KadCache: Employing Kad to Mitigate Flash Crowds and Application Layer DDoS Attacks Against Web Servers

Jie Yu*†,
*Department of Computer Science, National University of Defense Technology, China
yj@nudt.edu.cn

Liming Lu†
†Department of Computer Science, National University of Singapore, Singapore
luliming@comp.nus.edu.sg

Zhoujun Li‡
‡School of Computer Science and Engineering, Beihang University, China
lizj@buaa.edu.cn

ABSTRACT

Flash crowds or application layer DDoS attacks can severely degrade the availability of websites. Peer-to-peer (P2P) networks have been exploited to amplify DDoS attacks, but we believe their available resource, such as distributed storage and network bandwidth, can be used to mitigate both flash crowds and DDoS attacks. In this poster, we propose a server initiated approach to employing the P2P network as a distributed web cache, so that the workload directed to web servers can be reduced. The experiment using Kad demonstrates the feasibility and robustness of our approach. The latency is comparable to normal direct access to web servers, and the web contents cached in Kad remain reachable despite of the dynamic departure of peers.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: General—*Distributed Systems*

General Terms

Flash Crowds, Application Layer DDoS, Mitigation

1. INTRODUCTION

Flash crowd, or an unexpected surge in visitors to a website, occurs because of a sudden increase in the popularity of the website. It have been the bane of many web masters. On the other hand, application layer DDoS attack with the malicious purpose to block out benign users, sends out an overwhelming amount of requests. The attack requests follow the communication protocol and are indistinguishable from legitimate requests in the network layer. In addition, DDoS attackers are increasingly moving towards surreptitious attacks that hide in flash crowds of websites. Both flash crowds and application layer DDoS attacks are unstable, bursty, huge in traffic volume and difficult to prevent.

Recently there are growing interests in researching the peer behavior, performance and security of Kad, which is a structured P2P network. Among which, Kad has been shown empirically that it can be misused to perform DDoS attacks [3]. However, we believe using its available resources, such as distributed storage and network bandwidth, Kad or P2P networks in general, can be used to mitigate both flash crowds and DDoS attacks. In this poster, we propose to exploit Kad as a "distributed cache", namely *KadCache*, to address both flash crowds and application layer DDoS attacks against web servers.

Kad has millions of simultaneous users as to date. Each peer in Kad is identified and looked up by a 128-bit ID, and they communicate through UDP messages. Kad employs an iterative lookup process to locate peers in the tolerance zone and then publishes keywords or files into them. By exploiting Kad, defenders need not deploy any additional resources solely in preparation for flash crowds or DDoS attacks, and since there are millions of online peers in Kad, it is difficult for attackers to break down KadCache. The scalability and robustness of it is guaranteed by Kad.

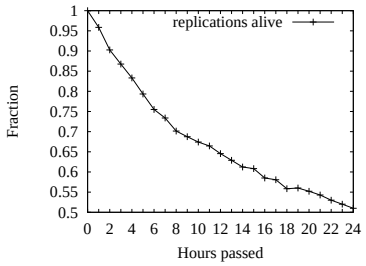
Previous works have proposed schemes named Squirrel [1] and PROOFS [2] to mitigate flash crowds based on P2P web caching. In Squirrel, objects of visited web pages are voluntarily cached by peers, and located through Pastry. PROOFS uses a centralized server to maintain a list of peers' neighbors. Peers are allowed to query only their neighbors for cached objects or iteratively query neighbors' neighbors. Both these schemes require clients' initiatives in caching and lookup for objects; whereas our design is server initiated, in which servers proactively publish to peers the updated web contents and respond to clients with the relevant peers' information for locating the cached objects. The server initiated approach not only simplifies client's operations in object location and thus reduces the latency, but also increases the probability of finding objects compared to having peers cache only the objects they have visited.

2. DESIGN OF KADCACHE

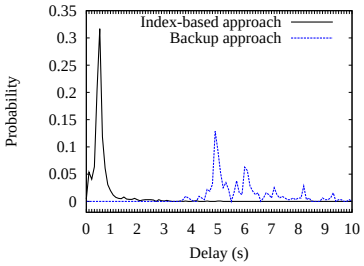
In this poster, we only consider caching the static web contents, e.g. static HTML web pages, images or javascripts, which we all call *objects*. When an object is created or modified on a protected web server, the mitigation mechanism automatically publishes it into Kad network. Afterwards, the requests of this object will be redirected to the peers in Kad, instead of passed to the web server. To facilitate object lookup, the mitigation mechanism uses a hash table to keep track of the group of peers that store this object. It responds to clients' object requests with the information of corresponding peers.

2.1 Object Publication

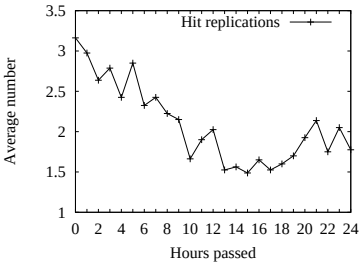
An object is published into selected peers using the keyword publishing request messages of Kad. The key in this message is the hash of the object's URL, and its metadata is the object. Peers are selected as cache if the first h bits of their IDs are the same as that of the URL's. In our implementation, we choose $h = 8$, which is the same as the default setting for the size of a tolerance zone in Kad. Since peers in Kad typically clear the stored keywords every 24 hours,



(a) Aliveness of objects cached in Kad



(b) Distribution of latency



(c) Number of reachable replications

we should republish each object at least every 24 hours.

If the hash function used here is uniform (e.g., MD5), objects will be distributed uniformly over the hash space and thus be distributed uniformly among all peers of Kad. This design makes the DDoS attack to the cached objects almost impossible. Attackers in an attempt to disrupt the availability of KadCache need to bring down all the peers storing the distributed, replicated objects. The number of peers keeping an object can vary between $[N_{min}, N_{max}]$, depending on the popularity of each object. Typically, $N_{min} = 10$ and $N_{max} = 1000$. With N peers keeping an object, together they can serve N times of object requests.

When the mitigation mechanism looks up peers to cache an object, it can either (i) uses standard lookup process of Kad to locate peers in the tolerance zone; or (ii) keeps track of the information of all the peers and selects peers locally; followed by using the keyword publishing messages to publish the object to the selected peers. The choice of strategy depends on the number of objects that will be published into Kad and the frequency they are updated. In our design and implementation, we choose the second strategy, since (i) most websites involved in flash crowds or DDoS attacks are large-scale sites with thousands to millions of objects, and (ii) using Kad's BOOTSTRAP process, it is easy and fast to crawl the information of all peers.

2.2 Object Location

Each server maintains a hash table to locate the objects cached in Kad. This greatly reduces the access latency, since clients need not spend time to look up which peers are caching the object. To reduce the frequency of server's lookup in the hash table, we realize server's object lookup response via URL rewriting. Normally, a webpage includes serval objects. A complete access of a webpage induces downloads of multiple embedded objects. When we publish a static webpage into Kad, its inner URLs are replaced by the information of peers that store the corresponding objects. Then all the following downloads will be done directly in Kad and need not access the web server anymore.

If the hash table listing the cached objects is unavailable, as a backup approach, clients themselves can firstly hash the URL of this object, then use the standard Kad lookup process to locate peers whose IDs are close to the hash value, and finally use the searching process to fetch the object. Furthermore, both the index-based location and the backup approach can use pipelining to download objects from distributed peers in parallel, to reduce the latency.

3. EXPERIMENT

We have monitored Kad by capturing its snapshot several times a day since Nov 2008. The snapshots were obtained by a crawler we implemented that uses BOOTSTRAP requests to traverse the Kad network. It takes 25 to 40 minutes to collect information of all Kad peers. Each snapshot we collected contains about 2.5 to 3.8 million peers in routing tables and about 1.2 to 1.8 million active peers (i.e. peers who respond to standard Kad messages).

To verify the feasibility and robustness of KadCache, we performed several experiments. Firstly, to investigate the aliveness of objects cached in Kad under churn of peers, we published objects to 1000 randomly selected active peers, and searched these objects every hour for 24 hours. The average fraction of alive replications of an object found over time is shown in Fig.(a). Due to the dynamic departure of peers or increase of the cleanup frequency, the fraction of alive replications gradually decreases over time. Since more than half of the replications exist after 24 hours, we only need to respond with several peers to make sure that the client can successfully locate an object with high possibility, e.g., 6 peers would give a success probability of over 98%. Secondly, we plot the distribution of latency for both location approaches in Fig.(b). Note that latency here means the time spent to both look up and fetch an object. In Fig.(b), the peak for index-based approach indicates that most object location takes about 500ms. This is similar to the download time of getting an object directly from the web server under no attack or flash crowd. The latency of the backup approach is typically 5 seconds, which is acceptable compared to webpage unavailability during attacks or flash crowds. Thirdly, to evaluate the reachability of objects located using the backup approach, we published each object to 10 peers and measured hourly the average number of replications found for an object for 24 hours. The result is shown in Fig.(c). Since the search is successful as long as there is a single hit, this approach is also robust on average.

4. ONGOING WORK

We are working on to secure the objects in KadCache from malicious behaviors, and to explore more robust designs of distributed web caching.

5. REFERENCES

[1] S. Lyer, A. Rowstron, and P. Druschel. Squirrel: A decentralized peer-to-peer web cache. In *Proc. of PODC'02*, 2002.

[2] A. Stavrou, D. Rubenstein, and S. Sahu. A lightweight, robust P2P system to handle flash crowds. *IEEE Journal on Selected Areas in Communications*, 2004.

[3] J. Yu, Z. Li, and X. Chen. Misusing Kademlia protocol to perform DDoS attacks. In *Proc. of ISPA'08*, 2008.