Data Caching in Ad Hoc Networks using Bloom Filters

Julinda Taylor

Department of Electrical Engineering and Computer Science, College of Engineering

Abstract. Data caching provides efficient data access by maintaining replicas of data in strategic parts of the network. However, current research in this area does not manage memory space of each node efficiently. We propose an improvement by considering Bloom filters, a fast, space-efficient probabilistic method for looking up data. We compare the system the system performance with and without Bloom filters and show the performance is very close, even though the Bloom filter only takes half the space of the nearest cache table.

1. Introduction

Ad Hoc networks are multi-hop wireless networks of computing devices with wireless interfaces. The computing devices could be conventional computers (e.g., PDA, laptop, or PC) or backbone routing platforms, or even embedded processors such as sensor nodes. Caching is a method of storing data in multiple places on the network, typically to reduce response time and network traffic.

The authors in [1] proposed a benefit based data caching technique in ad hoc networks to increase the efficiency of data access. It is a distributed technique wherein each node decides whether to cache passing-by data by observing the local traffic load. It achieves good system performance in terms of access query delay, query success ratio, and total number of messages. However, benefit based data caching is not space efficient. A Bloom filter [2, 4] is a well known randomized data structure for representing a set to support membership queries. We show that Bloom filters can improve the space efficiency of the existing work while not affecting the network performance.

2. Benefit Based Data Caching and Bloom Filter

Benefit based data caching: Authors in [1] consider the cache placement problem of minimizing total data access cost in ad hoc networks with multiple data items and nodes with limited memory capacity. This optimization problem is NP-hard. Defining benefit as the reduction in total access cost, they present a polynomial-time centralized greedy algorithm that provably delivers a solution whose benefit is at least one-fourth of the optimal benefit. The centralized approximation algorithm works by iteratively selecting a data item to cache in a node that gives the highest benefit. For the distributed algorithm, to make intelligent caching decision, each node maintains a *nearest cache table* to keep track of the closest cache node of each data. The size of the nearest cache table is equal to the total number of data items in the network which is not space efficient

Bloom filter: Consider a set $A = \{a_1, a_2, ..., a_p\}$ of p elements. Bloom filters describe membership information of A using a bit vector V of length m. For this, k hash functions, $h_1, h_2, ... h_k$ with $h_i : X \to \{1..m\}$ are used. If a_i is a member of A, in the resulting Bloom filter V all bits obtained corresponding to the hashed values of a_i are set to 1. Testing for membership of an element is equivalent to testing that all corresponding bits of V are set. Thus *false positives* in membership queries exist; that is, queries might incorrectly recognize an element as member of the set.

One prominent feature of Bloom filters is that there is a tradeoff between the size of the filter and the rate of false positives. Observe that after inserting n keys into a filter of size m using k hash functions, the probability that a particular bit is still 0 is:

$$p_0 = \left(1 - \frac{1}{m}\right)^{kn} \approx 1 - e^{-\frac{kn}{m}}.$$

Hence, the probability of a false positive (the probability that all k bits have been previously set) is:

$$p_{err} = (1 - p_0)^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$$

2. Experiment, Results, Discussion, and Significance

We use ns-2 [5], a network simulator to simulate the ad hoc network performance. There are 100 ad hoc nodes

and 1000 data items in the network. Each node accesses each data item with some access frequencies.

We first compare the false positive ratio (FPR) between the simulation result and the theoretical result for Bloom filter. In simulation, the FPR is measured as ratio of the total number of false positive query replies divided by the total number of query replies. We vary the size of the bloom filter as 400, 600, 800, and 1000. Figure 1 shows that the simulated and theoretical results are very close.

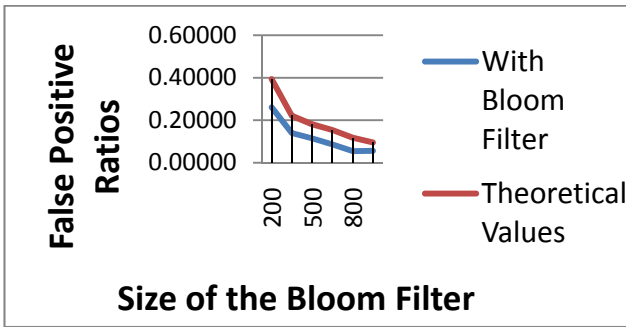

Figure 1. The FPR simulation results vs. the theoretical calculated results for the Bloom Filter

We then compare the network performance with and without bloom filter. Here we vary the query generating time of each node as 3, 5, 10, 20, 30, 40 seconds. We set the bloom filter size as 500. Figure 2 shows such comparison in terms of the number of messages, the average query delay, and the query success ratio. It shows the performance with bloom filter is very close to that without bloom filter, even though bloom filter only has half size of the space as the nearest cache table. To calculate the theoretical result we assume n is equal to 100 which is the maximum number of data items the data can cache. However in the simulation, not all the nodes cached 100 items..

3. Conclusions

Data caching provides efficient data access by maintaining replicas of data in strategic parts of the network. However, current research in this area does not manage memory space of each node efficiently. We propose an improvement by considering Bloom filters, a fast, space-efficient probabilistic method for looking up data. Using ns2, a popular network simulator, we compare the system performance of data caching in ad hoc networks with and without bloom filters. We compare the system the system performance with and without Bloom filters and show the performance is very close, even though the Bloom filter only takes half the space of the nearest cache table..
Figures 2.1 -2.3 demonstrate the efficiency of the Bloom filter compared with the benefit caching.

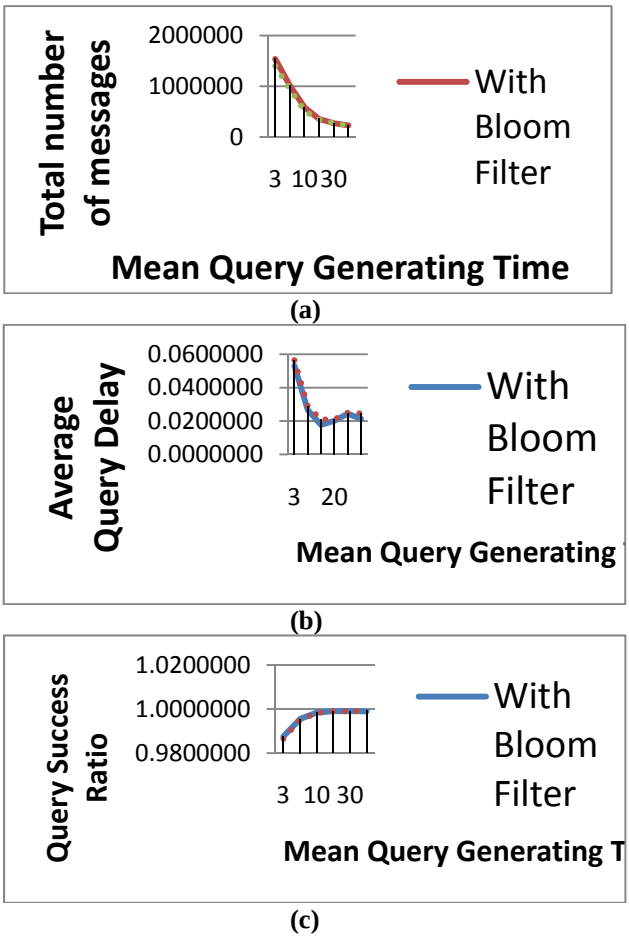

Figure 2. The system comparison between with and with out Bloom filter: (a) total number of messages, (b) average query delay, (c) querry success ratio.

4. Acknowledgements

I would like to thank Dr. Bin Tang for all of his help and guidance. I would also like to thank the Department of Electrical Engineering and Computer Science for their support and assistance in this project.

[1] Bin Tang and Himanshu Gupta and Samir Das, Benefit Based Data Caching in Ad Hoc Networks, IEEE Transactions on Mobile Computers, Vol 7, No 3, March 2008.

[2] Wing Ho Yuen and Henning Schulzrinne, Improving Search Efficiency Using Bloom Filters in Partially Connected Ad Hoc Networks: A Node-Centric Analysis, Department of Computer Science, Columbia University, New York, NY 10027.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[4] ns2: http://www.isi.edu/nsnam/ns.