Computer Networks 58 (2014) 180–191

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Optimized hash for network path encoding with minimized false positives

Laura Carrea [a,*], Alexei Vernitski [b], Martin Reed [a]

[a] School of Computer Science and Electronic Engineering, University of Essex, UK
[b] Department of Mathematical Sciences, University of Essex, UK

ARTICLE INFO

Article history:
Received 16 July 2012
Received in revised form 29 July 2013
Accepted 2 September 2013
Available online 12 September 2013

Keywords:
Bloom filter
Random data structure
Packet forwarding
Information centric networks
Multicast

ABSTRACT

The Bloom filter is a space efficient randomized data structure for representing a set and supporting membership queries. Bloom filters intrinsically allow false positives. However, the space savings they offer outweigh the disadvantage if the false positive rates are kept sufficiently low. Inspired by the recent application of the Bloom filter in a novel multicast forwarding fabric, this paper proposes a variant of the Bloom filter, the *optihash*. The optihash introduces an optimization for the false positive rate at the stage of Bloom filter formation using the same amount of space at the cost of slightly more processing than the classic Bloom filter. Often Bloom filters are used in situations where a fixed amount of space is a primary constraint. We present the optihash as a good alternative to Bloom filters since the amount of space is the same and the improvements in false positives can justify the additional processing. Specifically, we show via simulations and numerical analysis that using the optihash the false positives occurrences can be reduced and controlled at a cost of small additional processing. The simulations are carried out for in-packet forwarding. In this framework, the Bloom filter is used as a compact link/route identifier and it is placed in the packet header to encode the route. At each node, the Bloom filter is queried for membership in order to make forwarding decisions. A false positive in the forwarding decision is translated into packets forwarded along an unintended outgoing link. By using the optihash, false positives can be reduced. The optimization processing is carried out in an entity termed the Topology Manger which is part of the control plane of the multicast forwarding fabric. This processing is only carried out on a per session basis, not for every packet. The aim of this paper is to present the optihash and evaluate its false positive performances via simulations in order to measure the influence of different parameters on the false positive rate. The false positive rate for the optihash is then compared with the false positive probability of the classic Bloom filter.

1. Introduction

Bloom filters are very good data structures to represent concisely a set in order to support membership queries [1]. They are randomized data structures since they require the employment of hash functions for their construction (we will describe them in details in Section 2). Consequently, they have some probability of giving false positives when queried; that is, an element may appear to belong to the set when in fact it is not. Because of their succinct size, they have become very popular for network application. As Broder and Mitzenmacher have pointed out in [2], "there are many places in the network where one might like to keep or send a list, but the complete list would require too much space". In fact, Bloom filters offer a representation which significantly reduces space at a cost of false positives. For many applications false positives

* Corresponding author. Tel.: +44 7831784094.
E-mail address: lcarrea@essex.ac.uk (L. Carrea).

1389-1286/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2013.09.002

can be tolerated or can be made low enough so that the space saving offered by the Bloom filter compensates for the probability of errors. Since Bloom filters have became very popular, having a broad range of applications, many variants of the Bloom filter have been proposed to optimize the data structure with regard to the false positive issue. A thorough survey on Bloom filter variants can be found in [3].

One of the most recent application of Bloom filter in networking is for in-packet forwarding [4,5]. Contemporary packet forwarding techniques generally fall into two categories: either destination based forwarding utilizing switch or routing tables as in Ethernet or IP; or label swapping techniques as used by MPLS or ATM. However, with recent interests in alternative network architectures, such as content centric networking [6] or information centric networking [7], it is of interest to consider alternative forwarding strategies that might be more suitable for these new architectures or bring more general improvements. One such strategy is to use a Bloom filter [1] as a fixed header identifier that encodes a complete network path in an space efficient manner. While this could be used to replace a strategy such as MPLS labeling [8], it could also be used to implement a forwarding layer replacing IP in a clean-slate network architecture. False positives are a problem that is inherent to Bloom filters. In the case of network path encoding, a false positive means that traffic may pass over links that were not intended to be part of the path. Thus, false positives may cause bandwidth wastage or may generate a loop [9]. Although this may be a very rare event, it can cause major problems in the network. Hence, there is an interest in minimizing the false positive occurrences or being able to control them for this application while keeping the length as small as possible, since the Bloom filter will occupy part of the packet header. One of the basic design variants of the Bloom filter is its length, a longer filter has generally lower false positives. Consequently, in a given application the length can be adjusted to suit a particular false positive rate. However, for this application the length of the Bloom filter is a critical parameter that cannot easily increased.

In this paper, we present the optihash, a new variant of the Bloom filter. When compared to the Bloom filter, we show that it offers an optimization mechanism which reduces the false positives occurrences or alternatively reduces the data structure length for a given false positive rate. If false positive occurrences cannot be eliminated, the mechanism offers the possibility to choose which element would result in a false positive as would best suit a particular network application. The performance of the optihash is tested for the new forwarding mechanism which has been proposed within the framework of the project PSIRP (Publish/Subscribe Internet Routing Paradigm).[1] This was a substantial effort, which aimed to re-design the whole Internet architecture above the physical layer. For this type of application, it may be useful to generally control the false positive occurrences, not necessarily to simply minimize them. The optihash offers this flexibility. However, the implementation of the optihash for in-packet forwarding is out of the scope of this paper.

The rest of the paper is organized as follows. In Section 2 we revisit Bloom filters and we present their application for in-packet forwarding introduced under the aegis of the PSIRP project. In Section 3 we describe the optihash, the proposed new variant of the Bloom filter. Section 4 introduces the optihash for in-packet forwarding. In Section 5 the performances of the optihash are evaluated for in-packet forwarding in a regular network for different parameters and the false positive rates are experimentally estimated. Section 6 concludes the paper.

2. Bloom filters for the multicast forwarding fabric

Since the main construction suggested in this paper is inspired by the concept of the Bloom filter, this section reviews the Bloom filter concept and the multicast forwarding scheme based on the Bloom filter. Further details on Bloom filters can be found in the literature, for example in [2].

Bloom filters [1] are space-efficient probabilistic data structures for representing sets and supporting set-membership queries. They were introduced by Bloom [1] in 1970 to represent words in a dictionary. For some time, they have been mainly used in database applications [10,11]. In the late 1990s, Bloom filters started attracting the interest of the networking-research community, especially because of their simplicity and wide applicability in aggregating data sets providing low information processing and networking costs. Surveys on network applications of Bloom filters [2,3] show employments of Bloom filters for overlay networks, data centric routing, traffic monitoring, caching, security, etc.

Recently, Bloom filters had an important role in the design of the forwarding fabric [4] of the information centric network introduced under the aegis of the PSIRP/PURSUIT projects[2] [12]. In this scheme, Bloom filters encode links and routes between nodes in a source routing fashion. In this section, the Bloom filter is introduced and its use as the forwarding fabric in PSIRP is described.

2.1. Standard Bloom filters

Consider a subset $A = \{a_1, a_2, \ldots, a_n\}$ of n elements of a fixed set (called a universe) U of N elements. If it is desired to represent A, a naïve approach would be to represent each element as a fixed length binary identifier, a vector v, of length m and simply concatenate the representation into a binary identifier of length $|A|m$. Determining if an element $a \in A$ is in the identifier can be achieved by a linear search through the concatenated elements to see if they match. An example of this naïve approach in packet forwarding is IP strict source routing, where the IP addresses of the gateways are concatenated in an option header. For a large set this is relatively inefficient but distinct and complete. A Bloom filter is an alternative representation of the

[1] http://www.psirp.org/ accessed in July 2013.

[2] http://www.psirp.org/ and http://www.fp7-pursuit.eu/PursuitWeb/ accessed in July 2013.