Queen: Estimating Packet Loss Rate between Arbitrary Internet Hosts

Y. Angela Wang[1], Cheng Huang[2], Jin Li[2], and Keith W. Ross[1]

[1] Polytechnic Institute of NYU, Brooklyn, NY 11201, USA
[2] Microsoft Research, Redmond, WA 98052, USA

Abstract. Estimate of packet-loss rates between arbitrary Internet hosts is critical for many large-scale distributed applications, including overlay routing, P2P media streaming, VoIP, and edge-server location in CDNs. iPlane has been recently proposed to estimate delay, packet-loss rates, and bandwidth between arbitrary hosts [1]. To our knowledge, iPlane is the only published technique for estimating loss rates between arbitrary Internet hosts. In this paper, we present Queen, a new methodology for estimating packet-loss rates between arbitrary hosts. Queen, extending the King [2] methodology for estimating delay, takes advantage of the open recursive DNS name servers. Queen requires neither additional infrastructure deployment nor control of the DNS recursive servers. After describing the methodology, we present an extensive measurement validation of Queen's accuracy. Our validation shows that Queen's accuracy is reasonably high and, in particular, significantly better than that of iPlane for packet-loss rate estimation.

Keywords: Recursive DNS, Retransmission Pattern, Loss Rate.

1 Introduction

End-to-end packet loss-rate and delay are fundamental network metrics, both of which impact the performance of network applications. Estimate of delay and packet-loss rate between arbitrary Internet hosts is critical for many large-scale distributed applications, including overlay routing, P2P media streaming, VoIP, and edge-server location in CDNs. Applications can measure latency and loss rate by passive monitor or active probe. For example, Akamai's EdgePlatform [3], deployed in 70 countries, continually monitors Internet traffic, trouble spots, and overall network conditions. Ensuring high-quality service to its customers, this monitoring is an indispensable component of Akamai. RON [4] measures latency and loss rate continuously, and switches overlay paths accordingly. However, active probing often requires control of at least one end of the path, which imposes a significant coverage limitation. In academia, researchers are generally limited to paths imposed by the PlanetLab platform. Due to deployment and maintenance costs, large commercial entities are limited to modest-scale deployments of measurement platforms; for example, Keynote's platform is only available in 240 locations [8]. To overcome these limitations, several schemes have been developed to provide delay estimates without access to either end of an Internet

S.B. Moon et al. (Eds.): PAM 2009, LNCS 5448, pp. 57–66, 2009.

path. King [2] leverages DNS infrastructure to measure latency between arbitrary end hosts. Network coordinate systems construct virtual coordinate spaces from limited latency measurements, and then make latency predictions based on the virtual coordinates [7,9,10]. Azureus builds one of such latency estimation scheme into its production P2P client [10], in order to make better peer selection.

Although there is significant research on delay estimation between arbitrary end hosts, there has been relatively little work on packet-loss rate. To our knowledge, the only existing published methodology for estimating packet-loss rates between arbitrary end-hosts is iPlane [1], which also measures other metrics. It predicts the loss rate between arbitrary hosts by composing the performance of measured segments of Internet paths. In this paper, we present *Queen*, a new methodology for estimating packet-loss rates between arbitrary hosts. *Queen* requires neither additional infrastructure deployment nor control of end hosts.

Queen builds upon the well-known latency measure tool King [2]. *Queen* approximates packet loss rates between arbitrary hosts by finding two DNS servers near them and determining the packet loss rates between these two DNS servers. However, although *Queen* gets its initial inspiration from King, it is nevertheless very different – its design has required a deep understanding of how currently deployed DNS servers operate. In particular, we have discovered that all DNS servers are configured with highly regular retransmission mechanisms, allowing packet loss to be inferred from observed excessive latencies. Because the gap in DNS retransmissions is large and regular, *Queen* is accurate even though latencies can vary wildly between end systems over short period of time.

The contribution of this paper is as follows: *(i)*: We develop a new methodology that estimates packet loss rate between arbitrary Internet end hosts without control on either end. We first characterize the retransmission behavior of deployed DNS servers; we then propose a loss-rate formula based on this behavior. *(ii)*: Based on the methodology, we develop a tool, *Queen*, which is made public at *http://cis.poly.edu/∼angelawang/projects/lossrate.htm*. *(iii)*: We conduct extensive measurements to validate the accuracy of *Queen*. In particular, we show that Queen is more accurate than iPlane for estimating packet-loss rates. *(iv)*: As a case study, we perform an Internet-wide packet-loss rate measurement. The results are informative and can also provide realistic Internet characteristics to other platforms, such as Emulab [11].

The rest of the paper is organized as follows. In Section 2, we briefly review how King works, then we present the design of *Queen* in Section 3 and evaluate its accuracy in Section 4. In Section 5, we present an Internet-wide experiment result. Afterwards, we present related work and conclusion in Section 6 and 7.

2 Brief Review of King

King, developed by Gummadi et al. [2], is a methodology to estimate latency between arbitrary Internet hosts. They propose a simple version that requires no external setup and a somewhat more complex one with much improved accuracy and additional setup. We only review the latter version for this work.

To measure the latency between arbitrary end hosts A and B, King (*i*) finds DNS name servers that are topologically close to A and B (say R and T, respectively); (*ii*) estimates the latency between the two DNS name servers R and T using DNS queries; and (*iii*) uses the measured latency between R and T as an estimate of the latency between A and B (see Figure 1).

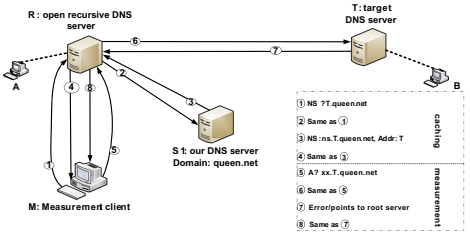


Fig. 1. King measures the latency between R and T by *tricking* R to directly query T

The key step is to estimate the latency between R and T. This requires at least one of the servers to be an *open recursive DNS server*, that is, a DNS server allowing recursive DNS queries from *arbitrary* Internet hosts. Henceforth, assume that R is such an open recursive server. The important issue is how to "trick" R to send DNS queries *directly* to T in order to measure the latency between them. The key idea is: (*i*) first trick R into believing that T is the authoritative name server for a special domain; and (*ii*) then query R for a host in this domain, which will trigger R to forward the DNS query directly to T. We refer to steps (*i*) and (*ii*) as the caching and measurement stages, respectively.

For the caching stage, we need operate an authoritative DNS server (call it S1) for a special domain (say queen.net). A measurement client M (arbitrary) sends to R a recursive NS type query containing encoded T's IP address for a sub-domain (say t.queen.net), which will ultimately be answered by S1, the authoritative DNS server for domain queen.net. S1 is programmed to reply that T is the authoritative DNS server for sub-domain t.queen.net, and this reply will be cached at R. This completes the caching stage. From this point on, any subsequent recursive query through R for a host belonging to sub-domain t.queen.net will be forwarded to T directly. Since T is actually *not* the authoritative name server for this sub-domain, it will return an error message (or a pointer to the root DNS servers). R will in turn report query failure to M. Thus, R is tricked into querying T directly, and the latency between them can be estimated.

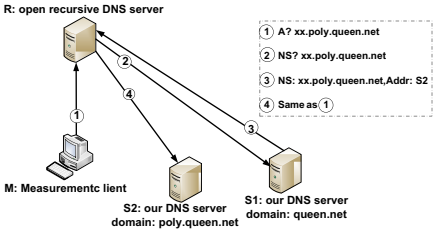
3 Methodology

DNS queries are transmitted using UDP by default. An interesting question is: What happens if there is packets lost (either query or response) between the two representative name servers R and T? This leads us to a new methodology for estimating packet-loss rates between arbitrary Internet end hosts.

3.1 Retransmission Pattern

When a DNS server sends a query to another DNS server, either the query or the response could get lost along the path. All DNS servers have built-in mechanism to deal with such losses. In either case, if the querying name server does not receive response within a certain period of time, it will resend the query until a retry limit is reached. Intuitively, intervals between retransmissions in most DNS servers should be substantially larger than RTT of most paths. We now confirm this intuition by studying the retransmission patterns of DNS servers.

Architecture. To dig a DNS server's retransmission pattern, we force the server to resend queries until its retry limit exhausted. Fig. 2(a) shows our architecture. We operate two name servers, S1 and S2, which are configured as: 1) S1 is the authoritative name server for domain queen.net, and delegates sub-domain poly.queen.net to S2; 2) S2 runs as a simple DNS black hole, which only records the received time of all incoming queries but does *not* reply to any query. When client M sends to R a recursive DNS query for a host (say host.poly.queen.net) in the sub-domain, R will be redirected to S2 after first contacting S1. Since S2 never replies, R will resend the query until exhausting its retry limit. We encode R's IP address inside the query (together with a unique identifier), so that S2 can easily extract the address, as well as match queries. Finally, we collect the timestamps of all queries from each R, and calculate the retransmission pattern.



(a) Method to measure a name server's retransmission pattern

Pattern	# of Servers	% of Total
2-2-8-16	15255	51.1
2-2-8-16-30-30	7600	25.4
4-8	2106	7.0
0.5-0.5-1-2-4-8-10	1471	5.0
2-2-8	823	2.8
2-2-8-16-30	355	1.2
4-8-16	148	0.5
2-8-16	146	0.5

(b) Most Common Retransmission Patterns of DNSs

Fig. 2. Retransmission pattern measurement

Experiment. We setup an experiment to discover the retransmission patterns of ∼30,000 DNS servers picked from a large list of unique open recursive DNS servers with wide coverage, obtained in our previous study [6]. Those servers cover 6 continents and 147 countries. For each name server R, we send a unique recursive query from our measurement client M. The retry at M is set to 0 to ensure that exactly one query is sent to each R and all the duplicate queries recorded at S2 are generated solely by R. We found that there are small number of common retransmission patterns among DNS servers, showed in Fig. 2(b). Those patterns cover 93.5% measured servers. As an example, pattern 2-2-8-16 means the server will retry 4 times, 1st after 2 seconds timeout, 2nd after another 2 seconds timeout, 3rd after another 8 seconds, and 4th after another 16 seconds. Furthermore, about 94% servers will wait at least 2 seconds before retry.

3.2 Loss Rate Estimation

We are now ready to describe how to infer packet losses from large latencies, and how to estimate the packet-loss rate.

Packet Loss Definition. We will use an example to explain how to infer packet losses from large latencies. In Fig. 3, we measure the latency between two name servers R (in S. Africa) and T (in Seattle) using DNS queries with exponential inter-arrival time, where the average is 200ms and their associated counting process is Poisson, over a 15-minute duration. Here, R is an open recursive name server with retransmission pattern 2-2-8-16s. We can see most latencies fall into a range (490-500ms) – the regular RTTs between R and T. However, there are some latencies far larger than the regular ones. We infer that these large latencies correspond to packet losses. Specifically, we compute a latency threshold based on R's retransmission pattern. Because the regular RTT is about 500ms and R sends out its first retransmission after 2 seconds if there is no response, we set the threshold to be 500ms+2s. Thus, any latency around 2500ms implies one packet loss between R and T, either on forward path or reverse path. Similarly, any latency around 500ms+2s+2s implies two packet losses, and so on.

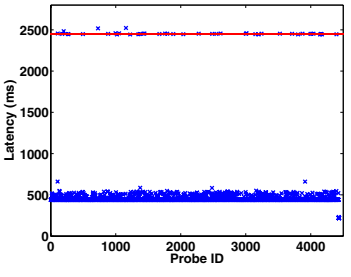


Fig. 3. Packet Loss Example

Loss Rate Computation. Now we know whether there is a packet loss from the measured latency. Next is how to compute the packet loss rate exactly. Suppose we send out N total queries, where M queries receive response and obtain the latencies, while the other $(N - M)$ queries have failed (without responses). To compute the total number of lost packets, we consider those M queries only (the reason to be explained later). With the loss definition in previous section, we can infer how many queries (say L out of M) have excessive latencies, as well as the total number of retransmission packets L' (clearly, $L' \geq L$). Then, the packet loss rate is simply calculated as $L'/(M + L')$.

So far, we have presented our methodology to estimate the packet-loss rate from excessive DNS query latencies. However, what's the effect if packets get lost between M and R, or R and S1 in Fig 1? The facts are: 1) if the packet is lost between M and R, no matter forward or reverse, M will not receive response at all. This is exactly the reason to exclude those failed queries when computing

the loss rate; 2) Packet losses between R and S1 will be automatically taken care of by R's retransmission mechanism. In addition, it happens during the caching stage, so the excessive latency here will *not* affect the measurement stage at all. Thus, the losses inferred in our method will only be the losses between R and T.

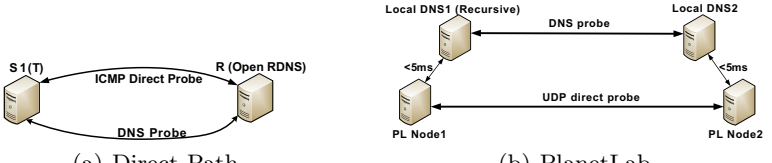
Another potential problem is, when T returns an error to R, depending on its configuration, R may have two other outlier actions: 1) It may retransmit the same query several times. In this case, even without packet loss, the estimated latency will be a factor of K larger, where K is the number of retransmissions. It will introduce noise to the real loss. 2) Alternatively, it may stop forwarding further queries for the same sub-domain to T. If so, R will not contact T directly any more after an error, which causes the algorithm to fail. Fortunately, we can easily point our own DNS server as T and send trial DNS queries to examine all open recursive DNS servers (R's). By parsing the query logs on our own DNS server, we can identify which R's behave as outliers and simply filter them out in future measurements. Based on our observation, there are only few such outlier-behaving DNS servers; majority can handle exception normally.

4 Validation

In this section, we present quantitative validation of the accuracy of *Queen*.

4.1 Direct Path Validation

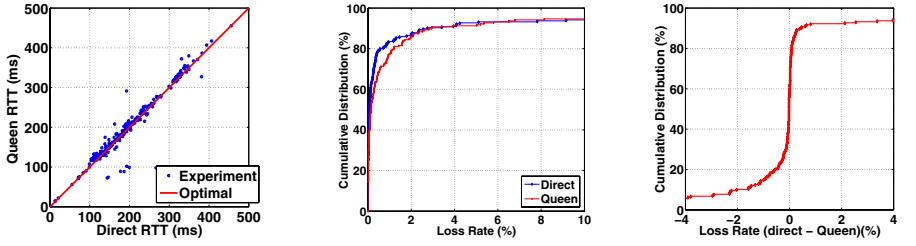
First, we fix the target T to be S1. We trick open recursive DNS servers (R's) to query S1 and estimate the packet loss rate between them, as shown in Fig. 4(a). We send out DNS queries with exponential inter-arrival time and 200ms in average over a 15-minute duration, so that their associated counting process is Poisson. In parallel, we also run direct probing by sending ICMP packets from S1 to R, which serves as ground truth loss rate. Note that, in this validation, Queen is estimating the packet loss rate on exactly the same path as the direct probing.



(a) Direct Path (b) PlanetLab

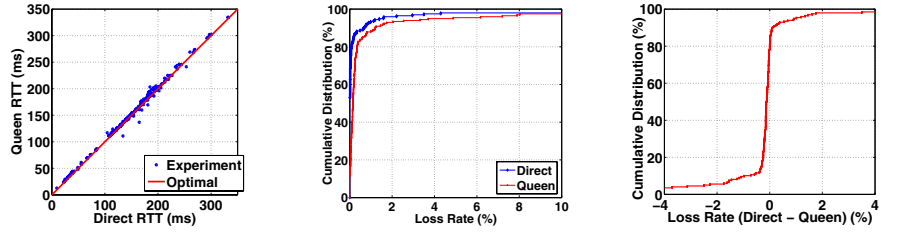
Fig. 4. Validation Path Setup

We randomly choose 370 open recursive name servers from 5 continents. In the end, we get results for ∼330 paths, where 210 experience loss either in direct probing or Queen. Fig. 5(a) compares two latencies. Optimally, they will align at 45° straight line. As we can see they indeed match very well. Fig. 5(b), 5(c) compare the loss rate estimated by two methods. They also match very well. In particular, the absolute loss rate difference between direct probing and Queen is



(a) Latency Comparison (b) LossRate Comparison (c) LossRate Difference

Fig. 5. Direct Path Validation Results



(a) Latency Comparison (b) LossRate Comparison (c) LossRate Difference

Fig. 6. PlanetLab Validation Results

within 1% for more than 80% of paths. Finally, it appears that iPlane does *not* return loss rate for any of the paths in this experiment. Apparently, these paths are not covered by its database.

4.2 PlanetLab Validation

In this set, we use PlanetLab (PL) to conduct validation, as in Fig. 4(b). Similar to 4.1, we again run two kinds of probes in parallel with the same pattern. The difference is: in direct probing, we run a UDP probing client at one PL node and a UDP echoing server at another PL node, to get the ground truth loss rate. We locate two nearby corresponding DNS servers (e.g. no more than 5ms away), one for each PL node, and estimate the loss rate between them. We compare loss rate estimated by both methods to see whether they match each other. In this set of validation, the path whose loss rate is estimated by Queen is slightly different from the path estimated by direct probing – the former path is between the two name servers while the later one is between the two PL nodes.

We choose 5 PL nodes with recursive local DNS servers as sources. Each source picks ∼70 target PL nodes from different continents. We get results for ∼260 paths, where 200 show loss either in UDP probing or Queen. Fig. 6 depicts the results. Again, two latencies and loss rates match very well. In particular, the absolute loss rate difference between two methods is within 1% for more than 85% paths. At the meantime, iPlane returns *zero* loss rates for all the paths in this experiment, which is quite far from both Queen and the ground truth.

Previous study [15] suggested that small 40-byte probes tend to experience less losses than large 1000-byte probes. Since query packets generated by Queen can be at most 280 bytes (limited by the maximum length of DNS hostnames, which is 255), it raises a concern that Queen might under-estimate the true loss rate. To study this issue, we re-run the same validation in parallel with different probing size of 80B, 160B and 240B. We observe that packet size in fact has very little effect on loss rate. In addition, we compare direct UDP probes with size of 80B, 240B and 1200B (very large, close to MTU) and the packet size also has very minimum effect(details skipped due to space constraint).

5 Experiment

After validating that our method has reasonably high accuracy, we conduct a loss rate measurement for a large geographic area with world-wide coverage.

5.1 Measurement Setup

We pick one server for each country from our open recursive DNS server list and measure the loss rate between each pair of servers. The final data set covers 6 continents and 147 countries, with 10,731 paths in total. The complete measurement involves a large number of paths. On each path, Queen sends query probes following exponential inter-arrival time with average 500ms for 15 minutes, so that their associated counting process is Poisson. To speed up the measurement process, we have developed a distributed execution platform, which splits the complete task into many smaller jobs, spreads these jobs onto PL nodes and executes them in parallel. This platform helps to complete the measurement quickly (e.g., with 300 PL nodes, 10,000 paths, each node only needs to run 33 jobs, and the entire task takes slightly more than 8 hours to complete).

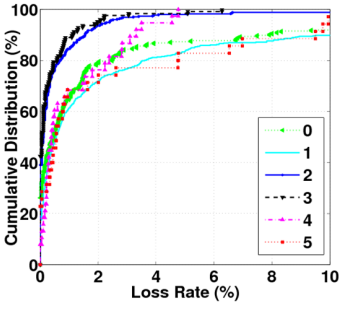
5.2 Summary of Results

We group the sampled servers by continent and analyze loss rates within/cross continents. Fig. 7(a) shows the loss rate statistics within each continent, and Fig. 7(b) cross continents. Some results are intuitive – North America and Europe have low loss rates, no matter intra-continent or cross-continent. This is clearly due to good networking infrastructure with the two regions and connectivity between them. In addition, North America and Europe always have lower loss rates within the continent than cross to the other continent. Not as intuitive though, we also observe that, for other continents, the loss rates are in fact lower cross to North America or Europe than within the continent itself. This, we believe, reflects the fact that North America and Europe are currently the hubs of the Internet.

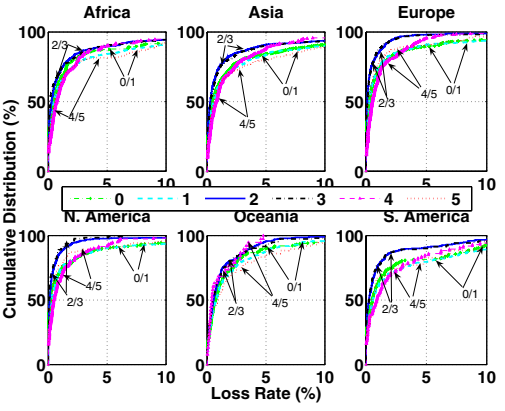
6 Related Work

The study of Internet packet loss rate can be dated back to more than a decade ago. It is conducted to understand Internet itself, as well as the impact on

Continent	# of Countries	Continent	# of Countries
Europe	43	Asia	37
Africa	25	N. America	20
S. America	12	Oceania	10



(a) Intra-Continent Loss Rate



(b) Cross-Continent Loss Rate

Fig. 7. Continent Loss Rate Statistics. Numbers 0-5 represent 6 continents, respectively. *0-Africa, 1-Asia, 2-Europe, 3-N.America, 4-Oceania, 5-S.America.*

the performance of applications [13,14]. Constant efforts are continuously being pushed to improve the accuracy of packet loss rate estimation [12]. Tools [15] are developed to use loss rate to troubleshoot path failures. All these work rely on sending out active UDP/ICMP probes. Hence, they require controlling of either one or both ends of the path being studied.

iPlane [1] is the only other tool *close* to be able to estimate packet loss rate without requiring access to either end. It constructs an annotated map of the Internet by: (*i*) sending probes from a large number of various vantage points, such as PlanetLab nodes and traceroute servers; (*ii*) clustering interfaces into PoPs based on response source address or returned TTLs to all the vantage points. By collecting all probes and processing the measurement data, it characterizes the loss rate of all inter-cluster links in the measured topology. Then, it may indirectly predict packet loss rate between a pair of end hosts by compounding the packet loss rate of each segment link along the path. However, iPlanes's coverage is limited as it can not provide packet loss rate on a path if neither end of the path exists in the database. Thus, it doesn't really provide loss rate between arbitrary two end-hosts, as it still requires contributions from one end. In addition, it does not perform measurement on demand.

7 Conclusion

In this paper, we presented *Queen*, a tool that estimates loss rate between arbitrary Internet end hosts without control of either side. We validate *Queen* with two different data sets. They all show that our method has reasonably high accuracy. We used *Queen* for an Internet-wide experiment, which provides informative results and realistic Internet characteristics.

References

1. Madhyastha, H.V., Isdal, T., Piatek, M., Dixon, C.: iPlane: An Information Plane for Distributed Services. In: USENIX OSDI (2006)
2. Gummadi, K.P., Saroiu, S., Gribble, S.D.: King: Estimating Latency between Arbitrary Internet End Hosts. In: ACM SIGCOMM IMW (2002)
3. EdgePlatform, Akamai Inc., `http://www.akamai.com/html/technology/index.html`
4. Andersen, D.G., Balakrishnan, H., Kaashoek, M.F., Morris, R.: Resilient Overlay Networks. In: ACM SOSP (2001)
5. Yang, H.-Y., Lee, K.-H., Ko, S.-J.: Communication quality of voice over TCP used for firewall traversal. In: ICME (2008)
6. Huang, C., Wang, A., Li, J., Ross, K.W.: Understanding Hybrid CDN-P2P: Why Limelight Needs Its Own RedSwoosh. In: NOSSDAV (2008)
7. Ng, T.S.E., Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches. In: IEEE INFOCOM (2002)
8. Keynote Global Test and Measurement Network, Keynote Inc., `http://www.keynote.com/company/keynote_network/methodology.html`
9. Dabek, R., Cox, R., Kaashoek, M.R., Morris, R.: Vivaldi: A Decentralized Network Coordinate System. In: ACM SIGCOMM (2004)
10. Ledlie, J., Gardner, P., Seltzer, M.: Network Coordinates in the Wild. In: USENIX NSDI (2007)
11. White, B., et al.: An Integrated Experimental Environment for Distributed Systems and Networks. In: USENIX OSDI (2002)
12. Sommers, J., Barford, P., Duffield, N., Ron, A.: Improving Accuracy in End-to-end Packet Loss Measurement. In: ACM SIGCOMM (2005)
13. Paxson, V.: End-to-end Routing Behavior in the Internet. In: ACM SIGCOMM (1997)
14. Bolot, J.C.: End-to-end Packet Delay and Loss Behavior in the Internet. In: ACM SIGCOMM (1993)
15. Mahajan, R., Spring, N., Wetherall, D., Anderson, T.: User-level Internet Path Diagnosis. In: ACM SOSP (2003)