First, it places the monitor next to the target peer by assigning it the identifier 1 XOR $ID(P_t)$. Second, it adds the monitor to the target peer's routing table by exchanging the proper messages. Such a close-by monitor for target peer P_t can capture P_t's destination traffic in the following two steps as shown in Figure 1:

1. When P_t receives a request from a request originator, 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.
2. 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.

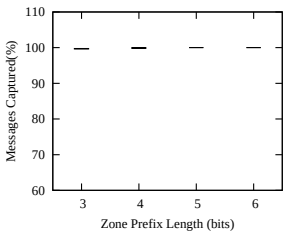
Minimizing Disruption in a DHT: In order to capture a more representative view of traffic in a DHT, one needs to monitor a large number of peers; if the monitors are visible it may affect traffic in the target DHT. Montra addresses this problem by deploying Minimally Visible Monitors (MVMs). The basic idea is to minimize the visibility of each monitor, P_m, by announcing its presence only to its target peer, P_t. To achieve this goal, P_m only responds to the messages issued by P_t and silently receives messages from all other peers. Other peers that may learn about an MVM from P_t treat P_m as a departed peer and do not add it to their routing tables permanently.

MVMs neither route traffic nor store content. Since MVMs are essentially invisible, placing a large number of MVMs does not disrupt or change the system. Furthermore, the lightweight nature of MVMs helps in deploying a large number of monitors while using minimal resources.

Identifying Destination Traffic: An MVM receives a small fraction of the requests for IDs that are relatively close to its associated target peer, P_t, but for which P_t is not the final destination. In other words, a fraction of traffic observed by an MVM is not destination traffic. A single MVM cannot distinguish between routing traffic and destination traffic because it does not know about the presence of a peer that is closer to the target ID. In order to accurately distinguish destination traffic from routing traffic for each monitored peer, we monitor all peers in a continuous region of the ID space called the *monitoring zone*. A zone is specified by x high order bits of the ID (or *prefix*) that is common among all IDs in that zone. For example, 0x$a4$ is a prefix for an 8-bit zone ($x = 8$). Any requests for an ID that has

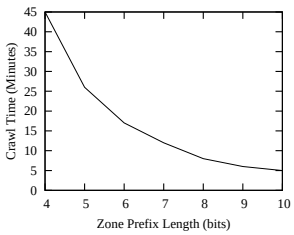


(a) Kad

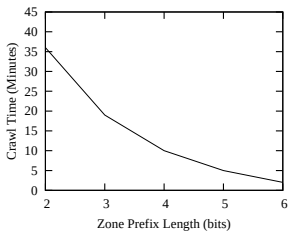


(b) Azureus

Figure 3: Montra Validations



(a) Kad



(b) Azureus

Figure 4: Crawl Times

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 the destination traffic for the zone. Although a request may be observed by multiple MVMs in a zone, Montra can establish the single closest peer during the post-processing phase. In practice, we first identify all active peers in a zone by back-to-back crawling of that zone using Cruiser [10]. Then, we monitor all the discovered active peers in that zone by attaching a separate MVM to each peer. The number of concurrently monitored peers depends on the available resources and implementation details. For example, our implementation of MVM can monitor 32,000 Kad peers using a moderately configured PC (Intel Core 2 Duo 2.2 GHz machine with 1 GB RAM) while capturing 50,000 packets per minute at peak load with less than 0.001% packet loss.

DHT Specific Issues: To properly interact with a particular DHT, Montra must be slightly customized based on the organization of information in different messages and the ID assignment strategy of the target DHT. Below, we explain these customizations for the Kad and Azureus DHTs.

In **Kad**, each lookup request carries the following information: *(i)* the type of request (publish or search), *(ii)* the requested content ID, and *(iii)* the ID of the destination peer. Montra, in its basic form, can capture this information. However, to capture additional information about each request in Kad, such as the classification of content (keyword or file) and file information (*e.g.*, size and type), MVMs respond to selected lookup messages from other peers in addition to responding to all the messages received from target peers. More specifically, when an MVM encounters a lookup message for a new content ID, it sends a response indicating its liveliness to the request originator, which in turn sends a second-phase message that carries the above meta-data as shown in Figure 2. Montra caches the content meta-data for the duration of measurement and the MVMs do not respond to any further lookup messages for this content ID. Responding to messages increases the visibility of an MVM, but the disruption is minimal since the MVM is visible only for peers reporting new content IDs and only briefly. Kad recently introduced a protection mechanism against sybil attacks; we have adapted Montra as explained in our related technical report [9].

In **Azureus**, the most challenging aspect is placing MVMs sufficiently close to the target peers. Peer IDs in Azureus are based on a combination of IP address and port number. To cope with this problem, we use a pool of 115 IP addresses from the 128.223.8/24 subnet of the Computer Science Department at the University of Oregon. Each IP address can use a UDP port from the range (1025, 65535). This combination of IP addresses and port numbers yields more than 7 million Azureus peer IDs, providing adequate flexibility to place an MVM near any of the 300,000 peers in Azureus.

Finally, we note that Montra cannot capture *get* requests from the Azureus DHT. A get request combines lookup and retrieval phase, and terminates after finding 8 other peers participating in the same swarm, as described in Section 2. Thus, with high probability, a get request may terminate before reaching the closest possible peer to the swarm ID, *i.e.*, the true destination. Montra cannot capture those requests that do not reach their true destination. In the rest of the paper, all references to Azureus requests refer to *put* requests.

Monitoring Overhead: The messages that are sent to individual MVMs increase the overall traffic associated with the target DHT. These extra messages can be viewed as monitoring overhead. We can estimate the percentage of this overhead as a function of DHT's relevant properties. Suppose that the average distance (*i.e.*, the number of rounds of routing) between two peers in a DHT is d and the level of redundancy in routing messages (*i.e.*, the number of routing messages sent in each round) is r. Therefore, the total number of routing messages to identify proper destinations during the lookup phase is on average $r*d$. Ideally, out of these messages only one is sent to an MVM. Thus, the monitoring overhead is $\frac{1}{r \cdot d}$. This simple model enables us to assess the basic overhead of Montra for a given DHT. For example, in Kad, the level of routing redundancy is $r = 3$, and the average number of hops between two peers is

d>3.2 [11]. This suggests that the basic overhead of Montra for Kad is 10.41%.

In practice, multiple MVMs close to the destination ID may receive same routing messages during the lookup phase. This in turn increases the overhead of Montra but the effect is DHT specific. We quantify this effect by examining the distribution of the number of duplicate messages among MVMs in a single 6-bit zone in Kad and repeat this analysis for ten 6-bit zone. Figure 5 presents all ten distribution of number of duplicate messages received by MVM in each zone by showing the minimum and maximum y values (for each x value). In essence, the derived CDF for each targeted zone is between the min and max lines in Figure 5. The min and max lines are very close to each other in Figure 5. This suggests that the distribution of the number of duplicate message per zone is very stable (*i.e.,* it is zone independent). Figure 5 shows that for 90% of lookup events, 3 or less MVMs receive routing messages, while the average number MVMs that receive routing messages is 2.17. This implies the actual overhead of Montra for Kad is 10.41% * 2.17 or 22.6%. The extended version of Montra for Kad has a slightly more overhead since it exchanges additional messages with other peers. However, since these additional messages are sent only once per content ID, this overhead is not significant. The examination of MVM logs revealed that this addition is only 13%.

In conclusion, the monitoring strategy in Montra increase the DHT traffic. The actual amount of increase depends on the details of the targeted DHT. It is important to note that this overhead is not likely to increase with the size of DHT since both the value of d and the number of duplicate messages are likely to increase. This suggests that the total overhead is likely to be independent of peer population. Furthermore, we are not aware of any other DHT traffic monitoring approaches that are capable of monitoring the traffic in large scale DHTs without disrupting the system or causing a significantly larger overhead than our approach.

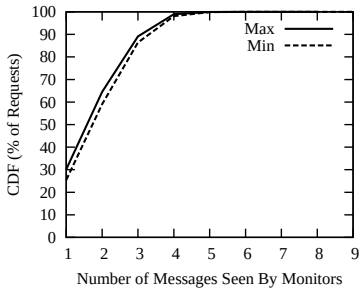


Figure 5: Routing Overhead in Kad

4. Validation of Montra

In this section, we quantify the percentage of traffic that Montra can capture. First, we deploy MVMs in a randomly selected zone in a DHT. Next, we run an instrumented peer with a randomly selected ID outside the monitored zone. The instrumented peer sends a large number of lookup requests towards random IDs within the monitored zone. Finally, we calculate the percentage of issued requests that are successfully captured by Montra. In addition to evaluating the accuracy, we also want to identify the largest possible zone that Montra can accurately monitor. Therefore, we explore accuracy as a function of zone size.

The validation results for the Kad and Azureus DHTs are shown in Figures 3a and 3b, respectively. Both figures show the zone prefix length on the x-axis and the percentage of successfully captured requests on the y-axis. The zone size doubles each time the zone prefix decreases by one bit. The top and bottom lines of each box reflect the 95% confidence interval and the middle line is the mean.

Figure 3a reveals that Montra captures almost 95% of destination traffic in Kad when the prefix length is 6 bits or larger. However, as the prefix length falls below 6 bits, Montra's accuracy drops significantly. For longer prefix lengths, Montra does not capture 100% traffic because *stale peers* exist in the routing table of alive peers. Stale peers are those departed peers whose departure has not been detected and are thus not removed. Stale peers prevent the lookup traffic from reaching the closest possible destination. If the traffic does not reach its true destination, then Montra cannot capture it. For larger zones, Montra's accuracy is affected significantly because of the large crawl time to discover all peers. As the prefix length decreases, it takes longer to crawl the ID space, as shown in Figure 4a. Longer crawling times lead to longer delays in discovering new peers and attaching monitors to them which in turn reduces accuracy. Based on these results, we monitor 6-bit zones.

Figure 3b indicates that Montra intercepts almost all Azureus destination traffic regardless of zone size. Unlike

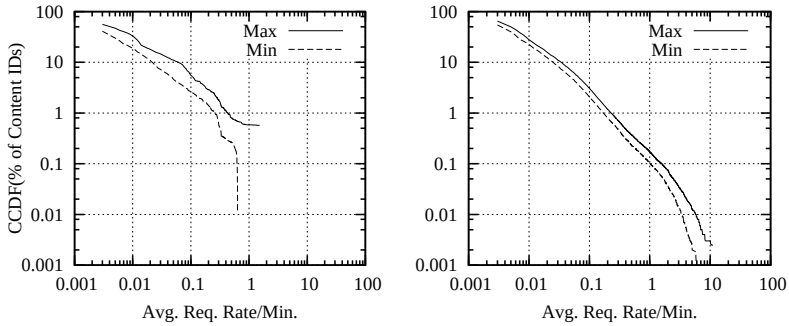


Figure 6: Kad Search Rates: a. Keywords, b. Files

Kad, neither the effect of stale peers nor crawl time is evident. Given the trend of crawl time shown in Figure 4, we believe the effect of crawl time would become visible if we evaluated Montra for larger Azureus zones. However, we were unable to test Montra over Azureus prefixes shorter than 3 bits due to the volume of incoming traffic (around 50 Mbps).

The effect of churn is absent in Azureus because of the lack of timeouts. Azureus peers allow the requests to reach the true destination, and thus route around any stale entries in the path. If a request reaches its true destination, then Montra can capture it. In summary, these results reveal that monitoring 3-bit zones in Azureus is the most appropriate given our bandwidth limitations.

5. Characterization of Traffic

The granularity of Montra (described in Section 4) allows us to monitor $\frac{1}{64}$ of Kad (32,000 peers) and $\frac{1}{8}$ of Azureus (37,000 peers) ID space at any point of time. In a series of observations, we monitored each of the 64 Kad zones and 8 Azureus zones. Each set of observations is 6 hours in length. To ensure robustness to any time-of-day effects, the observations began at different times of the day (3pm, 9pm, 3am, or 9am) chosen at random.[2] The Kad dataset covers May 2007 through October 2008 while the Azureus dataset covers June 2009. In this section, we leverage these datasets to examine several characteristics of traffic in the Kad and Azureus DHTs

Plotting the Cumulative Distribution Function (CDF), or its complement (CCDF) for every dataset on one graph to compare them is impractical due to the large number of datasets. In order to present the data succinctly, we plot the minimum and maximum y values (for each x value) across all distributions, resulting in two lines per graph that succinctly capture the data. The CDF and CCDF for each dataset falls somewhere between the minimum and maximum lines. In general, we found that the minimum and maximum lines are relatively close together, demonstrating little variation between zones.

5.1. Message Rates

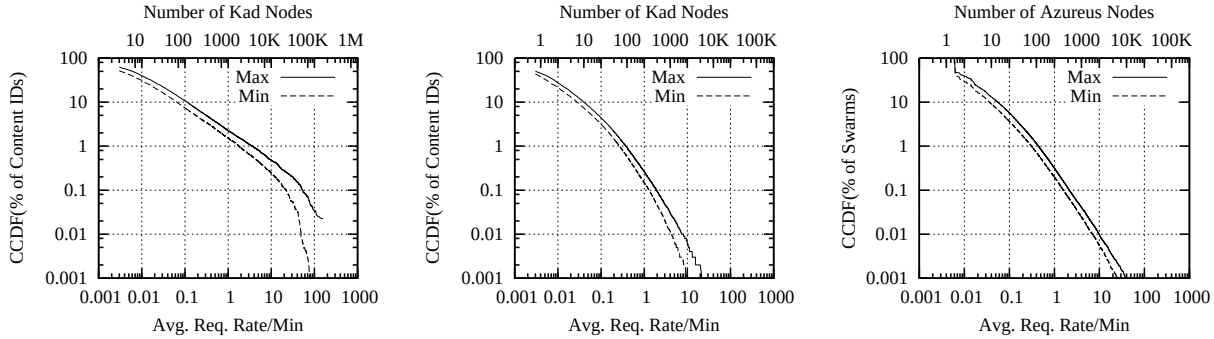
To begin our study, we examine the rates at which different content IDs are requested. As discussed in Section 2, there are two basic types of messages monitored by Montra: *(i)* search and *(ii)* publish. Due to the technical limitations (see Section 3), we are unable to capture search requests for Azureus.

Figure 7 shows the distributions of publish requests over the set of IDs, as CCDFs in log-log scale. Kad and Azureus both send automatic publish messages at 4-hour intervals, allowing us to estimate the number of peers publishing each piece of content. The number of peers are presented as a second *x*-axis along the top of each graph.[3] Figure 7a shows that 10% of published keyword IDs have a request rate of more than 0.1 per minute, while 0.1% have a rate of more than 30 per minute. For all three types, the distribution is heavily skewed, with the vast majority of messages targeting a tiny fraction of the observed content IDs. In other words, a tiny fraction of content is widely popular, while the majority of content is found on only a few peers. This heavily skewed distribution is consistent with observations of other file-sharing systems [12].

Figure 6 shows the distribution of search requests over the set of IDs, as CCDFs in log-log scale. Comparison of Figures 7 and 6 shows the rate of publish requests is much higher than the rate of search requests. For example, some

[2]All times in this paper are in Pacific Time.

[3]Departure and arrival of peers will cause some error in these values, but they are a useful first-order approximation.



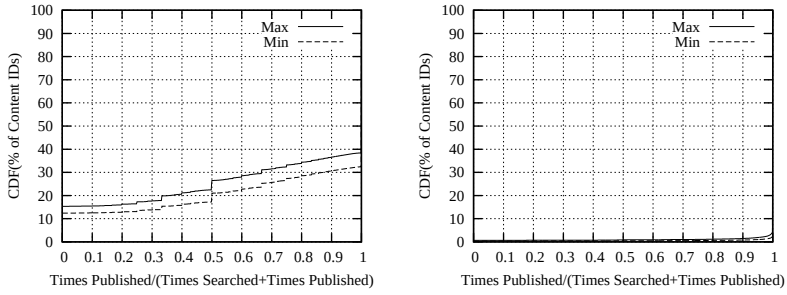
(a) Kad Publish Keyword (b) Kad Publish File (c) Azureus Publish File

Figure 7: Publish Request Rates

keywords have a publish request rate greater than 100 requests per minute, while the highest observed keyword search request rate is less than 2 requests per minute. Publish requests are automatically generated, while search requests are manually generated. Thus, the majority of request messages are generated by Kad nodes for the purpose of protocol maintenance. The traffic generated by direct user activity only accounts for a small percentage of total traffic.

Relation Between Published & Searched Content A DHT, in essence, provides a distributed rendezvous mechanism for content contributors (publishers) and consumers (searchers). Publish requests for a given content demonstrate its availability, whereas search requests represent the demand. This section examines the balance between availability and demand for individual content. If a file is excessively published but never searched, then that file has lost its popularity and is now easily available in the system. On the other hand, if a file is searched by a large population but is never published then that file has gained popularity recently and is not available in the system yet.

We explore this balance using the formula $(\frac{P}{S+P})$ for each content ID, where P is No. of Publish Requests and S is No. of Search Requests observed during a measurement window. In order to find true availability of a given content we discard duplicate publish requests from the same publisher. We use IP, port combination to identify content publishers. Duplicate search requests, unlike duplicate publish requests, show true demand. Therefore, we do not discard search requests. Figure 8a and 8b show the CDF of $(\frac{P}{S+P})$ per content ID for files and keywords, respectively. Interestingly, Figure 8a reveals that 15% of files are searched but never published $(\frac{P}{S+P} = 0)$. Most likely, these files became recently popular and thus are not widely-available in the system. Figure 8a also show that 60% of files are published but never searched during our measurement window $(\frac{P}{S+P} = 1)$. These files are very well-spread in the system and therefore are rarely searched. Figure 8b shows that 95% of keywords are published but never searched during our measurement window $(\frac{P}{S+P} = 1)$. This shows that very small fraction of keywords is actually used for search and lot of resources are wasted on publishing additional keywords.

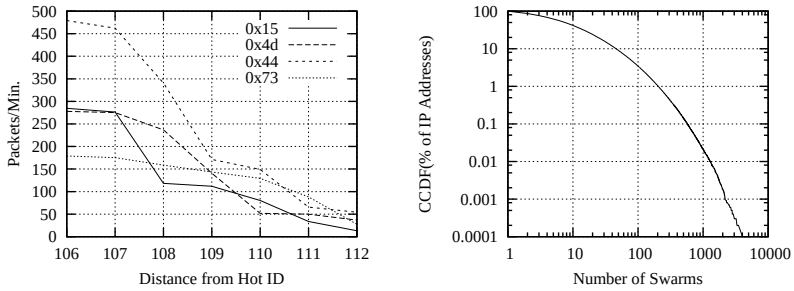


(a) Publish vs Search Requests for Files (b) Publish vs Search Requests for Keywords

Figure 8: Relation Between Published & Searched Content

Radius of Hot IDs: We define hot IDs as extremely popular content IDs. The skewed distributions shown in Figure 7 and 6 demonstrate that hot IDs exist. The largest impact of a given hot ID is observed by its destination peer, which will receive several orders of magnitude more traffic than a typical peer. However, peers close to the destination are also affected because they route the traffic. We define the radius of a hot ID region as the distance between

the destination peer and the farthest peer that observes a dramatically increased request rate. In this subsection we empirically characterize the radius of hot ID regions in Kad.



(a) Impact Zone of Hot ID (b) Swarms Per Host

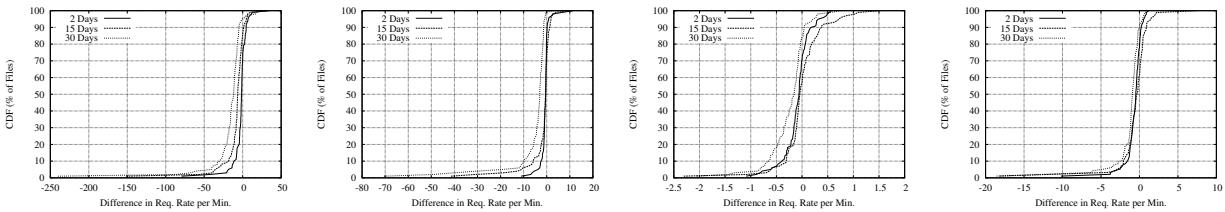
Figure 9: a. Radius of Hot IDs, b. Swarm Participation

We examine the radii of the four most popularly published keywords from four different zones. We choose publish keyword requests since they are the largest source of DHT traffic in Kad. We measure the radius by placing 7 instrumented Kad clients at varying distances from each hot ID. An instrumented client is a regular Kad client modified to log all incoming request messages. The key idea is to place enough instrumented clients at varying distances such that we can observe the variation in traffic rate. We place the first instrumented client at a distance of 106 bits. This client has 22 high order bits in common with the hot ID (106 + 22 = 128), which in most cases is sufficient to make it the closest peer to the ID. We place the remaining 6 clients at increasing distances of 107 bits, 108 bits and so on. Figure 9a shows the radii for four measured hot IDs. The graph legend shows the 6-bit zone prefixes from which the hot IDs are chosen. The figure shows that the traffic declines steeply as a peer moves further from the hot ID, from 250–500 messages per minute down to 5–50 messages/minute. The sharp decrease is because of the exponential increase in number of peers that can route the traffic towards the destination.

We were not able to conduct this analysis for the Azureus DHT because of the lack of peer IDs that can be varied bit by bit. The pool of 7 million peer IDs, mentioned in Section 3, is enough to monitor the destination traffic of individual peers. However, it is an extremely small percentage of total Azureus peer IDs. Thus, we cannot select the peer IDs with enough precision.

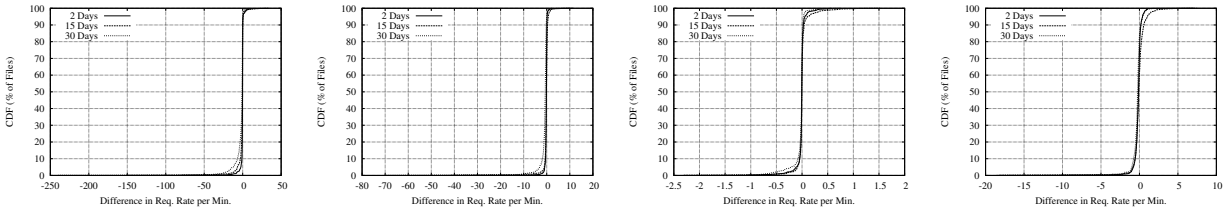
Swarm Participation per Peer: Next, we explore the number of swarms in which each Azureus peer participates. Figure 9b shows the CCDF of the number of swarms per peer on a log-log scale. The *x*-axis shows the number of swarms and the *y*-axis shows the percentage of peers. We identify a peer using its IP address, instead of IP:port combination, so that anomalous peers using a large number of ports cannot bias the results. The data shows that most peers participate in small number of swarms swarms, while a tiny percentage of peers participate in a large number of swarms. For example, 50% of peers participate in six or fewer swarms, while 0.001% of peers participate in 1,000 or more swarms.

This analysis cannot be performed for Kad because NAT peers use “buddies” for publishing files. If several Kad peers publishing the same file choose the same buddy then the population of the file will be underestimated. On the other hand, Azureus peers avoid the effect of NAT, as mentioned in Section 2.



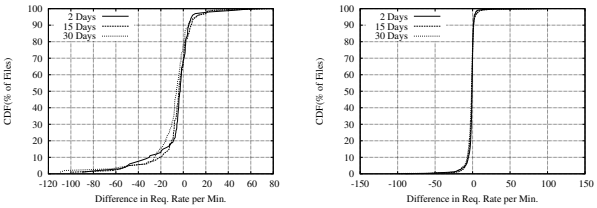
(a) Publish Keyword (b) Publish File (c) Search Keyword (d) Search File

Figure 10: Change in popularity for top 100 Kad files



(a) Publish Keyword (b) Publish File (c) Search Keyword (d) Search File

Figure 11: Change in popularity for top 1000 Kad files



(a) Top 100 (b) Top 1000

Figure 12: Change in popularity for top 100 and 1000 Azureus files

5.2. Evolution of Availability & Popularity of Individual Files

In order to understand how content availability and popularity evolves over time, we study the temporal property of content request rates for the Kad and Azureus DHTs. For both the DHTs we monitor a given zone for 6 hours every 2 days for 30 days. We refer to each 6-hour dataset as a *snapshot*. For each snapshot, we can leverage the publish rate of individual keywords or files to determine their availability. Similarly, we can use the rate of search keyword or search file messages to determine their popularity (*i.e.*, demand) at any point of time. If we consider the first snapshot as a reference, comparing later snapshots with the reference reveals any change in the availability or popularity of available files over time as well as any new files that might have become available.

Figure 10a and10b show the distribution of change in availability of individual keywords and files while Figure 10c and Figure 10d show the change in popularity of top-100 keywords and files, respectively. Each figure depicts the corresponding changes over three intervals 2, 15 and 30 days. The negative values on x-axis show a decrease in availability and vice versa. Figure 10a and Figure 10b reveal that the availability of content gradually decreases over time. For example, after 2, 15 and 30 days 40%, 80% and 95% of keywords become less available, respectively. The publish rate for a few keywords drops by approximately 250 requests per minute after 1 month that suggests these keywords lose roughly 300,000 publishing peers. A majority of keywords lose 10 to 50 requests per minute (14,400 to 72,000 publishers) after 1 month. The evolution of popularity for top-100 keywords and files in Figure 10c and 10d exhibit a similar decreasing trend with time but with the varied pace.

We have conducted the same analysis for Azureus. Figure 12a and 12b show the change in the availability of top 100 and top 1000 files in Azureus. Similar to Kad, these results exhibit a gradual decline in the availability of files in Azureus. However, the availability of files in Azureus drops slower than Kad. More specifically, after 30 days the availability of almost all files in Kad drops while such a drop is seen for only 80% of files in Azureus. Furthermore, the examination of top 1000 files in Azureus confirms that top 100 files exhibit a significantly larger change in their availability than in Kad.

To examine the evolution of content availability and popularity over a larger population of files, Figure 11 presents the same analysis (as in Figure 10) for top 1000 files and keywords. This figure show very little change in availability and popularity of top-1000 files and keywords even after 30 days. This suggests that most of the fluctuation in availability and popularity is associated with the top 100 files and keywords.

5.3. Churn in Aggregate Available Content

In this subsection, we focus on the change in aggregate available content in the DHT over time. We compare any snapshot with the first one (as a reference) to determine what fraction of files have been added or removed from the system. The results for Kad and Azureus DHTs are shown in Figure 13, normalized by the total number of files in the reference.

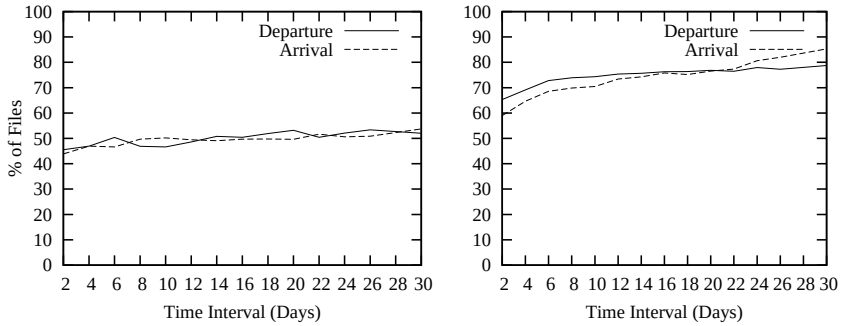
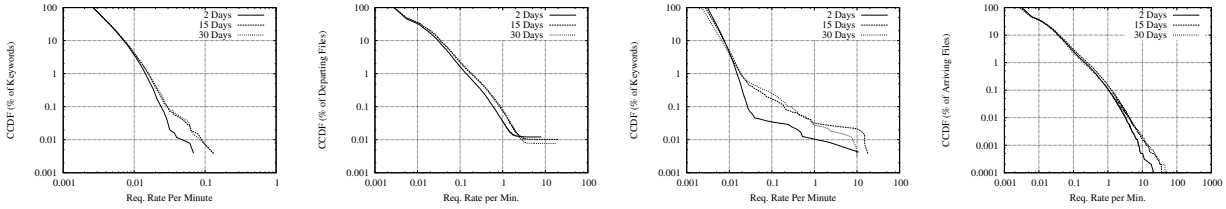


Figure 13: Arr. and Dep. of Files: a. Kad, b. Azureus

Two qualities are notable about these results. First, in each system the number of newly arrived files balances the number of departing files. Thus, the overall number of files available remains roughly the same even though the selection of files varies dramatically. Second, almost all files that were still present after 2 days were also present after 30 days (resulting in the nearly horizontal lines). Rather than seeing a gradual decay of the original set of files, a large number of files departed during the first 2 days but few files departed thereafter. Presumably, this effect is caused by the skewed number of peers sharing each file, shown in Figures 7b and 7c. The many files shared by just a few peers become unavailable quickly, while the files shared by many peers remain available.

Figure 14 depicts CCDF of the publish rate for files that depart or arrive in each DHT in log-log scale within a particular window of time to provide a micro-level view of file churn. Figure 14a and 14b show that the publish rate for a majority of departing files is low over different window of time. This implies that files even with extremely low availability can remain in the system for more than 30 days. For example 5-10% of departing published files have the original request rate of 0.01 or less. This result explains why highly available content do not suddenly disappear from the DHT. Figure 14c and 14d indicate that the distribution of publish rate for newly arriving files is very skewed over all time intervals in both the DHTs. A majority of files have a very low publish rate but a small fraction of files reach a very high availability even within a couple of days. For example, roughly 1% of newly arriving files in Kad reach the highest publish rate of 10 requests per minutes (as shown in Figure 7b) even within 2 days.



(a) Departing Kad Files (b) Departing Azureus Files (c) Arriving Kad File (d) Arriving Azureus Files

Figure 14: Dist. of Publish Rate for Departing and Arriving Files in Kad and Azureus

	Publish Files	Publish Keywords	Search Files	Search Keywords
Italy	22.687	25.343	12.836	17.637
Spain	21.306	21.025	10.594	10.522
France	12.502	13.532	9.838	9.247
Brazil	5.586	7.115	4.591	3.838
China	4.449	3.126	30.105	19.699
Other	33.470	29.859	32.036	39.057

(a) Kad

	Publish Files
US	25.52
France	8.59
Canada	8.33
Great Britain	6.39
Spain	5.11
Other	46.06

(b) Azureus

Table 1: Percentage of DHT Traffic Originating from Top 5 Countries.

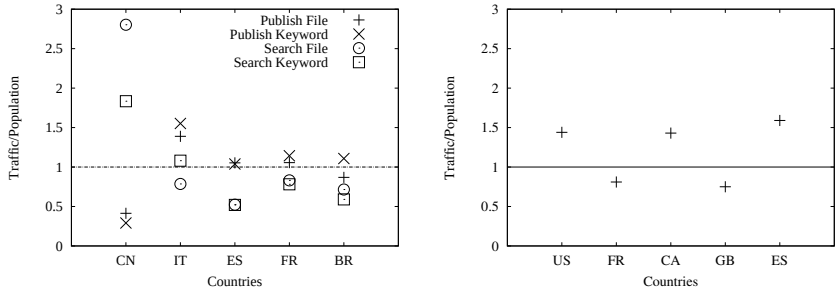
5.4. Geographical Characteristics

In this subsection, we briefly explore the geographical characteristics of Kad and Azureus traffic. For this analysis, we use 24-hour traffic traces in order to avoid the time-of-day effect. In order to understand the geographical characteristics of captured traffic, we classify the requests by type and country of origin. The results are summarized in Table 1 for Kad and Azureus. Table 1a reveals that Italy is the biggest contributor of content (files as well as keywords) to Kad while China is the biggest consumer of content. Almost one third of request (of different type) are originating from other countries. Table 1b presents the breakdown of Azureus traffic across different countries which looks different from Kad. US is the main contributor of traffic in Azureus. Spain and France are among the top 5 contributors but with a much smaller share than Kad. Furthermore, the contribution of traffic across different countries is less skewed in Azureus since almost half of the traffic is originating from other countries compare to 66% in Kad.

Ideally, when a peer downloads a file, it automatically publishes the file back. Thus, one expects that the contribution and consumption of each country is relatively balanced and it is proportional with the population of peers in that country. However, our results in Table 1 do not support this hypothesis. To explore this issue more closely, we quantify the normalized contribution and consumption with the following ratio:

$$\frac{\%\text{ of Total Traffic Generated by the Country}}{\%\text{ of Peers Located in the Country}}$$

Figure 15a depicts the normalized contribution and consumption of top five countries. The normalized value of one indicates the perfect balance between traffic and population for each country. Figure 15a reveals that Italy, Spain, France and Brazil publish proportionally more content and consume less content than fraction of their population whereas China is the exact opposite. Closer examination of our results led to two reasons for the excess consumption by Chinese peers. First, Figure 15a revealed that the contribution-deficit by Chinese peers is very close to the contribution surplus by Italian. This must be due to the fact that Chinese users behind NAT boxes use Italian peers as buddies. Thus in reality, neither Chinese peers are under-contributing nor Italian peers are over-contributing. Second, we identified a single peer in China who searches significantly more often than normal users. When we eliminate this particular peer from the analysis, then the normalized consumption and contribution of China in Figure 15a becomes very close to 1. Figure 15b depicts the normalized consumption and contribution of top 5 countries in Azureus. . This figure shows that US, Canada and Spain contribute more while France and Great Britain contribute less content than their fair share. However, unlike Kad, lower contribution in Azureus is not an anomaly. Rather it simply shows while lot of peers from France and Great Britain are connected to the DHT, not all are involved in file exchange. Content under-contribution is an anomaly only when it is complimented by content over-consumption.



(a) Kad (b) Azureus

Figure 15: Traffic to Population Ratios

6. Related Work

Characterization of P2P systems via measurement study has attracted much attention. Earlier measurement studies focuses on unstructured P2P systems such as Gnutella, Napster, BitTorrent, etc. [13, 14, 15, 16, 17, 18]. Among them, [13, 14, 15] conduct application-level studies by deploying instrumented peers and characterize the traffic observed by those peers, while [16, 17, 18] examine the traffic collected at a router inside the network. There are pros and cons associated with individual methodologies. In this paper, we explore application-level measurement approach for its flexibility.

DHTs were deployed in the past several years, and to the best of our knowledge there are three prior empirical studies on deployed DHT traffic. Falkner et al. [4] use 258 passively participating peers to examine core DHT characteristics such as the peer session length, new peer joining policies, and content availability in the Azureus DHT. Qiao et al. [5] use four passively participating peers to study query success/failure rates and overhead in the Overnet DHT. Compared to the two aforementioned studies, Montra is able to monitor two to four orders of magnitude more peers at a time. Steiner et al. [6] developed a large-scale traffic monitoring tool, called Mistral, which places many monitoring peers in Kad. Mistral routes any incoming traffic only to its own monitors, effectively shutting out most normal peers. In a nutshell, Mistral adds 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. As a result, Montra is more efficient and less invasive than Mistral.

Other DHT measurement studies have looked at non-traffic aspects of DHTs. For example, Steiner et al. developed Blizzard [19], a Kad crawler. The data collected from Blizzard is used to analyze peer properties such as population, session time, and IP addresses. Our earlier study [11] explored lookup performance in Kad by studying several variations of the Kademlia lookup algorithm over the real Kad network, and introduced Cruiser [10], a Kad variation of P2P crawler.

7. Conclusion

This paper presented Montra, a new technique for large-scale monitoring of a DHT without disrupting the system. Section 3 described the general technique of using lightweight monitors with minimum visibility, along with specific implementation issues encountered using Montra on the Kad and Azureus DHTs. Montra is highly scalable, allowing a significant fraction of a large DHT to be monitored with only modest resources.

Section 4 presented experiments evaluating the comprehensiveness of Montra, demonstrating roughly 95% packet capture in Kad and virtually 100% packet capture in Azureus. Finally, Section 5 characterizes several aspects of traffic in Kad and Azureus using data captured with Montra. In particular, the data shows that a large majority of the DHT traffic is spent maintaining information on rarely-queried items.

References

[1] emule, Website, 2011. `http://www.emule-project.net`.

[2] Azureus, Website, 2011. `http://azureus.sourceforge.net`.

[3] G. Memon, J. Li, R. Rejaie, Tsunami: A Resilient, Parasitic Botnet, Technical Report CIS-TR-2011-05, Eugene, OR, USA, 2011.

[4] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, T. Anderson, in: IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM, New York, NY, USA, 2007, pp. 129–134.

[5] Y. Qiao, F. E. Bustamante, in: ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference, USENIX Association, Berkeley, CA, USA, 2006, pp. 31–31.

[6] M. Steiner, W. Effelsberg, T. En Najjary, E. W. Biersack, in: DBISP2P 2007, 5th International Workshop on Databases, Information Systems and Peer-to-Peer Computing, September, 24, 2007, Vienna, Austria.

[7] G. Memon, R. Rejaie, Y. Guo, D. Stutzbach, in: IPTPS'09, 8th International Workshop on Peer-to-Peer Systems, April 21, 2009, Boston, USA.

[8] P. Maymounkov, D. Mazières, in: IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag, London, UK, 2002, pp. 53–65.

[9] G. Memon, D. Stutzbach, Y. Guo, R. Rejaie, Montra: A Generic Technique for Monitoring DHT Traffic, Technical Report CIS-TR 2009-03, Eugene, OR, USA, 2009.

[10] D. Stutzbach, R. Rejaie, in: Global Internet Symposium, Miami, FL, pp. 127–132.

[11] D. Stutzbach, R. Rejaie, INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings (2006) 1–12.

[12] D. Stutzbach, S. Zhao, R. Rejaie, Multimedia Systems Journal 1 (2007) 35–50.

[13] A. Klemm, C. Lindemann, M. K. Vernon, O. P. Waldhorst, in: IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, ACM, New York, NY, USA, 2004, pp. 55–67.

[14] B. Krishnamurthy, J. Wang, Y. Xie, in: IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, ACM, New York, NY, USA, 2001, pp. 105–109.

[15] M. Ripeanu, A. Iamnitchi, I. Foster, IEEE Internet Computing 6 (2002) 50–57.

[16] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, J. Zahorjan, in: SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM, New York, NY, USA, 2003, pp. 314–329.

[17] S. Sen, J. Wang, IEEE/ACM Trans. Netw. 12 (2004) 219–232.

[18] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, M. Faloutsos, in: Proceedings of the GLOBECOM 2004 Conference, IEEE Computer Society Press, Dallas, Texas, 2004.

[19] M. Steiner, T. En-Najjary, E. W. Biersack, in: IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM, New York, NY, USA, 2007, pp. 117–122.