Kademlia protocol statistics

based on: "A global view of KAD",by M.Steiner, T. En-Najjary and E.W. Biersack
&&
"Profiling a million user DHT", by J. Falkner, M. Piatek, J.P.John, A.Krishnamurthy, T.Anderson

Elena Digor

March 16, 2009

DHTs analysis have shown following results

- short session times on average, but heavy tailed
- lookups are robust, because of long-lived nodes that shoulder a relatively large amount of traffic
- "closest node" for a given key is partially inconsistent in both long term and short term, motivating redundant storage and refresh

Kademlia challenges:

- **Churn:** nodes arrive and depart rapidly, making the routing table entries stale;
- **Consistency:** routing tables are not only stale but also inconsistent. Reason: use of replication
- **Failures:** message transport is done via UDP. Message and node-failures are detected by application level. (Message failure: 20 sec timeout; node failure: no response to 2 back-to-back messages)

Experiment characteristics:

- *analyzed protocol*: Azureus DHT (based on Kademlia) - an implementation of BitTorrent protocol
- *vantage points*: 250 PlanetLab & 8 UW
- *time interval*: February-May, 2007

Why Azureus?

- Widely deployed, and is in use by more than 1 million users
- Azureus - maintains a DHT based on Kademlia, to address a scalability bottleneck, present in BitTorrent
- parallel lookups based on each nodes buckets is limited to 10

Session lengths determines the rate at which routing table entries need to be probed for freshness.

- Method used: random sampling method
- Number of tested nodes: 300,000 (gathered by get at random)
- sampling rate: 2.5 minutes
- experiment length: 48 hours

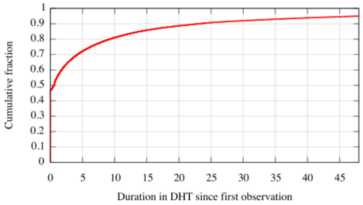


Figure 1: Freshness of DHT routing table entries and persistence for 48 hours after first observation.

Challenges:

- cannot predict a node's longevity;
- risk of polluting routing tables with short-lived nodes

Solution:

- nodes are added instantly, but routing tables are updated slowly

Testing characteristics:

- 125 Planet Lab vantage points have joined simultaneously the Azureus client.
- experiment length: 2 days

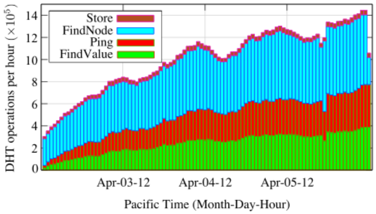
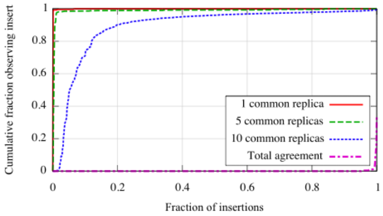


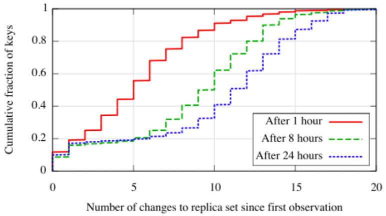
Figure 2: Two day trace of DHT messages received by 125 vantage points on PlanetLab. Bars give the absolute number of each measure type over a one hour period.

- overhead: FindNode and Ping messages comprise 81% of

- **short term consistency:** perform (total of 931) *puts* of random keys into DHT, measuring the visibility of each insertion from 250 PlanetLab vantage points
  - different vantage points observe different sets of replicas
- **long term consistency:** reinsert periodic data into the DHT
  - 125 PlanetLab vantage points.
  - Each select a random key and a 1,8 or 24 hour interval



(a) Short-term consistency



(b) Evolution of replica set

Figure 3: Profiling routing table consistency. ***Left:*** **The cumulative fraction of vantage points observing a DHT put immediately after insertion.** ***Right:*** **Evolution of replication set over time from the perspective of a single vantage point.**

- conclusion: periodic insertions can be performed at the granularity of hours with little impact on data persistence

Measurements done over 45 million messages sent during the trace experiment from PlanetLab and UW vantage points

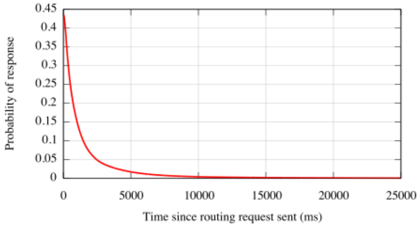


Figure 4: The probability of receiving a reply to a DHT message as a function of time after sending.

- Observed that: response probability distribution is remarkably stable over both short and long time scales and across all vantage points

Performance is controlled by the parameters that impact routing delay:

- lookup parallelism;
- message timeouts;
- rate-limits on message processing

Original DHT characterized by:

- hard bounds on outstanding messages
- rate-limits on message processing

Proposed solution:

- Remove time-bounds control, as DHT message fit in a single UDP packet (20 sec message timeout); (no need for complicated algorithms;)
- 6 parallel lookups
  - improved end-to-end lookup time from 127 sec to 13 sec.
  - total number of nodes contacted increased from 52 to 150
  - by eliminating timeouts, the number of contacted nodes decreased by an order of magnitude.

- KAD - a DHT based on Kademlia, part of eDonkey2000
- experiment duration: about 6 months
- idea: "crawl" KAD - 1 crawl every 5 minutes

- KAD ID - 128 bit long; randomly generated once when the client application is started for the first time
- routing: based on prefix matching (same as KADEMLIA)
- publishing: 2 different keys:
  - **source key:** - identifies content of a file (hashes content of a file); republished every 5 hours;
  - **keyword key:** -classified the content of a file (hashes the tokens of the name of a file); republished every 24 hours
- publishing done on 10 different peers; KAD ID should agree at least in the first 8-bits with the key (tolerance zone)

Developed a crawler that:

- logs per each peer the time of the crawl, IP address of the peer, and its KAD ID
- use only one machine; keep all relevant information in main memory
- runs on the local disk, knowing several hundred contacts to start with
- uses 2 threads: 1 -sends out asynchronous requests; 2 - stores results
- speed limited by the available bandwidth (100Mbit/sec bi-directional)
- 2 Blizzards: University of Manheim, Germany and Institut Eurecom, France

- full crawl - performed in 8 minutes
- first million different peers identified in 10 seconds; 2nd million - 50 sec
- about 3GBytes of inbound and outbound traffic each.
- experiment will be based on zone crawl (8-bit zone) - less than 2.5 sec

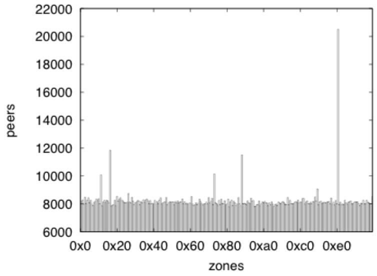


Figure 1: The distribution of the peers over the hash space. The 256 8-bit zones on the x-axis go from 0x00 to 0xff.

- modified KAD clients are typical to the same single country.
- users with same KAD ID cannot see one another and cannot download from one another

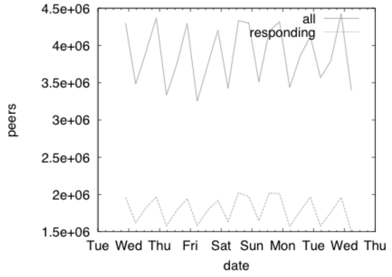


Figure 2: The number of kad peers available in entire kad ID space depending on the time of day.

- peers behinds NATs or firewalls use KAD to publish information about the content they share, but do not participate in storing published information (no contribution to the operation of KAD)

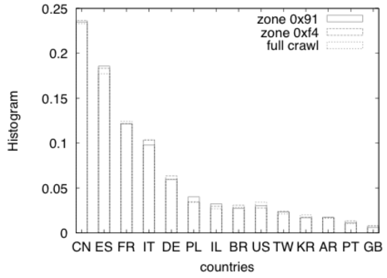


Figure 3: Histogram of geographic distribution of peers seen on 2006/08/30.

- Maxmind database to resolve IP address to countries and ISPs

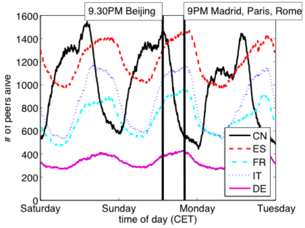


Figure 4: Peers online according to country of origin.

	Total	China	Europe	Rest
Different KAD IDs	400,278	231,924	59,520	108,834
Different IP addresses	3,228,890	875,241	1,060,848	1,292,801
KAD IDs seen for a single session	174,318	131,469	11,644	31,205
KAD IDs with LT $\leq$ 1 day	242,487	183,838	15,514	43,135
KAD IDs seen for the first time on				
- 1st crawl	5,670	455	2,879	2,336
- 1st day	18,549	4,535	6,686	7,328
- 60th day	1,893	1,083	259	551
KAD IDs seen for the first time on 1st day				
- with LT $\leq$ 1 day	2,407	1,568	286	553
- 1 day $<$ LT $\leq$ 1 week	1,368	497	393	478
- 1 week $<$ LT $\leq$ 1 month	2,735	791	944	1,000
- LT $>$ 1 month	12,039	1,679	5,063	5,297
- LT $>$ 3 months	8,423	936	3,679	3,808

KAD ID aliasing:

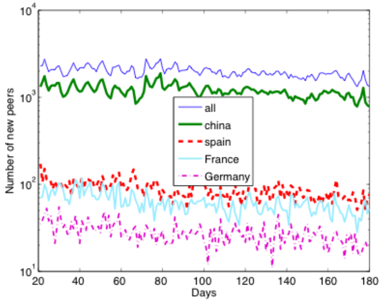


Figure 7: New kad IDs according to country of origin.

Look up for peers with static IP:

- **pivot set:** 160,641 peers with same IP and KAD ID both on 20th March, 2007 and 30th March, 2007.
- take logs from April 1st, 2007 – see how many peers have changed their KAD ID

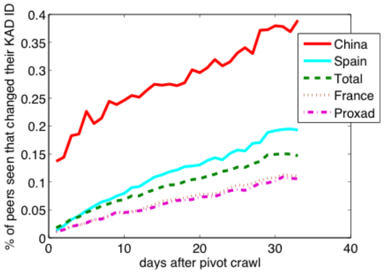


Figure 8: The fraction of peers in pivot set that changed their kad ID at least once.

- A **peer** is an instance of KAD identified by a fixed KAD ID
- An **end-user** is a physical person that launches a peer to participate in KAD. Same end-user can, at different times, participate via different KAD peers.

We cannot characterize the lifetime of end-users, as compared to lifetime of peers!

- Sessions vary from minutes to months;
- sessions have long tails (lasting to as long as 78 days)
- nodes are uniformly distributed
- lookups are generally robust

Thank you for your attention!

Questions?:)