Future Generation Computer Systems 75 (2017) 108–127

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A Distributed Event-Based System based on Compressed Fragmented-Iterated Bloom Filters

Cristina Muñoz [*], Pierre Leone

Computer Science Department, University of Geneva, Carouge, Switzerland

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form 5 June 2016
Accepted 9 February 2017
Available online 9 March 2017

Keywords:
Distributed event-based system
Bloom filter
Wireless sensor network
Iterated hash function
Compressed bloom filter

ABSTRACT

In this research, we propose the construction of a new architecture of Fragmented-Iterated Bloom Filters to redirect events of a distributed event-based system. We introduce two novel structures of Bloom Filters: Fragmented Bloom Filters and Iterated Bloom Filters. The aim of Iterated Bloom Filters is to discard single events that do not match any subscription. Then, Fragmented Bloom Filters deal with conjunctive and disjunctive set of events. Whether a match is found at the Fragmented Bloom Filters the publication is forwarded. Our strategy is compared to the alternative one relying on Standard Bloom Filters. The results show that Fragmented Bloom Filters lead to save memory and computational resources at the membership test. Moreover, we show that there is no memory cost for dividing a Bloom Filter in smaller Bloom Filters using the same: (1) number of elements to insert and (2) probability of false positives. Then, we prove that fast hash functions required for Fragmented Bloom Filters present a lower complexity that those required for Standard Bloom Filters. Additionally, we determine that the double hashing technique does not result in a lower complexity. Afterwards, we show that the construction of a structure of Iterated Bloom Filters using an Iterated Hash Function reduces the complexity because smaller filters and less hash functions are required. Furthermore, if information is structured in a tree Iterated Bloom Filters decrease the probability of false positives. We also focus on the improvement of data exchange for updating Fragmented-Iterated Bloom Filters between nodes. The goal is to reduce data transmitted. For this purpose, we study the effect of compressing Fragmented-Iterated Bloom Filters. The main benefit of Compressed Bloom Filters is that they transmit less bits. Therefore, less bandwidth is required and the latency of the network is reduced. The choice of Compressed Fragmented Bloom Filters preserves all these positive effects by limiting the number of transmitted bits due to its flexible structure. Besides, Compressed Iterated Bloom Filters also decrease computation.

1. Introduction

The dissemination of sensing data requires the use of different sources and destinations. Typically, in an ubiquitous sensing scenario some nodes provide data and other nodes use these data as actuators. Then, a distributed event-based system may be used to exchange information. In such a system, publishers and subscribers do not have any information about each other. They depend on the event notification service to match publications with subscriptions. In distributed networks, this service is implemented using a network of brokers nodes. A broker node is any node in the network that has information about any single or set of subscriptions. Publishers must contact a broker node to route events. Similarly, subscribers rely on broker nodes to save subscriptions. The selection of broker nodes requires the use of an overlay layer on the top of the network layer. Distributed Hash Tables (DHT) [1] construct the overlay layer mapping a key to a particular node with storage location properties. Other techniques select broker nodes that isolate a part of the network as cluster heads [2]. Furthermore, brokers can be selected on a tree or a set of independent trees [3]. All these techniques need a network protocol to provide point-to-point communication on the network layer. Recently, we proposed [4] to merge the network and the overlay layers of distributed event-based systems so that no other network protocol is needed. The advantage of this strategy is that it is no necessary to maintain the network topology. The main consequence is that nodes, which do not actively participate in the system, do not keep any information about topology. This leads to save energy and computing resources in those devices. Paths between publishers, subscribers and brokers are well-defined using an efficient variation of random walks.

We propose to implement a new architecture of Bloom Filters (BFs) at broker nodes of the overlay layer proposed in [4]. Broker nodes implement Fragmented–Iterated Bloom Filters (FIBFs) at

* Corresponding author.
E-mail address: Cristina.Munoz@unige.ch (C. Muñoz).

http://dx.doi.org/10.1016/j.future.2017.02.021
0167-739X/© 2017 Elsevier B.V. All rights reserved.

each interface of communication. FIBFs save effectively a set of subscriptions that use conjunctive and disjunctive operations. When a publication arrives to a certain interface, the corresponding FIBFs are checked to decide if it has to be forwarded or not. A publication is forwarded if there are any subscribers behind that interface, which are waiting for that specific type of events.

The dispatching algorithm proposed at individual brokers works in two steps: first, a Validation Table (VT) is used to discard individual events that do not match any subscription. Second, a Routing Table (RT) for each outgoing interface is used to match event conjunctions against subscriptions. The VT reduces the number of combinations to produce conjunctions.

Our system is focused on the use of constrained devices as sensor motes. Nevertheless, the system is also suited to type of networks that require limited devices or optimization of available resources. Our approach is compared theoretically and practically, using three different wireless sensor motes, with the use of Standard BFs. The results show that complexity and memory are reduced.

Moreover, we improve the performance of Fragmented Bloom Filters (FBFs) and Iterated Bloom Filters (IBFs) when they are sent through the channel. Broker nodes of the overlay layer of our distributed-event based protocol require to update the FIBFs at each interface by contacting their broker neighbors. This behavior is similar to the one of distance-vector protocols [5] that periodically exchange topological information. In our case, instead of exchanging topological information we exchange the content of the subscriptions that can be reached through each interface of broker nodes, so that events are correctly forwarded when arriving.

The methodology followed to improve the exchange of subscriptions requires a compression algorithm. This strategy was introduced before by Mitzenmacher [6] to define Compressed Bloom Filters (CBFs). A CBF is a Standard BF of Maximum Entropy (MEBF) that has increased its size and reduced the hash functions needed per element. Then, the number of 0's and 1's is different so that the entropy is decreased. It must be taken into account that MEBF cannot be compressed. This method leads to have less bits after compression so that the bandwidth required and the latency of the network are decreased.

Firstly, Compressed Fragmented Bloom Filters (CFBFs) are theoretically analyzed. After this, CFBFs that use different sizes and CFBFs of the same size are studied. It is shown that CFBFs of the same size provide more flexibility so that its use is recommended. Then, Compressed Iterated Bloom Filters (CIBFs) are also theoretically studied and a practical case is discussed. We compare all strategies with CBFs and prove that the design of CFBFs and CIBFs is very convenient because the same benefits of CBFs are achieved and additional advantages are found.

The rest of this paper is organized as follows: Section 2 points out related work. Section 3 details the main design concepts. Section 4 presents the research problem. Section 5 describes the design of FIBFs in the event-based system. Section 6 analyzes theoretically and practically the performance of our design. Section 7 details the principles of CBFs. Sections 8 and 9 analyzes theoretically and practically the suitability of CFBFs and CIBFs. Finally, Section 10 summarizes our proposal.

2. Related work

Bloom filters [7], have been widely studied, mainly because they effectively group information. This is important, specially when dealing with constrained resources.

A few event-based systems use BFs. In Lipsin (LIne speed up Publish/Subscribe INter-networking) [8], the topology of the network is discovered. The source-route forwarding decision to redirect events use BFs to save link identifiers, which determine the outgoing links. The weakest point of this solution, is that the topology of the network must be previously discovered in order to build a tree matching publications with subscriptions. Moreover, BFs are not used with the purpose of saving subscriptions but to identify the individual links needed to build the tree.

In [9], BFs are required to discover the identification of specific subscribers. The strategy followed uses two different structures defined by BFSiena (Bloom Filter Scalable Internet Event Notification Architectures) [10]. The first structure, *bfposet*, uses BFs to define the different values of attributes related to the predicate defined by the subscription. The second one, *sbstree*, is a tree structure formed using the values defined at *bfposet*. Once all attributes of a subscription have been appropriately saved in the tree, the identification of the subscriber sits at the corresponding leaf. The main drawback of this approach, is that combinations of predicates need to be saved separately and using a certain branch of the tree. Furthermore, we do not route publications but we finally get the ID of subscribers, so that we need other protocol in order to route the publication.

Attenuated BFs have also been proposed [11] in order to attract events in a network. Nevertheless, the model works with a certain probability, so that it is not guaranteed that a publisher matches all subscribers.

Content-Centric Networking (CCN) [12,13] may be compared to our distributed-event based system because the overlay layer presented requires the content of subscribers to route messages. CCN distinguishes between two different messages: Interests and Data packets. Interests represent the requests based on content and are routed to the source nodes that are producing the content. Data packets contain the content to be forwarded.

The Internet architecture of CCN defines three data structures at each node of the network:

1. *The content store (CS):* The CS is a cache memory that stores content with the aim of reducing network bandwidth and latency.
2. *Pending Interest Table (PIT):* The PIT has two purposes. On one hand it saves the interfaces from which Interests arrive to easily forward Data packets. On the other hand it reduces network traffic. If the same Interests are received from different nodes only the first one is sent to the source of content. Then, when the pertinent Data packet arrives it is forwarded through all interfaces that demand the same Interest.
3. *Forwarding Information Base (FIB):* The FIB acts as a routing table being in charge of saving the corresponding interfaces to reach a certain source of content so that Interests can be forwarded appropriately. Typically BFs are used for the construction of the FIB.

The main difference between the system proposed in this work and CCN is that in our proposal the content (which may be compared to events) is proactive, while in CCN the content waits for a request to send the data. This implies that in our system the content is redirected when a publisher notifies an event and in CCN requests, that may be compared to subscriptions, are redirected until reaching the source of the content, that is publishers. Typically, counting BFs [14] are used to cancel a subscription. A counting BF adds a counter to each position of a BF to keep the number of times that a position has been set to 1. When a node cancels a subscription the corresponding counters of the BF are decreased. Whether a counter arrives to 0 the position is set to 0. Several CCN protocols require BFs.

In TB^{2}F (Tree-Bitmap and Bloom Filter) [15] content is saved following a tree structure. The first leaves of the tree are saved following a T-segment Tree. Then, the rest of leaves are saved using counting BFs that require some bits per position to keep track of